Fachbereich Informatik
Universität Koblenz-Landau, Campus Koblenz

Migration einer Website mit dem Referenz-Prozessmodell ReMiP

– Anwendung für die Migration der GXL-Website nach Plone –

Masterarbeit

zur Erlangung des Grades

Master of Science

im Studiengang Informationsmanagement

vorgelegt von

Johannes Caspary

Matrikelnummer 202110008

Betreuer:
Dr. Torsten Gipp,
CMS-Betreuung, Universität Koblenz-Landau

Erstgutachter:
Dr. Andreas Winter,
Institut für Informatik, Johannes Gutenberg-Universität Mainz

Zweitgutachter:
Prof. Dr. Klaus Troitzsch,
Institut für Wirtschafts- und Verwaltungsinformatik, Universität Koblenz-Landau

Koblenz, im Dezember 2006

Erklärung

Hiermit erkläre ich, dass die vorliegende Arbeit von mir selbstständig verfasst wurde und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt wurden. Mit der Einstellung dieser Arbeit in die Bibliothek bin ich einverstanden. Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.

Koblenz, den

Johannes Caspary

Kurzfassung

In der Nutzung betrieblicher Informationssysteme ist die Wahrung ihrer Aktualität und Ausbaufähigkeit ein wichtiger Aspekt. Immer wieder ergeben sich aber Situationen, in denen mit dem alten System, dem Legacy-System, diese Ziele nicht mehr erreicht werden können. Als Ausweg bietet sich dessen Migration in eine neue Umgebung an. Die Software-Migration ist als Teilbereich der Software-Technik im Rahmen der Software-Wartung anzusiedeln.

In dieser Arbeit wurde das in einer weiteren Abschlussarbeit des Instituts vorgestellte Referenz-Prozessmodell zur Software-Migration ReMiP angewandt. Ziel dieser Untersuchung ist es, das vorgestellte Referenz-Prozessmodell für den Fall der Migration einer Website praktisch umzusetzen und auf Gültigkeit zu überprüfen.[1] Interessante Fragestellungen umfassen die Vollständigkeit und Allgemeingültigkeit des Modells für Migrationsprozesse als auch Erfahrungen mit der Intensität der Durchführung der einzelnen Migrationsaktivitäten.

Als Zielumgebung der Website-Migration wird das Content Management System Plone verwendet. Im Rahmen der Arbeit wird es ausführlich beschrieben und schließlich die zu migrierende Website in die neue Zielumgebung überführt werden. Das Ergebnis dieser Arbeit ist die migrierte GXL-Website im Zielsystem Plone unter Anwendung des ReMiP. Durch diese Migration konnte der ReMiP als Referenz-Prozessmodell für Software-Migrationen erfolgreich validiert werden.

Summary

In usage of information systems the maintenance of their actuality and their extensibility is of importance. There is a constant reoccurrence of situations, when these goals cannot be supported by keeping on the old system, the legacy system. A solution for that is its migration into a new environment. The migration of software is to be considered as a subdiscipline of software engineering, more precisely, as a part of software maintenance.

In this thesis the reference migration process model ReMiP, introduced in another thesis of the institute, was applied. The goal of this examination is to bring this model to practice in a website migration project and to analyze its validity. Here, especially the subject of completeness and generalizability of the model for migration processes and the gathering of concrete experiences with the intensity of its activities is of interest.

The target environment for the website migration will be the content management system Plone. Within this thesis it will be described thoroughly. Finally, the website to migrate will be brought to its new target environment. The result of this thesis is the migrated GXL-Website in the target system Plone using the ReMiP. By this migration, the ReMiP was successfully validated as a reference process model for software migrations.

[1] Bei dieser Website handelt es sich um www.gupro.de/GXL.

Inhaltsverzeichnis

1 Einleitung

Der Einsatz der Informationstechnologie hat die heutigen Unternehmungen intensiv durchdrungen. Man kann sich kein größeres Unternehmen vorstellen, dass nicht in irgendeiner Weise auf ihre Nutzung angewiesen ist. Die Verwendungsbereiche sind vielfältig, sowohl intern als auch im Kundenkontakt wird auf informationstechnische Unterstützung aufgebaut.

Tatsächlich bildet die Informationstechnik zwar in aller Regel keine Kernkompetenz von Unternehmen, dennoch greifen fast alle Bereiche auf sie zu. In der Soft- und Hardware eines Unternehmens spiegelt sich konzentriertes Wissen über die Geschäftsabläufe wieder. Erst durch das geregelte Funktionieren der IT-Infrastruktur können Betriebe ihrer eigentlichen Tätigkeit effizient nachgehen. Somit bedeutet die Informationstechnologie eine Grundvoraussetzung für die Wettbewerbsfähigkeit, wenn nicht sogar die Existenz der Unternehmung.

Im Laufe der Zeit verändern sich Geschäftsabläufe im Unternehmen ebenso wie die technologischen Rahmenbedingungen. In beiden Fällen wird es auf lange Sicht Anpassungsbedarf an der IT-Infrastruktur des Unternehmens geben. Handelt es sich nur um geringfügige Eingriffe in das System, so kann dies im Rahmen der regulären Wartungstätigkeiten durchgeführt werden.

Je komplexer der Änderungsbedarf allerdings wird und die technologische Umwelt sich weiterentwickelt, desto schwieriger wird es, das Software-System auf einem angemessenen Stand zu halten. Allein schon durch die Steigerung der technologischen Möglichkeiten ergab sich in der Vergangenheit immer wieder die Notwendigkeit, ein altes System in eine neue Umgebung zu überführen, wie z.B. der Übergang zur objektorientierten Programmierung seit den neunziger Jahren des vergangenen Jahrhunderts [Sneed (1999), S. 4].

Software, die entworfen wurde, um die Geschäftsprozesse eines Unternehmens zu einem bestimmten Zeitpunkt abzubilden, ist nicht für die Ewigkeit gemacht. Es wird immer wichtig sein, in betriebswirtschaftlich sinnvollen Abständen das veraltete Software-System, *Legacy-System* genannt, zu überprüfen und zu aktualisieren. Mit den Alternativen und Vorgehensweisen dieser Thematik beschäftigt sich der Bereich der *Software-Migration*.

Migration deckt einen großen Teil von Umstellungstätigkeiten der Informationssysteme in Unternehmen ab. Dennoch ist nicht unter jeder Übertragung eines Legacy-Systems in ein neues System eine Migration zu verstehen. Das wesentliche Grundprinzip der Migration liegt darin, dass bei der Überführung in ein neues Zielsystem die fachliche Funktionalität nicht verändert wird [Ackermann (2005), S. 39]. Aktivitäten, die mit der Erweiterung der fachlichen Funktionalität in Zusammenhang stehen, sind nicht der Migration, sondern davon ausgelagerten Projekten zuzuordnen.

1.1 Zielstellungen der vorliegenden Arbeit

Die Software-Migration stellt ein komplexes Themenfeld innerhalb der Softwaretechnik dar. Zur wissenschaftlich fundierten Beschreibung und Systematisierung des Prozesses der Migration hat [Ackermann (2005)] eine aufschlussreiche Arbeit geliefert. Durch ihre theoretische Behandlung der Phasen, Kernbereiche und Basisbereiche existiert ein detailliertes Referenz-Prozessmodell zur Durchführung einer Migration.

Was die Arbeit von Ackermann nicht leisten konnte, war die praktische Erprobung des Vorgehensmodells. Dies ist die Aufgabe dieser Arbeit. Hierbei interessieren Fragestellungen über die *Machbarkeit des ReMiP*. Zum einen wurde überprüft, ob das Referenz-

Prozessmodell vollständig ist und für die konkrete Anwendung alle notwendigen Aktivitäten erfasst. Zum anderen konnten erste Erfahrungen mit diesem Referenz-Prozessmodell gesammelt und mit der Intensität der einzelnen Aktivitäten gesammelt werden.

Wahl des Migrationsobjekts
Als Grundlage für die Durchführung einer Migration zur Überprüfung des ReMiP wird die Migration einer Website gewählt. Hierbei handelt es sich um die offizielle Website des GXL-Projekts, das wesentlich durch das Institut für Softwaretechnik der Universität Koblenz-Landau getragen wird. An diesem Institut wird auch die Website betrieben. Als Zielsystem für die zu migrierende Website wurde das Content Management System Plone ausgewählt. Erläuterungen zu Content Management Systemen und Plone im Speziellen werden in den Abschnitten 1.4 bzw. 1.5 gegeben.

Die Migration der GXL-Website ist im Rahmen der Anfertigung einer Masterarbeit eine angemessene Lösung. Zum einen kann sie innerhalb der vorgegebenen Zeit zur Anfertigung der Abschlussarbeit durchgeführt werden, zum anderen ist selbst angesichts des überschaubaren Migrationsumfangs die Sammlung einer ausreichenden Datenbasis zur Überprüfung des Referenz-Prozessmodells möglich. Als Ergebnis dieser Arbeit soll *eine nach Plone migrierte GXL-Website* vorliegen.

Anwendbarkeit des ReMiP für das Migrationsobjekt
Der ReMiP befasst sich originär mit der Migration von betrieblichen Informationssystemen [Ackermann (2005), S. 9]. Sie beschreiben im Allgemeinen sozio-technische Systeme, in denen Maschinen und Menschen eingebunden sind [Abts / Mülder (2004), S. 12] und sind auf ein betriebliches Anwendungsgebiet ausgerichtet. Im engeren Sinne beschreiben innerhalb dieses Ganzen Anwendungssysteme den automatisierten Teil, der Hardware-, Software, Daten und Netzwerkkomponenten umfasst [Abts / Mülder (2004), S. 14].

Im betrieblichen Umfeld können Web-Technologien für verschiedene Aufgaben herangezogen werden. Beispiele hierfür sind Web Publishing (Veröffentlichung von Informationen), Web Integration (Integration von Informationen aus heterogenen Quellen), GroupWeb (gemeinsame Kommunikation, Kollaboration und Koordination innerhalb einer Gruppe) und E-Business (Vertrieb von Produkten über die Website) [Wöhr (2004), S. 29].

In der Anwendung für die GXL-Website ist die Nutzung als Web Publishing zu interpretieren. Auf diese Weise werden Forschungsergebnisse und Kontaktinformationen leicht veröffentlicht. Zwar fällt eine solche Website nicht in den klassischen Bereich betrieblicher Informationssysteme, dennoch sieht der Verfasser keine Argumente, die gegen die Anwendung des ReMiP zur Migration der GXL-Website sprechen.

1.2 Das Referenz-Prozessmodell ReMiP

Obwohl der Problemkreis der Software-Migration weitreichende Bedeutung für alle Einsatzgebiete von Informationstechnologie im Unternehmen besitzt, halten Forschung und Praxis kaum allgemein anerkannte, systematische Vorgehensmodelle bereit [Ackermann (2005), S. 16]. Überwiegend aus der Praxis existieren Ansätze, die sich mit der systematischen Auseinandersetzung mit dem Thema Migration befassen.

Im Rahmen einer Diplomarbeit beschäftigte sich Ackermann intensiv mit der Fragestellung, wie die Durchführung einer Migration wissenschaftlich fundiert in ein generisches Prozessmodell überführt werden kann [Ackermann (2005), S. 16]. Hierbei stützt Ackermann sich auf Erfahrungsberichte aus der Praxis und stellt gleichzeitig Parallelen zwischen konventionellen Software-Entwicklungsprozessen heraus. Aus dieser mehrdimensionalen Betrachtung des Problemkreises der Migration entwickelt sie unter Rückgriff auf den Rational Unified Process ihr eigenes Referenz-Prozessmodell zur Beschreibung des Vorgehens bei

Migrationen [Ackermann (2005), S. 139 ff.]. Der Reference Migration Process (ReMiP) gibt in einer ausführlichen Behandlung Vorschläge zur Beantwortung des wesentlichen Fragen der Strukturierung des Migrationsprozesses.

In diesem Abschnitt wird ein Überblick zu den Elementen des ReMiP geschaffen. Damit ist gewährleistet, dass bereits zu Beginn der Arbeit ein grundlegendes Verständnis vorhandenen ist. Im Verlauf der Arbeit wird an geeigneter Stelle nochmals auf die einzelnen Bereiche eingegangen.

Es ist zu erwähnen, dass der ReMiP als Referenz-Prozessmodell keine feste Vorgabe für die Durchführung vorschreibt, sondern im Problemkreis der Software-Migration allgemein gültige Lösungsvorschläge macht [Ackermann (2005), S. 24]. In dieser Eigenschaft ist er für konkrete Migrationen konfigurierbar [Ackermann (2005), S. 25]. Daraus folgert, dass die darin angesprochenen Aktivitäten durchaus angepasst, gekürzt und erweitert werden können, wenn die Anwendung es erfordert.

1.2.1 Phasen einer Migration

In Hinsicht auf die zeitliche Dimension werden Software-Entwicklungsprozesse häufig in sequenzielle Phasen eingeteilt. Damit soll gewährleistet sein, dass die Ziele jeder Phase erreicht werden und das Projekt im Gesamten beherrschbar wird [Ackermann (2005), S. 147 / 148]. Auch für Migrationsprojekte ist die Berücksichtigung der zeitlichen Komponente eine wichtige Voraussetzung. Die durchschnittliche Dauer zur Durchführung einer Migration wird von Sneed abhängig von der Größe des Systems zwischen 6 Monaten und zwei Jahren angegeben [Sneed (1999), S. 32].

Der ReMiP integriert in seiner Betrachtung der Migration die Existenz von vier unterscheidbaren Phasen. Diese werden nacheinander durchlaufen und geben beim Abschluss Artefakte, so genannte Meilensteine, an die nächste Phase weiter. Abbildung 1 gibt in Anlehnung an [Ackermann (2005), S. 148] eine Übersicht der Abfolge der Phasen einer Migration. Nachfolgend werden die einzelnen Phasen beschrieben.

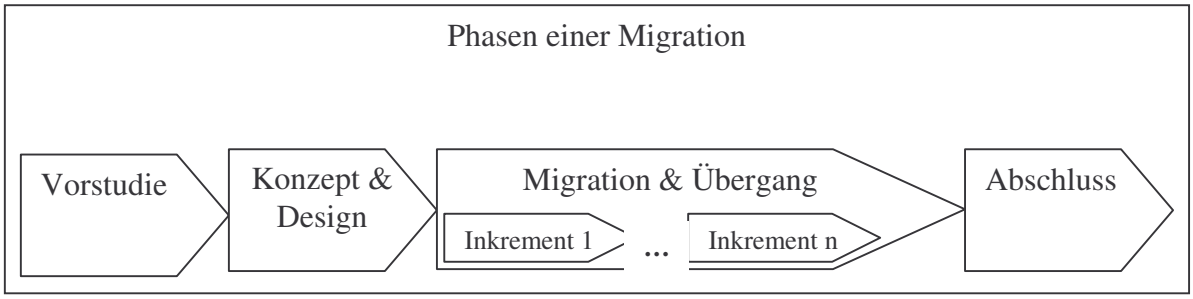

Abbildung 1: Phasen einer Migration nach ReMiP

Vorstudie

Die Phase *Vorstudie* beschäftigt sich mit der Untersuchung des zu migrierenden Legacy-Systems unter wirtschaftlichen, technischen und organisatorischen Gesichtspunkten [Ackermann (2005, S. 149]. Sie umfasst neben der aus dem Projektmanagement bekannten Machbarkeitsanalyse auch die Formulierung von Zielen der Migration und der dabei angewandten Migrationsstrategie.

Die gewonnenen Erkenntnisse werden nach Abschluss der Phase Vorstudie in den Artefakten *Projektziel* und *Vorgehensempfehlung* festgehalten. Sie hat auf Grund der zunächst oberflächlichen Beurteilung des zu migrierenden Systems einen provisorischen Charakter und muss von allen Beteiligten an diesem Projekt angenommen werden. Die Vorstudie umfasst

die Definition des Projektumfangs, der Anforderungen und Einschätzungen zu Kosten, Zeitaufwand und Risiken [Ackermann (2005), S. 152]. Zur Sammlung erster Erfahrungen mit der Migrationsumgebung kann es dort bereits kleine Teilmigrationen geben.

Konzept & Design
Die Phase *Konzept & Design* beschäftigt sich mit der eigentlichen Planung der Migration [Ackermann (2005), S. 149]. Die Richtung der Untersuchung des zu migrierenden Legacy-Systems wird dabei aus den Artefakten der Vorstudie entnommen. Im Mittelpunkt der Untersuchung liegt das Systemverstehen zur Ableitung einer geeigneten Migrationsstrategie. Hierfür ist eine an den projektbedingten Restriktionen und der Migrationsproblematik orientierte, detaillierte technische Prüfung notwendig.

Die Erstellung eines *verbindlichen Masterplans* schließt die Phase *Konzept & Design* ab. Dieser Meilenstein hält in erster Linie die gewählte Migrationsstrategie fest, gibt aber auch eine genaue Beschreibung des geplanten Zielsystems wieder. Zur erfolgreichen Ablieferung des verbindlichen Masterplans gehört auch eine eingehende Prüfung bezüglich der Belastbarkeit der Anforderungen und des Zielsystems sowie der projektbedingten Rahmenbedingungen wie Einhaltung des Zeitplans und Kontrollierbarkeit von Risiken [Ackermann (2005), S. 152].

Migration & Übergang
In der Phase *Migration & Übergang* wird die zuvor im verbindlichen Masterplan definierte Migrationsstrategie angewandt und das Legacy-System in das Zielsystem transformiert. Hierbei kann entweder ein schrittweises Vorgehen gewählt werden, bei dem Komponenten des Legacy-Systems iterativ umgestellt werden, oder die gesamte Migration in einem Schritt abgeschlossen wird. Beide Alternativen sind durch den ReMiP erfasst.

Im Rahmen der Migration sind eventuell Sanierungsmaßnahmen auf Programmen und Daten des Legacy-Systems notwendig. Je nach Zustand seiner Komponenten oder der Erfordernisse des Zielsystems ermöglicht die Sanierung des Legacy-Systems erst die Durchführung der Migration. Für die Umwandlung der Programme, Benutzeroberflächen und Daten des Alt-Systems werden Transformationsregeln formuliert.

In zeitlicher Hinsicht müssen fortlaufende Wartungsarbeiten am Legacy-System berücksichtigt werden. Dies kann dazu führen, dass nach bereits erfolgter Migration eine so genannte Deltamigration erforderlich ist, die Veränderungen am Legacy-System ins Zielsystem überführt [Ackermann (2005), 150].

Nach der Transformation des Legacy-Systems muss überprüft werden, ob die funktionale Äquivalenz des Zielsystems gewährleistet ist [Ackermann (2005), S. 151]. Solche Tests werden als Regressionstests bezeichnet. Die Vorgabe der funktionalen Äquivalenz des Zielsystems hängt eng mit dem Grundprinzip der Migration zusammen, dass die fachliche Funktionalität nicht verändert werden darf.

Zum Ende der Phase *Migration & Übergang* werden sämtliche Bestandteile der Übergangsarchitektur der Migration entfernt. Hierzu gehören z.B. Gateways, die die Interoperabilität zwischen Legacy- und Zielsystem für die Dauer der Migration hergestellt haben [Ackermann (2005), S. 151]. Zudem muss den Anwendern und dem Wartungspersonal die Nutzung des migrierten Zielsystems durch Training näher gebracht werden.

Die Übergabe des migrierten System-Releases ist der Meilenstein, der die Phase *Migration & Übergang* beschließt. Dieses Release umfasst alle Programme, Benutzungsschnittstellen und Daten des Zielsystems [Ackermann (2005), S. 153]. Zur endgültigen Systemabnahme muss dieses noch auf Kriterien wie Stabilität und Performance untersucht werden.

Abschluss
Die *Abschluss*-Phase beschäftigt sich mit der Abwicklung des Migrationsprojekts. Zur Übertragung der Erkenntnisse im Verlauf der Migration wird eine Ergebniskontrolle durchgeführt, die die Kongruenz zwischen den Projektvorgaben und der Zielerreichung untersucht [Ackermann (2005), S. 151]. Ebenso von Bedeutung ist die Archivierung der während der Migration erstellten Dokumente, um die Nachvollziehbarkeit beim Vorgehen der Migration und die Wartbarkeit des Zielsystems zu gewährleisten. Mit dem Abschluss des Migrationsprojekts werden auch die am Projekt beteiligten Mitarbeiter von ihren Aufgaben entbunden und freigesetzt.

Das Artefakt der Phase *Abschluss* ist die Abschlussdokumentation. In ihr werden die oben angesprochenen Punkte festgehalten und in einer abschließenden Bewertung das Projekt für abgeschlossen erklärt.

1.2.2 Kernbereiche einer Migration

Die Einteilung der Migration in Phasen gibt einen strengen sequenziellen Ablauf vor. Es muss immer eine Phase mit einem verabschiedeten Meilenstein abgeschlossen sein, bevor die Migration in die nächste Phase übergehen kann. Die Strukturierung der Migrationsaktivitäten in Kernbereiche sowie Basisbereiche (siehe Abschnitt 1.2.3) verfolgt hingegen ein anderes Ziel. Es geht vielmehr um die logische Zusammenfassung von verwandten Aktivitäten, die im Migrationsverlauf nebenläufig ausgeführt werden. Hierbei werden die Aktivitäten der Kernbereiche in der Regel nicht nur einmal durchlaufen, sondern inkrementell und mit unterschiedlicher Intensität vervollständigt [Ackermann (2005), S. 153 / 154].

Der ReMiP unterscheidet im Migrationskontext sieben Kernbereiche. Diese umfassen typisch migrationsspezifische Aktivitäten auf operativer Ebene. Während die im vorigen Abschnitt angesprochenen Phasen zur Beherrschung des Migrationsprojekts sich auf einem höheren Verantwortungsniveau bewegen, gehen die Kernbereiche auf in den Phasen beinhaltete Tätigkeiten ein. Dabei lassen sich die Kernbereiche oftmals nicht vollständig, sondern nur schwerpunktmäßig bestimmten Phasen zuordnen. Abbildung 2 stellt in Anlehnung an [Ackermann (2005), S. 154] die migrationsspezifischen Kernbereiche einer Migration dar.

Spezifische Kernbereiche einer Migration						
Anforderungs-analyse	Legacy-Analyse / Aufbereitung	Ziel-Design	Strategie-Auswahl	Transfor-mation	Test	Übergabe

Abbildung 2: Spezifische Kernbereiche einer Migration nach ReMiP

Die Abschnitte 2 bis 12zur Beschreibung der Migration der GXL-Website sind strukturell an den Kern- und Basisbereichen einer Migration ausgerichtet. Innerhalb der Unterabschnitte stellt eine weitere Erläuterung der Kernbereiche einen Bezug zu dieser Arbeit her. Zudem wurden teilweise bereits Bereiche in der Beschreibung der Phasen einer Migration explizit angesprochen.

Anforderungsanalyse
Der Kernbereich der Anforderungsanalyse im Rahmen einer Migration soll klären, was das zu entwickelnde Zielsystem leisten soll. Dabei ist eine systematische Vorgehensweise zur Ermittlung der Anforderungen an das Zielsystem notwendig. Der ReMiP stellt hierfür die Aktivitätsgruppen *Problem analysieren*, *Anforderungen ermitteln*, *System definieren* und *Anforderungen verwalten* zur Verfügung [Ackermann (2005), S. 171].

Problem analysieren

Um das bei der Migration vorliegende Problem zu analysieren, ist die Kenntnis der vom Migrationsprojekt betroffenen *Stakeholder* von Interesse. Deren vollständige Identifizierung dient der Erfassung ihrer Bedürfnisse und die Ableitung von Rahmenbedingungen für die Migration.

In Absprache mit den Stakeholdern wird nun das Ziel verfolgt, die Problemstellung genauer zu spezifizieren. Dabei interessieren die Festlegung von Systemgrenzen, die Erfassung von Rahmenbeschränkungen und eine grobe Ausarbeitung der Problemlösung [Ackermann (2005), S. 172]. Das Ergebnis dieser Problembeschreibung wird in der *Vision* festgehalten.

Im Rahmen des Migrationsprojekts ist es wichtig, eine unmissverständliche Kommunikation zwischen den Projektbeteiligten zu gewährleisten. Hierfür wird ein *Glossar* erstellt, das die zentralen Begriffe im Zusammenhang mit der Migration festschreibt und erläutert.

Anforderungen ermitteln

Im Teilbereich *Anforderungen ermitteln* ist es wichtig, alle Anforderungen zu erkennen, zu verstehen und zu formulieren [Ackermann (2005), S. 173]. Da sich Anforderungen im Verlauf eines Projekts häufig ändern, ist damit keine definitive Festlegung der Anforderungen gemeint. Vielmehr ist hierunter zu verstehen, dass alle bereits zu Beginn des Migrationsvorhabens bekannten Anforderungen erfasst werden. Zu erwartende Anforderungsänderungen müssen über ein Anforderungsmanagement verwaltet werden (siehe Abschnitt 2.3).

Zunächst werden die *Stakeholderbedürfnisse* ermittelt. Diese sollen Wünsche und Anforderungen aus der Sicht der Stakeholder möglichst vollständig erfassen. Dabei können die Bedürfnisse der Stakeholder untereinander durchaus divergieren, eine konsistente Abstimmung der Anforderungen ist hier noch nicht das Ziel [Ackermann (2005), S. 173 / 174]. Die im Folgenden hervorgehobenen identifizierten Bedürfnisse der Stakeholder wurden den einzelnen Stakeholdern zugeordnet.

Bei der *Ableitung der Anforderungen* wird auf mehrere Informationsquellen zurückgegriffen. Einen wesentlichen Beitrag dazu liefert die Legacy-Analyse durch die Beschreibung der Funktionalität, die auf das Zielsystem übertragen werden soll. Aber auch die im Rahmen der Problemanalyse gewonnenen Erkenntnisse fließen hier ein [Ackermann (2005), S: 174].

Je nach Migrationsstrategie ist sind die *Anforderungen an temporäre Komponenten* festzuschreiben. Vor allem bei einem inkrementellen Übergang vom Legacy- zum Zielsystem muss genau spezifiziert werden, wie die Elemente des Legacy-Systems im Migrationsverlauf miteinander interagieren sollen.

Nachdem alle Anforderungen erfasst wurden, dient die Spezifierung der Anforderungen der vollständigen, konsistenten und eindeutigen Beschreibung der Zielsystemeigenschaften [Ackermann (2005), S 174]. Dabei gilt es, Konflikte zwischen Anforderungen zu erkennen und zu vermeiden, und die Aufgabe der Migration genau abzustecken. Anforderungen, die nicht mit dem Grundprinzip der Migration der Beibehaltung der fachlichen Funktionalität konform sind, sind in Nachfolgeprojekte auszulagern [Ackermann (2005), S. 174].

System definieren

Durch detailliertere Beschreibung und *Verfeinerung der Anforderungen* wird das Zielsystem definiert. Dabei soll das Detaillierungsniveau so festgelegt werden, dass Designer und Tester davon ausgehend ihre Aufgaben wahrnehmen können.

Anforderungen verwalten

Da sich Anforderungen im Projektverlauf verändern und durch die zunehmende Kenntnis des Gegenstands der Migration neue Anforderungen hinzukommen, ist die *Verwaltung von Anforderungen* ein elementarer Bestandteil der Anforderungsanalyse. Ihre Aufgabe ist die vollständige, korrekte und konsistente Dokumentation der Anforderungen im Projektverlauf [Ackermann (2005), S. 176].

Durch die *Erstellung eines Anforderungsmanagement-Plans* wird eine Konvention verabschiedet, wie Anforderungen zu erfassen sind. Dabei ist auch die Art der Dokumentation der Attribute und der Beziehungen zwischen den Anforderungen von Bedeutung.

Auf Grundlage des Anforderungsmanagement-Plans kann die *Verwaltung der Abhängigkeiten* zwischen Anforderungen durchgeführt werden. So ist können bei Änderung oder Wegfall einer Anforderung die Folgen auf andere leicht nachvollzogen werden.

Legacy-Analyse / -Aufbereitung

Der Kernbereich der Legacy-Analyse stellt das Verstehen des Legacy-Systems durch Analyse der Programme, Datenstrukturen und Schnittstellen in den Vordergrund [Ackermann (2005), S. 177]. Die gewonnenen Erkenntnisse wirken sich direkt auf die Kernbereiche „Ziel-Design“ und „Strategie-Auswahl“ aus.

Design & Verhalten des Legacy-Systems grob rekonstruieren

Die Aktivitätsgruppe zur Rekonstruktion von Design und Verhalten des Legacy-System verfolgt das Ziel, für die nachfolgenden Schritte der Analyse Vorarbeit zu leisten und zu vertiefende Untersuchungsgegenstände zu identifizieren.

Die *Inventarisierung des Legacy-Bestandes* systematisiert das häufig unstrukturiert vorliegende Legacy-System. Bei dieser Aktivität ergeben sich oft auch erste Einblicke in eventuell vorliegende Schwachstellen.

Durch intensiven Austausch mit den Legacy-Experten erfolgt eine abstrakte *Beschreibung des Legacy-Systems und der darin gehaltenen Daten*. Ebenso kann hier auf bestehende Systemdokumentation zurückgegriffen werden, um das Alt-System zu beschreiben.

Die in den vorhergehenden Aktivitäten gesammelten Ergebnisse werden schließlich in einer *Anforderungsspezifikation* festgeschrieben. Der Vorgang der groben Beschreibung des Legacy-Systems wird durch die Überprüfung der Legacy-Analyse-Dokumente abgeschlossen.

Legacy-System global bewerten

Nebenläufig zur groben Beschreibung des Alt-Systems wird seine globale Bewertung durchgeführt. In der Aktivität „Legacy-System global bewerten“ sollen auf Grundlage der Systembewertung wichtige Erkenntnisse und Entscheidungshilfen für die Wahl der Migrationsstrategie gefunden werden. Bis zu diesem Zeitpunkt ist noch unklar, welche Art von Migration dem Legacy-System am besten entspricht. Erst durch die Daten aus diesem Arbeitsschritt kann eine fundierte Entscheidung getroffen werden.

Zur Evaluation von Legacy-Systemen rät Ackermann in Rückgriff auf [Sneed (1999), S. 85 / 86] zur Verwendung einer Portfolio-Analyse, die die Kriterien „Technische Qualität“ und „Betriebswirtschaftliche Bedeutung“ in Verhältnis bringt [Ackermann (2005), S. 184]. Für die Messung der Parameter eines jeden Kriteriums muss eine *Bewertungsstruktur* definiert werden, die einen sinnvollen Ausgleich schafft zwischen Aufwand der Datenerhebung und dem Detaillierungsgrad [Ackermann (2005), S. 185]. Je nachdem kann es in einigen Fällen

ausreichen, eine *Expertenmeinung* einzuholen. In anderen Situationen wiederum kann nur eine formale Bewertung bedeutungsvolle Ergebnisse liefern.

Nach der Sammlung der *Daten* über das Legacy-System werden diese *ausgewertet* und in das Portfolio-Modell abgebildet. Die Portfolio-Analyse kann nur die Bewertung des Legacy-Systems auf Grundlage der Kriterien technischer Qualität und betriebswirtschaftlicher Bedeutung durchführen. Hierbei handelt es sich um die wichtigsten Faktoren zur Bestimmung der Migrationsstrategie. Allerdings gilt es auch, *organisatorische Faktoren* zu berücksichtigen, die sich auf die Migrationsentscheidung auswirken. Das Ergebnis der gesamten Bewertung muss schließlich noch geprüft werden.

Legacy-System detailliert reverse-engineeren

Nach der erfolgten Grobbeschreibung des zu migrierenden Legacy-Systems und seiner Bewertung dient die Aktivitätsgruppe „Legacy-System detailliert reverse-engineeren“ dem Verstehen des Alt-Systems auf einem für die Migration angemessenen Detaillierungsniveau [Ackermann (2005), S. 186]. Wie genau das System analysiert werden muss, hängt wesentlich von der favorisierten Migrationsstrategie ab, die auf Grundlage der Ergebnisse der Bewertung des Legacy-Systems entworfen wurde.

Das Vorgehen ähnelt sehr den Aktivitäten zur Grobbeschreibung des Alt-Systems. Unter Rückgriff der dort gesammelten Daten und Untersuchungshinweise werden problemspezifisch nochmals das *Legacy-Design und die Legacy-Daten untersucht*. Dabei sind der Austausch mit den Legacy-Experten und die Auswertung der bestehenden Legacy-Dokumentation hilfreich.

Nachdem das Legacy-System auf dem gewünschten Detaillierungsniveau beschrieben worden ist, werden die Ergebnisse in der Legacy-Anforderungsspezifikation festgehalten. Zuletzt werden die erstellten Dokumente überprüft.

Legacy-System vorbereiten

Der vom Titel dieses Abschnitts ebenfalls angesprochene Teilbereich der Legacy-Aufbereitung, auch Sanierung genannt, baut auf der Legacy-Analyse auf. Hierbei wird das Legacy-System den Anforderungen an die Migration angepasst. Dies wird notwendig, wenn das System eine hohe Komplexität aufweist oder Qualitätskriterien, z.B. bezüglich des Formats der gehaltenen Daten, nicht erfüllt werden [Ackermann (2005), S. 178].

Im Rahmen der *Sanierung* wird das System für die spätere Transformation vorbereitet. Dabei kann es zu einer Anpassung der Legacy-Daten oder zu einem Refactoring der Legacy-Programme kommen.

Durch die *Bildung von Migrationspaketen* wird das Alt-System für die Migration unterteilt. Dieses Vorgehen ist insbesondere bei einer inkrementellen Migration von Vorteil, da somit einzelne Pakete migriert werden können, während das restliche System in der alten Umgebung bleibt. Die Paketbildung bedeutet auch einen Zuwachs an Wissen über das Legacy-System, was sich in einer Verfeinerung des Legacy-Inventars niederschlägt.

Die Ergebnisse der Aktivitätsgruppe „Legacy-System vorbereiten“ müssen abschließend genau überprüft werden, um für die Migration schwerwiegende Fehlentscheidungen zu vermeiden.

Ziel-Design

Innerhalb des Kernbereichs des Ziel-Designs wird schrittweise die Struktur des Zielsystems entwickelt. Als wesentlicher Orientierungspunkt dient die aus der Legacy-Analyse gewonnene Beschreibung des Legacy-Systems [Ackermann (2005), S. 189]. Dabei sollte auf ein ausgewogenes Verhältnis zwischen Wiederverwendung des Alt-Systems und Weiterentwicklung des Zielsystems geachtet werden.

Zielsystem-Kandidaten definieren

Bei der Definition der Zielsystem-Kandidaten kommt es auf die Auswahl der für die Anforderungen der Migration am meisten geeigneten *Zieltechnologien* und die sie implementierenden *Produkte* an. Technologien wie Produkte werden in einer Liste zusammengefasst.

Ein Zielsystem wird von Ackermann als Kombination verschiedener Technologien und Produkte definiert [Ackermann (2005), S: 194]. Bei der Zusammenstellung ergeben sich mehrere Alternativen. Jedes Zielsystem hat eigene Auswirkungen auf die Wahl der Migrationsstrategie.

Zur Überprüfung der Eignung jedes einzelnen Zielsystem-Kandidaten wird auf Grundlage zielführender Kriterien eine Bewertung durchgeführt. Auf dieser Grundlage kann die Auswahl eines geeigneten Zielsystems vorgenommen werden. Auch die Definition des Kandidaten wird abschließend eingehend geprüft.

Zielsystem-Architektur verfeinern

Die Zielsystem-Architektur dient zur Schaffung eines Übergangs von den systembedingten Eigenheiten des Zielsystem-Kandidaten zu seinem konkreten Entwurf, was sich im Design von Programmen, Datenbanken und Benutzungsschnittstellen niederschlägt [Ackermann (2005), S. 195]. Die Zielsystem-Architektur wird dabei mit Fortschreiten der Migration immer wieder auf die Bedürfnisse des nächsten Migrationsschritts verfeinert.

Bei der *Analyse der Zielsystem-Architektur* wird der technische Rahmen des Zielsystems beschrieben [Ackermann (2005), S. 196]. Dabei ist die grundlegende Beschreibung der Soft- und Hardware-Konfiguration von Belang. Bei verteilten Systemen ist zugleich die *physische Verteilung* zu untersuchen. Aus diesen Aktivitäten ergibt sich ein eher abstraktes Bild des Zielsystems, auf dem die Design-Aktivitäten aufbauen können. Die *Zielsystem-Architektur* wird zum Abschluss *geprüft*.

Übergangs-Architektur entwerfen

Bei der inkrementellen Durchführung eines Migrationsprojekts werden im Migrationsverlauf einzelne Komponenten des Legacy-Systems in das Zielsystem überführt. Zur Gewährleistung der Kommunikation zwischen den noch im Alt-System befindlichen und bereits migrierten Komponenten muss eine *Übergangs-Architektur festgelegt* werden [Ackermann (2005), S. 196]. Wird stattdessen eine Migrationsstrategie verfolgt, bei der das Alt-System in nur einem Iterationsschritt in das Zielsystem überführt wird, entfällt diese Aktivität. Abschließend wird die *Übergangs-Architektur* auf seine Eignung *geprüft*.

Programme entwerfen

Die aus der Legacy-Analyse gewonnene Funktionalität des Alt-Systems wird in der Aktivitätsgruppe „Programme entwerfen“ in das Zielsystem überführt. Durch die *Erstellung des Anwendungs-Designs* werden die neuen Programme beschrieben. An dieser Stelle wird in der Regel nicht programmiert, sondern lediglich die Funktionalität beschrieben, die das Zielsystem nach der Transformation aufweisen soll. So muss das Design so detailliert sein, dass auf deren Grundlage die Transformationsregeln (siehe weiter unten) abgeleitet werden können [Ackermann (2005), S. 197]. Die *Prüfung des Anwendungs-Designs* schließt diese Aktivitätsgruppe ab.

Benutzungsschnittstellen entwerfen

Innerhalb des Entwurfs der Benutzungsschnittstellen wird die Interaktion zwischen dem Zielsystem und den Benutzern beschrieben. Neben der Definition der Interaktionselemente wie Fenster, Menüs und Dialoge gehört auch die Darstellung der Navigationspfade zum

Benutzungsschnittstellen-Design [Ackermann (2005), S. 198]. Um die Akzeptanz der Benutzer zu erhöhen, sollte die Benutzeroberfläche zu einem frühen Zeitpunkt der Migration in einem *Prototyp* demonstriert werden. Das Benutzungsschnittstellen-Design wird schließlich noch *geprüft*.

Datenbanken entwerfen

Die Aktivitätsgruppe „Datenbanken entwerfen" beschäftigt sicht mit der Erstellung der Datenstrukturen des Zielsystems. In diesem Schritt wird im *Datenbank-Design* ein Datenmodell entworfen, das zum Abschluss nochmals *überprüft* wird.

Transformationen entwerfen

Der Entwurf der Transformationen ist für die Migration eine zentrale Aktivität [Ackermann (2005), S 199]. An dieser Stelle werden die Regeln definiert, nach denen im Kernbereich „Transformation" zur Abbildung des Legacy-Systems ins Zielsystem vorgegangen wird.

Durch die *Identifizierung des „Legacy-Ziel"-Mappings* werden die Elemente des Legacy-Systems erkannt, die im Zielsystem wieder verwendet werden [Ackermann (2005), S. 200]. Dieses Mapping wird durch die *Definition der Transformationsregeln* formalisiert. Auf Grundlage dieser Regeln soll die Automatisierung der Transformation ermöglicht werden.

Die *Definition der Migrationsumgebungs-Konfiguration* beschreibt die für die Transformation notwendigen Soft- und Hardware-Ressourcen [Ackermann (2005), S. 200]. Auch zu diesem vorangeschrittenen Zeitpunkt der Migration kann es zur *Identifizierung von Aufbereitungsbedarf* kommen. Dieser Aufbereitungsbedarf umfasst sowohl Maßnahmen zur Vereinfachung der Transformation sowie der Erhöhung der Wiederverwendbarkeit des Legacy-Systems [Ackermann (2005), S. 200]. Die Transformationsregeln werden zum Abschluss dieser Aktivitätsgruppe *geprüft*.

Strategie-Auswahl

Der Begriff der Strategie-Auswahl für eine Migration beschäftigt sich mit den Strategiebereichen der Transformation, der Umstellung und der Übergabe [Ackermann (2005), S. 201]. Die Transformationsstrategie beschreibt, wie transformiert wird, also ob eine *Neuentwicklung*, *Konversion* oder *Kapselung* des Zielsystems angewandt wird. Bei der Umstellungsstrategie geht es um die Festlegung der Anzahl der inkrementellen Migrationsschritte. Innerhalb der Übergabestrategie wird die Art der Inbetriebnahme des migrierten Systems thematisiert. Hierbei ist ein Parallelbetrieb zwischen dem Alt- und dem migrierten System, aber auch ein vollständige Übergang von Legacy zu Zielsystem möglich.

Migrationsstrategien für Zielsystem-Kandidat erarbeiten

In der Aktivität „Migrationsstrategien für Zielsystem-Kandidat erarbeiten" werden zunächst die *Strategiealternativen* erfasst. Diese Alternativen umfassen die Definition der Transformationsstrategien, sowie auch der Umstellungs- und Übergabestrategien. In die Entscheidungsfindung gehen Informationen aus der Analyse des Legacy-Systems mit ein. Je nach Alternative kann sich ein *Aufbereitungsbedarf* des Legacy-Systems ergeben, der zu *ermitteln* ist. Eine solche Aufbereitung ist nur dann zu empfehlen, wenn damit eine erhebliche Vereinfachung der Transformation einhergeht [Ackermann (2005), S. 209].

Unter Rückgriff auf die Erkenntnisse bei der Paketbildung des Legacy-Systems in der Legacy-Aufbereitung werden *alternative Paketbildungsstrategien gesammelt*. Kann das Legacy-System sinnvoll in Pakete aufgeteilt werden, so sind potenzielle Bearbeitungsreihenfolgen zu definieren [Ackermann (2005), S. 210]. Eine Prüfung der Migrationsstrategien schließt diese Aktivitätsgruppe ab.

Migrationsstrategien für Zielsystem-Kandidat bewerten

Die in der vorhergehenden Aktivitätsgruppe durchgeführte Definition von Strategiealternativen wird in der Aktivitätsgruppe „Migrationsstrategien für Zielsystem-Kandidat bewerten“ näher untersucht. Bei der *Validierung der Strategien* muss geklärt werden, welche Strategien den Migrationsanforderungen genügen und ob sie technisch und organisatorisch durchführbar sind [Ackermann (2005), S. 210]. Die *Selektion der Migrationsstrategien* wählt aus allen erarbeiteten Alternativen die meist versprechenden aus. Die gesamte Bewertung der Migrationsstrategien wird abschließend *geprüft*.

Migrationsstrategie auswählen

In der Aktivitätsgruppe „Migrationsstrategie auswählen“ gilt es, die aus der Menge der Alternativen ausgewählten Migrationsstrategien einer tiefer gehenden Untersuchung zu unterziehen. Dabei sind vor allem wirtschaftliche Kriterien zu beachten [Ackermann (2005), S. 211].

Bei der *ökonomischen Bewertung der Migrationsstrategien* werden die mit den Alternativen verbundenen Kosten und Risiken eingehend untersucht. Eine genaue Bestimmung dieser Kriterien kann oft erst durch die *Verfeinerung der Migrationsstrategien* erreicht werden [Ackermann (2005), S. 212].

Nachdem die Kosten möglichst genau quantifiziert wurden, gilt es sie mit dem Nutzen der Migrationsstrategie bei einer *Kosten / Nutzen-Analyse* miteinander in Verhältnis zu setzen. Ebenso von Bedeutung ist die nachgelagerte Durchführung einer *Kosten / Risiken-Analyse*, um für jede Alternative das finanzielle Risiko bestimmen zu können.

Ausgehend von den Ergebnissen dieser detaillierten Betrachtung der Strategiealternativen erfolgt die *Selektion der globalen Migrationsstrategie*. Hierzu gehören auch Entscheidungen, die sich mit dem Outsourcing bestimmter Aktivitäten des Migrationsprojekts an externe Dienstleister [Ackermann (2005), S. 212]. Das Ergebnis der Selektion wird in einem Migrationsstrategieplan dokumentiert. Die Strategieauswahl wird abschließend *geprüft*.

Transformation

In den eingangs besprochenen Kernbereichen einer Software-Migration wurden vor allem Vorbereitungstätigkeiten für die Überführung des Legacy-Systems in das Zielsystem durchgeführt. Das Alt-System wurde ausführlich beschrieben und das Zielsystem entworfen. Der Kernbereich der Transformation beschäftigt sich nun mit der konkreten Überführung der Legacy-Pakete in das Zielsystem [Ackermann (2005), S. 214].

Migrationspaket isolieren

Für jeden Iterationsschritt der Transformation wird das *Migrationspaket in eine isolierte Umstellungsumgebung übernommen*. Innerhalb dieses Migrationspakets werden je nach Wahl der Migrationsstrategie die *Wartungsarbeiten reduziert* oder auch ganz eingestellt [Ackermann (2005), S. 219]. Die Übernahme des Migrationspakets wird schließlich noch überprüft.

Migrationspaket transformieren

Im Rahmen der Aktivitätsgruppe „Migrationspaket transformieren“ wird das isolierte Migrationspaket in das Zielsystem überführt. Je nach Vorgabe der Migrationsstrategie erfolgt dies im Allgemeinen automatisiert durch *Neuentwicklung*, *Konversion* oder *Kapselung* [Ackermann(2005), S. 219]. Ackermann hebt unter Verweis auf [Sneed (1999), S. 189] hervor, dass die automatisierte Transformation nicht vollständig ist und manuelle Nachbearbeitung des Programmcodes bedingt [Ackermann (2005), S. 219 / 220]. Die Transformation wird durch ihre Prüfung abgeschlossen.

Deltamigration durchführen

Unter der Annahme, dass im Verlauf der Transformation geringfügige Wartungsarbeiten am Legacy-Paket durchgeführt werden müssen, schließt sich der Transformation des Migrationspakets eine *Deltamigration* an. So müssen nach erfolgter Vormigration die veränderten Komponenten des Migrationspakets erneut durch Konversion oder Kapselung transformiert werden [Ackermann (2005), S. 220]. Damit es nicht zu einer sich immer fortsetzenden Folge von Deltamigrationen kommt, müssen die *Wartungsarbeiten am Migrationspaket vollständig gesperrt* werden. Wie in der Aktivitätsgruppe „Migrationspaket transformieren" wird die Deltamigration geprüft.

Komponenten implementieren

Manche Migrationspakete können qualitative Mängel aufweisen, die bewirken, dass sie sich nicht für eine Konversion oder Kapselung eignen. Sie können nicht migriert werden. Dort wird eine *Neuimplementierung* notwendig. Obwohl es sich bei der Neuimplementierung nicht um eine Migrationsaktivität handelt, hat Ackermann sie zur Vollständigkeit in ihr Referenz-Prozessmodell aufgenommen [Ackermann (2005), S. 221]. Der Quelltext der neu implementierten Komponenten des Zielsystems wird abschließend geprüft.

Test

Im Aktivitätsbereich „Test" erfolgt die Überprüfung, ob das Zielsystem dem Grundprinzip der Migration entspricht und funktionale Äquivalenz aufweist [Ackermann (2005), S. 222]. Im Kontext einer Migration spricht man bei diesen Tests von Regressionstests. Diese Tests beschränken sich auf die Untersuchung der Änderungen eines Systems [Sneed (1999), S. 228]. Bei einer Migration entspricht dies der Überprüfung von Abweichungen im Verhalten zwischen Legacy- und Zielsystem. Da die Durchführung von Tests oft mit einem erheblichen Aufwand verbunden ist, empfiehlt Ackermann die Ausnutzung von Automatisierungspotenzialen [Ackermann (2005), S. 222].

Globale Teststrategie definieren

Bei der *Definition der Teststrategie* umfasst die Dokumentation der durchzuführenden Tests, deren Ziele und Beteiligten sowie die Nutzung der Testdaten [Ackermann (2005), S. 226]. Zudem wird die technische Testumgebung spezifiziert. Bei der *Prüfung* der globalen Teststrategie wird die angemessene und vollständige Erfassung der Tests untersucht.

Test für Migrationspaket planen und vorbereiten

Die in der globalen Teststrategie definierten Tests werden in der Aktivitätsgruppe „Test für Migrationspaket planen und vorbereiten" konkretisiert. Hierfür wird die *Teststrategie detailliert* und die *Ausprägungen des Tests definiert*. Eventuell kann der *Einbezug bereits existierender Testdaten* aus dem Legacy-System eine Arbeitserleichterung bedeuten. Als letzte Vorbereitung für die Durchführung der Tests werden diese *strukturiert* und *geprüft*.

Test für Migrationspaket durchführen

Die Durchführung des Tests für das Migrationspaket implementiert den Testrahmen für die Programme. Zunächst wird der Test beim Legacy-System durchgeführt und protokolliert. Schließlich wird der Testrahmen für den Bereich des Zielsystems konvertiert, um auch dort den Test durchführen zu können. Die Ergebnisse des Tests beim Zielsystem werden ebenfalls protokolliert. Zur Gewährleistung der korrekten Ausführung der Tests ist eine Prüfung erforderlich [Ackermann (2005), S. 229].

Testergebnisse auswerten

Bei der Auswertung der Testergebnisse wird festgestellt, ob sich das Zielsystem funktional äquivalent zum Legacy-System verhält. Hierfür werden die *Testbestände analysiert* und in

einem Fehlerbericht dokumentiert. Die *Prüfung der Testauswertung* schließt diese Aktivitätsgruppe ab.

Übergabe
Im Kernbereich der „Übergabe“ wird die Durchführung der Auslieferung des migrierten Zielsystems oder von Teilen beschrieben. Hier wird die im Kernbereich „Strategie-Auswahl“ gewählte Übergabestrategie angewandt. Dabei unterscheidet Ackermann die inkrementelle Übergabe des Zielsystems durch Parallelbetrieb und die vollständige Ablösung des Legacy-System zu einem bestimmten Termin [Ackermann (2005), S. 230].

Zielumgebung einrichten
Die *Installation der Zielumgebungskomponenten* leistet eine wichtige Vorbereitungsmaßnahme für die endgültige Installation des Zielsystems. Dabei wird die Soft- und Hardware installiert, auf die das Zielsystem aufbaut. Dabei handelt es sich um Komponenten wie Betriebssysteme, Datenbanken oder Middleware [Ackermann (2005), S. 235]. Die *Zielumgebung wird getestet* und auf korrekte Installation für die Interaktion mit dem Zielsystem *geprüft*.

Übergabe planen
Bei der *Erstellung eines Übergabeplans* wird festgelegt, zu welchem Zeitpunkt das Zielsystem übergeben wird und das Legacy-System ablöst. Zur Gewährleistung der *Akzeptanz* des neuen Zielsystems werden Tests vorgesehen. Wenn ein Migrationspaket den Kernbereich „Test“ erfolgreich verlässt, wird daraus ein *Übergabepaket definiert*, das mit *Installationsanleitungen* dokumentiert wird. Die Planung der Übergabe wird schließlich noch *geprüft*.

Supportmaterial entwickeln
Zu jeder Software gehört auch die Erstellung von Dokumentation. Damit das Zielsystem von den Anwendern richtig bedient werden kann, ist es erforderlich, die *Endandwender-Dokumentation zu erstellen*. Neben dieser umfangreichen Beschreibung des Zielsystems ist auch die Vorbereitung von Trainingsmaterial für die Durchführung der Übergabe unerlässlich. Die in dieser Aktivitätsgruppe entwickelten Dokumente werden auf Eignung *geprüft*.

Übergabe durchführen
Bei der konkreten Durchführung der Übergabe werden die *Übergabepakete und* eventuell vorhandenen *temporären Komponenten installiert*. Nachdem das Paket in das System integriert wurde, können die *Akzeptanztests* bei den Anwendern *durchgeführt und ausgewertet* werden. Um die korrekte Bedienung des Systems zu gewährleisten, werden parallel *Trainings* durchgeführt. Der Erfolg der Übergabe wird abschließend *geprüft*.

Legacy-System ablösen
Mit dem Abschluss der Migration werden die *nicht vom Zielsystem genutzten Komponenten endgültig abgelöst*. Ebenso werden die nur während der Migration verwendeten *Gateways entfernt*, da sie keine Funktion mehr erfüllen. Die nicht mehr genutzten *Komponenten des Legacy-Systems werden archiviert*, was vor allem in Bezug auf die nur im Legacy-System vorhandenen Daten in Hinsicht auf gesetzliche Aufbewahrungsfristen von Bedeutung sein kann [Ackermann (2005), S. 239]. Die *Prüfung der Ablösung* schließt die Migration ab.

1.2.3 Unterstützende Basisbereiche

Neben den migrationsspezifischen Aktivitäten, die in den Kernbereichen zusammengefasst sind, existieren im Kontext einer Migration so genannte Basisaktivitäten, Diese Aktivitäten müssen zwar auch durchgeführt werden, lassen sich aber nicht auf Migrationsprojekte alleine

beschränken. Der ReMiP kennt für die Durchführung von Migrationen vier Basisbereiche, die in Abbildung 3 in Anlehnung an [Ackermann (2005), S. 156] dargestellt sind.

Unterstützende Basisbereiche			
Konfigurations- / Änderungs-management	Projekt-management	Mitarbeiter-qualifizierung	Migrations-umgebung

Abbildung 3: Unterstützende Basisbereiche nach ReMiP

Konfigurations- / Änderungsmanagement
Das Konfigurationsmanagement befasst sich mit der Erfassung und Aktualisierung der im Verlauf der Migration erstellten Artefakte, verwaltet deren Versionierung und ermöglicht die Nachvollziehbarkeit der Änderungen [Ackermann (2005), S. 156]. Ebenso wie das Konfigurationsmanagement werden durch das Änderungsmanagement Veränderungen im Verlauf der Migration erfasst. Allerdings sind hier nicht die Artefakte, sondern die Änderungswünsche der Untersuchungsgegenstand. Die Ermöglichung der Nachvollziehbarkeit der Änderungen im Legacy-System und in den Anforderungen für das Zielsystem ist wichtig für die Anpassung der Transformationen [Ackermann (2005), S. 157].

Projektmanagement
Zur Bündelung und zweckmäßigen Verwendung der im Projekt eingebundenen Ressourcen ist das Projektmanagement unerlässlich. Eine wichtige Aufgabe lässt sich u.a. aus der Überprüfung der Einhaltung der Vorgaben zur phasenorientierten Durchführung der Migration ableiten. Außerdem sind Abstimmung mit den am Projekt Beteiligten und die Kommunikation des Projektfortschritts Aufgaben des Projektmanagements.

Mitarbeiter-Qualifizierung
Bei einer Migration kommen verschiedene Gruppen von Mitarbeitern zum Einsatz. Ackermann teilt sie in Wartungs- und Migrationsteam ein [Ackermann (2005), S. 246]. Das Wartungsteam ist mit der Aufrechterhaltung der Funktionalität des Legacy-Systems betraut. Das Migrationsteam hingegen treibt die Entwicklung des Zielsystems voran und gewährleistet die Übertragbarkeit des Legacy-Systems in das Zielsystem. Für beide Gruppen werden unterschiedliche Qualifizierungsmaßnahmen benötigt, die ihrem Einsatzgebiet angemessen sind.

Migrationsumgebung
Die Migrationsumgebung deckt alle Bereiche der technischen Unterstützung der Migration ab. Hierzu gehört neben der Werkzeugunterstützung auch die Einführung eines Prozessmodells zur Durchführung der Migration [Ackermann (2005), S. 157]. Ziel des Basisbereichs *Migrationsumgebung* ist eine effektive und effiziente Unterstützung des Migrationsprozesses durch eine geeignete Infrastruktur.

1.3 Beschreibung der Umgebung von Websites

Websites haben sich in den letzten Jahren zu einem wichtigen Einsatzgebiet von Software entwickelt. Obwohl eine schlichte Textseite im Internet nicht per se als Software zu verstehen ist, sind heutige Websites häufig so funktional, dass die Klassifizierung als Software gerechtfertigt ist. Innerhalb des Begriffes „Software“ besitzen Websites eine Sonderstellung. Sie umfassen häufig ebenfalls software-spezifische Elemente wie Programme (implementiert z.B. in Skriptsprachen wie PHP oder ASP), Benutzerschnittstellen (z.B. zur Eingabe von

Login-Informationen oder einer Suchmaske) und Datenmodelle (z.B. zur Speicherung von Nutzerdaten in einer Datenbank). Dennoch verfügen sie bezüglich der genutzten Technologien über Eigenheiten im Vergleich zu „Offline“-Programmen.

1.3.1 Internet-Technologien

Die grundlegende Technologie für den Einsatz von Websites leiten sich von dem *Internet* ab. Es wurde bereits seit den sechziger Jahren entwickelt und ist heute ein weltumspannendes Netzwerk von technologisch heterogenen Computer-Netzen. Diese heterogenen Subnetze kommunizieren miteinander, indem sie auf die plattformunabhängigen Protokolle TCP / IP (Transmission Control Protocol bzw. Internet Protocol) zurückgreifen. Diese Protokolle stellen die Adressierung und den zuverlässigen Transport von Datenströmen sicher. Auf dieser Protokoll-Familie bauen diverse Dienstprotokolle auf, die Daten an Internet-Anwendungen übergeben. Dazu gehören z.B. SMTP (Simple Mail Transfer Protocol, E-Mail), FTP (File Transfer Protocol, Dateiübertragung) und *HTTP* (Hypertext Transfer Protocol). HTTP ist für die Übertragung von Webseiten das relevante Protokoll.

Webseiten sind in der Dokumentenbeschreibungssprache im Allgemeinen in *HTML* (Hyptertext Markup Language) geschrieben, die auf HTTP zurückgreifen kann. Sie wurde als Anwendung aus der SGML (Standard Generalized Markup Language) gebildet und kann in Verbindung mit HTTP Verweise zu anderen Dokumenten herstellen, die nicht auf demselben Rechner gespeichert sind. Dies sind die so genannten *Hyperlinks*. Unter diesem Mechanismus der Verbindung von Dokumenten über Hyperlinks unter Nutzung des HTTP-Dienstes des Internets versteht man die klassische Anwendung des *World Wide Web* [Wöhr (2004), S. 3].

HTML beschreibt die Struktur eines Dokuments, indem sie Elemente wie Absätze, Tabellen oder Überschriften durch *Markup* kennzeichnet. Dies geschieht in diesem Fall durch die Zuweisung von so genannten *Tags*, die das zu bezeichnende Element umschließen. Bei Absätzen sind dies * *, bei Tabellen *<table></table>* und bei Überschriften bspw. *<h1></h1>*. Diese Angaben sind überwiegend allgemein und geben keine Vorgaben für die konkrete Darstellung des Textes [Wöhr (2004), S. 11]. Dennoch gibt es Tags, mit denen man in HTML Anweisungen über die Formatierung der Seite machen kann. Dies sind z.B. Angaben über die Größe eines Elements *(<font-size>)*, die Schriftart *(<font face>)* und sonstige typographische Informationen.

Dennoch ist eine Trennung der Dokumentenstruktur von der Formatierung wünschenswert. So müssten Textelemente, die gleich formatiert sein sollen, wie z.B. Hervorhebungen oder speziell formatierte Programm-Code-Ausschnitte bei einer Änderung der Formatierung an jeder Stelle im Dokument geändert werden. Dies beeinflusst die Wartbarkeit negativ. Ebenso kann die Verwendung Browser-proprietärer HTML-Tags, also von Tags, die nur ein bestimmter Browser oder eine Klasse von Browsern interpretieren kann, zum Verlust der Portabilität des HTML-Dokuments führen [Wöhr (2004), S. 86]. Um HTML-Seiten bei ausschließlicher Verwendung struktureller Tags zu formatieren, können *Stylesheets* verwendet werden.

Stylesheets können auf verschiedene Arten implementiert werden. Dies erfolgt durch die Einbindung einer Style-Definition im HTML-Quelltext. Die größte Wiederverwendbarkeit wird dabei erreicht, wenn Stylesheets in externe Dateien ausgelagert werden und auf sie lediglich mit einem Hyperlink verwiesen wird. Hierbei handelt es sich um *Cascading Stylesheets*-Dateien mit der Dateiendung *.css*. Abbildung 4 zeigt eine exemplarische Anwendung eines externen Stylesheets. Es gibt neben der dort dargestellten Form noch andere Möglichkeiten, Cascading Stylesheets in Webseiten einzubinden.

```
<html>
<head>
        <link rel="STYLESHEET" href="/stylesheet.css" type="text/css">
</head>
</html>
```

Abbildung 4: Verweis auf ein externes Stylesheet

In der entsprechenden Datei stylesheet.css lassen sich die Layout-Informationen zu diesem Dokument finden. Hierbei handelt es sich um Festlegungen für allgemeine HTML-Elemente, z.B. wie eine Überschrift oder ein Absatztext anzuzeigen ist, aber auch um neu zugewiesene Formatierungsklassen.

Die besondere Stärke von Stylesheets liegt darin, dass sie dem Browser nach dem Einlesen der Stylesheet-Informationen die notwendigen Layout-Informationen übergeben. Zudem können unter Anwendung von Vorrangsregeln Stylesheets miteinander *kaskadiert* werden. Das heißt, es können sich mehrere dem Browser übergebene Stylesheets überlagern. Bei Überschneidungen von Layout-Festlegungen wird das im vorrangigen Stylesheet beschriebene Layout angewandt. Im untergeordneten Stylesheet vorhandene Layout-Definitionen, die nicht im vorrangigen Stylesheet festgelegt sind, bleiben weiterhin gültig. Nach dieser Systematik werden auch im Zielsystem Plone Festlegungen für die Anzeige der Webseiten gemacht.

Standards

Welche Tags in einem HTML-Dokument verwendet werden dürfen, wird in *Standards* festgelegt. In einem vollständigen HTML-Dokument wird deshalb auf die verwendete HTML-Version verwiesen, damit der Browser die Elemente korrekt interpretieren kann. Ein Beispiel, wie es auch auf den Webseiten der zu migrierenden Website verwendet wird, ist in Abbildung 5 zu sehen.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
```

Abbildung 5: Angabe zur verwendeten HTML-Version

Aus dieser Versionsangabe kann entnommen werden, dass die Webseiten der zu migrierenden Website unter Anwendung des HTML-Standards 4.0 erstellt wurden. Diese HTML-Standard-Version brachte im Vergleich zu seinen Vorgängern vor allem die Unterstützung von Cascading Stylesheets mit sich [Meinel / Sack (2004), S. 825]. Seit 2000 existiert neben HTML 4.0 auch der *XHTML*-Standard. Hierbei handelt es sich um eine HTML-Version, die auf der *Extensible Markup Language* (XML) basiert. Dieser Standard wurde vor allem aus Gründen der Kompatibilität und Integrierbarkeit mit anderen auf XML-basierenden Standards entwickelt [Meinel / Sack (2004), S. 878]. XHMTL stellt somit, obwohl es sich bezüglich der Menge der zulässigen Tags und der Funktionalität nicht von HTML 4.0 unterscheidet, eine flexiblere Dokumentenbeschreibungssprache dar. Ebenso ist die Zielumgebung Plone auf die Verwendung von XHTML ausgerichtet [McKay / da Silva (2006), S. 49].

Bei der Codierung des XHTML-Quelltextes müssen Konventionen eingehalten werden, damit das Dokument dem XML-Standard entspricht. Hierzu gehören Kriterien wie die Einfassung der Attribute im Tag in Anführungszeichen und das obligatorische Schließen geöffneter Tags. Wenn XML-Dokumente (und damit auch XHTML-Dokumente) den syntaktischen Kriterien entsprechen, spricht man von *wohlgeformten* XML-Dokumenten [Meinel / Sack (2004), S. 946]. Für diese Eigenschaft wird lediglich die Einhaltung der allgemeinen Regeln von XML-Dokumenten überprüft. Es ist aber auch möglich, für ein XML-Dokument auf eine genaue Beschreibung der zulässigen Elemente (z.B. das Kleinschreiben von Tags) und deren Beziehungen zueinander zu verweisen, eine so genannte *Document Type Definition* (DTD) [Wöhr (2004), S. 83]. Wenn ein XML-Dokument auch die dort in einem externen Dokument

festgelegten Regeln erfüllt, spricht man von einem validen oder *gültigen* Dokument [Meinel / Sack (2004), S. 947]. So ist ein XHTML-Dokument erst dann gültig, wenn es den Regeln in der zugewiesenen DTD entspricht.

Web-Programmierung
Neben den beschriebenen Elementen des World Wide Webs, die sich mit der Strukturierung und der Präsentation von Webseiten beschäftigen, ist es auch möglich, Funktionalität in Form von *Web-Programmierung* zu integrieren. Dies erfolgt durch server- oder clientseitige Programmierung, abhängig von dem Ort, an dem die Programme ausgeführt werden. Anwendungen für das Web werden entweder zur Implementierung von Interaktivität mit dem Anwender verwendet oder auch zur Verwaltung von Websites durch dynamische Generierung von Webseiten. In diesen Bereich fällt die Verwendung von *Templates*. Dabei handelt es sich in diesem Kontext um Vorlagen für Webseiten, die Platzhalter für variable Daten enthalten, so genannte Template-Variablen. Sie werden unter Nutzung einer serverseitigen Programmiersprache für gewöhnlich aus einer Datenbank ausgelesen oder auf einer anderen Weise dem Template übergeben.

In Websites eingebundene Programme werden häufig nicht nur von einem einzelnen Anwender genutzt. Eine *Web-Applikation* ist als Anwendung definiert, die von mehreren Benutzern gleichzeitig unter Nutzung eines Browsers bedient werden kann. Es spielen Kriterien wie Benutzerauthentifizierung und Autorisierungsregeln, Nebenläufigkeit der Anwendungen und Skalierbarkeit der gesamten Web-Applikation eine Rolle [Wöhr (2004), S. 24 / 25]. Elemente einer Web-Applikation umfassen insbesondere die Verwaltung der HTML-Dokumente, die Nutzung von client- und serverseitiger Programmierung und die Speicherung von Daten in einer Datenbank [Wöhr (2004), S. 25].

In der Diskussion von Internet-Technologien werden häufig Begriffe wie „Website“ und „Webseite“ überlappend gebraucht. In dieser Arbeit sollen die Begriffe folgendermaßen verstanden werden:

> Eine *Webseite* beschreibt ein einzelnes, in HTML verfasstes Dokument, das über die Nutzung des World Wide Web abrufbar ist. Dabei kann dieses Dokument statisch gespeichert vorliegen oder aber bei Anfrage dynamisch generiert werden. Eine *Website* beschreibt eine zusammengehörende Menge von Webseiten, die häufig durch einen gemeinsamen Adressbereich gekennzeichnet ist.

1.3.2 Stakeholder von Websites

Wie bei jedem Software-Projekt ist auch bei der Erstellung von Websites der Identifikation der am Einsatz und an der Erstellung beteiligten Personen, der *Stakeholder*, ein großes Gewicht einzuräumen. Die angesprochenen Gruppen für Website-Projekte unterscheiden sich nicht wesentlich zu sonstigen Software-Projekten. Dennoch besitzen bestimmte Rollen, die die Stakeholder einnehmen, eine besondere Gewichtung. Zu nennen sind hierbei vor allem Auftraggeber, Anwender, Designer, Redakteure, Programmierer und Domänenexperten [Kappel et al. (2004), S. 31].

Der *Auftraggeber* übernimmt für das Projekt der Website die Verantwortung. Er gibt eher grobe Vorgaben zur Erfüllung des Zwecks der Website und steckt zeitliche und finanzielle Rahmenbedingungen.

Die Rolle des *Anwenders*, wie sie aus der klassischen Software-Entwicklung bekannt ist, existiert auch im Zusammenhang mit Websites. Hierbei fungiert er verstärkt als Website-Betrachter, der die Inhalte der Website konsumiert. Dabei achtet er besonders auf die Darstellung der auf der Website vorhandenen Informationen und die Navigation zu den Inhalten der Website.

Neben dem Anwender findet sich auch die Rolle des *Designers* aus der Software-Entwicklung im Umfeld von Websites wieder. Der Designer ist für die nachvollziehbare Strukturierung der Inhalte der Webseite zuständig, die unter dem Begriff Informationsarchitektur zusammengefasst ist. Zudem muss er durch das Interaktions-Design dem Betrachter einer Website eine leicht verständliche Navigation zur Verfügung stellen. Durch das große Alternativangebot anderer Websites im Internet und die oft nicht vorhandene Kenntnis der Bedienung beim Anwender, muss der Designer ein besonderes Augenmerk auf ansprechende Gestaltung und Selbsterklärbarkeit der Website richten [Kappel et al. (2004), S. 14].

Der *Redakteur* erstellt Inhalte einer Website, wie z.B. Textartikel, ohne dass er sie zwingend selber einstellt. Es handelt sich bei ihm um den direkten oder indirekten Anwender einer Website im Sinne der Aktualisierung von Inhalten. Diejenige Person, die schließlich die vom Redakteur erstellten Inhalte auf der Website veröffentlicht, wird als *Veröffentlicher* oder Publisher bezeichnet. Er übernimmt für diesen konkreten veröffentlichten Teil der Website die Verantwortung.

Der *Programmierer* übernimmt vor allem die Bereitstellung einer Infrastruktur für die Website und implementiert die Funktionalität durch Programmierung in der verwendeten Programmiersprache. Dabei arbeitet er eng mit dem *Domänenexperten* zusammen. Der Domänenexperte kennt sich mit den veröffentlichten Inhalten aus, ohne dass er über technische Kenntnisse wie der Programmierer verfügen muss. Er kann dem Programmierer bezüglich der bestehenden Abhängigkeiten zwischen den auf der Website zu veröffentlichenden Inhalten Hinweise geben.

1.4 Das Content Management-Konzept

Websites sind viel mehr als eine Ansammlung einfacher HTML-Dokumente, die willkürlich miteinander mit Hyperlinks verbunden sind. Stattdessen gibt es, ähnlich wie bei jeder Software, Bedürfnisse wie die leichte Verwaltung der gespeicherten Daten und die Implementierung von Autorisierungsmechanismen. Diese und weitere Aspekte werden durch den Einsatz von *Content Management* berücksichtigt.

Die Zielumgebung Plone für die zu migrierende Website gehört innerhalb der Web-Anwendungen zu einer bestimmten Klasse von Zielsystemen, nämlich zu den so genannten *Content Management Systemen*. Bevor im Abschnitt 4.2 die Zielumgebung genauer beschrieben wird, werden in Abschnitt 1.4 die Begriffe Content und Content Management erläutert.

1.4.1 Content

Ein Content Management System beschäftigt sich, wie aus der wörtlichen Übersetzung entnommen werden kann, mit dem systematischen Verwalten von Inhalt. In Beschreibungen von Content wird dieser als *Information* aufgefasst, wobei Informationen mit Kontext angereicherte Daten sind. Boiko (2002) unterscheidet im Content Management Daten und Content und leitet die Unterscheidung durch verschiedene Perspektiven her. Aus der Sicht des Programmierers einer Website und des Computers handelt es sich immer um Daten. Dem gegenüber sucht der Benutzer stets Content, nämlich Informationen, die reich an Kontext und Bedeutung sind. So formuliert er die Gleichung, dass Content gleich „Information und *Metadaten*“ sind [Boiko (2002), S. 3].

Bei Metadaten handelt es sich nach Boikos Definition tatsächlich um Daten, da sie lediglich zur rechnergestützten Darstellung und Zugänglichmachung von Informationen verwendet werden sollen. Das Zielsystem Plone greift zur Verwaltung von Content ebenfalls auf solche

Metadaten zurück (siehe Abschnitt 1.5.3). Die folgende Auflistung gibt eine Grundlage für die Strukturierung von Metadaten [Boiko (2002), S. 461 / 462]:

- *Strukturmetadaten (structure metadata)*: Strukturmetadaten enthalten Angaben zur Struktur des Contents, den sie beschreiben. Darin werden Attribute und Funktionalität beschrieben, die den Content ausmachen. Sie umfassen den *eigentlichen Kern der Metadaten*.
- *Formatmetadaten (format metadata)*: Zu den hier abgelegten Metadaten gehören die Informationen, die die Formatierung von Elementen in der Präsentation und das Layout beschreiben. Sie werden auch Layout-Metadaten genannt.
- *Zugangsmetadaten (access metadata)*: Ähnlich wie die Strukturmetadaten beschreiben die Zugangsmetadaten die Struktur von Content. Sie befassen sich allerdings eher mit vorhandenen Zugriffsstrukturen auf Content, wie Verschlagwortung und hierarchische Angaben.
- *Verwaltungsmetadaten (management metadata)*: Hier befinden sich die Angaben, die sich weitestgehend mit dem allgemeinen Verständnis von Metadaten decken [Boiko (2002), S. 459]. Es handelt sich um Daten z.B. zu Titel, Autor, Erstellungsdatum, Änderungsdatum, Status, Größe, Besitzer, Veröffentlichungsdatum oder Auslaufdatum. Verwaltungsmetadaten wurden u.a. im Metadatenschema *Dublin Core* standardisiert, das in Plone implementiert ist. [2]
- *Einbeziehungsmetadaten (inclusion metadata)*: In diesen Metadaten wird festgelegt, wie auf referenzierte Elemente, wie Bilder und Hyperlinks, verwiesen wird.

Im Sinne einer einheitlichen und für diese Arbeit gültigen Definition soll *Content* als *rechnerverarbeitbare Information unter Einbeziehung von Metadaten* verstanden werden. Es handelt sich hierbei um Daten, die durch Kontext in eine semantische Struktur gebracht werden. Die daraus entstandenen Informationen werden wiederum mit Metadaten für den rechnergestützten Gebrauch verbunden. Metadaten können wie oben gesehen weiter aufgeschlüsselt werden und geben neben direkten Informationen zum Inhalt auch Anweisungen für Struktur und Layout an.[3]

Ausgehend von Content definiert sich ein weiterer Begriff, das *Asset*. Im Content Management wird darunter die Einheit von Content und den Nutzungsrechten an diesem Content beschrieben [Nakano (2001), S. 19]. Nun muss festgelegt werden, ob tatsächlich Content oder eher Assets den Gegenstand von Content Management darstellen. Es sollte nur Content veröffentlicht werden, für den der Verleger (im Englischen „publisher“) auch die notwendigen Rechte besitzt. Somit handelt es sich bei den veröffentlichten Elementen im legalen Idealfall immer um Assets. Allerdings wird die Verwaltung von Content und die Verwaltung der damit verbundenen Rechten, das in Form eines Digital Rights Managements durchgeführt werden kann, auf Grund der unterschiedlichen Domänen getrennt behandelt [Baumann (2003), S. 81]. Tatsächlich wird unter Content Management eine informatische und in keiner Weise juristische Aufgabenstellung verstanden.

1.4.2 Content-Typen

Content kommt in Content Management Systemen in verschiedenen Ausprägungen vor. Aus dem Umgang mit dem Internet sind dem Nutzer bereits Beispiele wie Webseiten, Bilder oder Informationen aus Suchabfragen bekannt. Um eine strukturierte Übersicht zu der Menge an Content zu geben, wird nach der allgemeinen Definition des Content-Begriffs nun eine Kategorisierung in Form von *Content-Typen* vorgenommen. Boiko beschreibt Content-Typen

[2] Zur Aufzählung und Beschreibung der Metadaten in Dublin Core sei auf http://dublincore.org verwiesen.
[3] Vgl. Gersdorf (2003), S. 63, die Bezeichnung „Inhalt“ sollte hierbei trotz der Nähe zum englischen Begriff „Content“ nicht verwechselt werden

auch als Content-Komponenten, verweist aber auf die Existenz mehrerer Begriffe zu Beschreibung des gleichen Gegenstand [Boiko (2002), S. 584].

Content-Typen können wie in der Objektorientierung als Klassen verstanden werden, die mit Attributen und Methoden ausgestattet sind. Diese Kategorisierung hilft dabei, Content nach unterschiedlichen Anforderungen besser zu verwalten. Zudem ist das Konzept der Content-Typen auch in der später erläuterten Zielumgebung Plone implementiert und stellt ein essenzielles Element bei der Durchführung der Migration dar.

Das *Dokument* ist einer der vielseitigsten und am häufigsten verwendeten Content-Typen. Beispiele hierfür sind Textdokumente in diversen Dateiformaten wie HTML, DOC, PDF, etc. In Plone bspw. existiert unter dem Content-Typ *Document* eine Vorlage für HTML-Seiten, die mit den Attributen Titel, Beschreibung und deren Body-Text belegt werden kann.

Obwohl auch Text Bestandteil von Multimedia ist, versteht man Bilder, Audio- und Videodateien als charakteristische *Multimedia-Formate*. Man unterscheidet diese Medien in Bezug auf die Sinne, die sie ansprechen (Gesichts- und Gehörsinn) und bezüglich ihres Verhaltens im Zeitverlauf (zeitdiskrete und kontinuierliche Medien). Sie sind in verschiedenen medienspezifischen Dateiformaten abgespeichert und werden üblicherweise in andere Content-Typen integriert. Plone stellt nur für den Import von Bildformaten standardmäßig den Content-Typ *Image* zur Verfügung. Hierfür kann man die Attribute Titel, Kurzbeschreibung und den Verweis auf das Bild selbst setzen.

Ähnlich wie in Dateisystemen dienen *Ordner* im Content Management zur Organisation und Strukturierung von Inhalten. Hierbei ist auch die Verteilung von Berechtigungen interessant, wobei anderen Nutzern nach bestimmten Regeln gestattet wird, in diesem Ordner Content zu verwalten. In Plone dienen Ordner, dort als Content-Typ *Folder* erfasst, außerdem für die automatische Erstellung der Navigation, so dass ausgehend von der vorgefundenen Ordnerstruktur ein hierarchischer Navigationsbaum erzeugt wird. Ordnern können in Plone lediglich die Attribute Titel und Beschreibung zugeteilt werden.

Zuvor wurde bei Dokumenten und typischen Multimedia-Formaten darauf hingewiesen, dass diese vor allem in *Dateien* gespeichert sind. Der Content-Typ „Datei“ ist aber allgemeiner, da er nicht spezielle Metainformationen und Funktionen wie für Textdateien und Bilder verfügt. Auch er ist in Plone standardmäßig als Content-Typ *File* enthalten und hat die Attribute Titel, Beschreibung und die zugewiesene Datei selbst.

Verweise verschiedener Art sind als Hyperlinks für das Internet essenziell. Beispiele für diese Verweise sind Links zu externen oder internen Seiten sowie Verlinkungen zu Email-Adressen oder Telefonnummern. Grundsätzlich kann jedes genannte Beispiel als eigener Content-Typ begriffen werden. Zu welcher Content-Typklasse er gehört, hängt von der Verwendung ab. Standardmäßig wird in Plone lediglich der Content-Typ *Link* angeboten, welcher für die Verlinkung mit externen Seiten gedacht ist. Seine Attribute in Plone sind ebenfalls eher einfach, umfassen Titel, Beschreibung und die URL.

Es fällt auf, dass die beschriebenen Content-Typen mit Ausnahme der Multimedia-Formate überwiegend Text beinhalten. Tatsächlich wird gemeinhin unter Content eher textueller Content verstanden [Gersdorf (2003), S. 64]. Vor allem von dieser Art von Content und durch Verbindung mit nicht-textuellen Content-Typen, wie z.B. Bildern, lassen sich etliche weitere Content-Typen bestimmen und ableiten. Lediglich der Zweckbezug setzt der Identifizierung von Content-Typen Grenzen. Im wissenschaftlichen Umfeld sind z.B. Literaturverweise, Angaben zu Arbeitsgruppen, Personen oder Lehrveranstaltungen denkbar und sinnvoll. Welche Content-Typen im Zusammenhang der durchzuführenden Websitemigration in Betracht kommen, wird später im Abschnitt 3 innerhalb der Legacy-Analyse geklärt.

1.4.3 Content Management

Unter Content Management wird das Erstellen, Aufbereiten und Ändern von Informationen in Form von Content verstanden [Zschau (2003), S. 51]. Dabei handelt es sich nicht originär um die Verwaltung von Websites, sondern umfasst ein allgemeineres Konzept zur Verwaltung von Content. Um die Funktionsweise des Content Management Systems Plone als Zielumgebung besser zu verstehen, wird an dieser Stelle die Begrifflichkeit des Content Management eingeführt.

Die Anforderungen an Content Management sind die *Wiederverwendbarkeit und konsistente Verwaltung* von Content durch eine saubere Trennung der gespeicherten Inhalte von der Art der Präsentation. Ein Beispiel hierfür wäre ein Dokument, das sowohl auf einem Computer als auch auf einem Mobiltelefon angezeigt werden soll. Außerdem sollte über die individuelle Zuweisung von Zugangsberechtigungen zu Content eine *Nutzerverwaltung* implementiert sein. Da Content in verschiedenen Formaten vorliegen kann, z.B. in HTML-Dokumenten, Textverarbeitungsdateiformaten, Datenbankabfragen oder Papierdokumenten, soll das Content Management auch eine *Integration der Content-Quellen* gewährleisten [Thome / Böhn (2006), S. 544]. Eine weitere wesentliche Komponente ist die Möglichkeit zur Implementierung von Publikationsregeln, *Workflows* genannt.

All diese Möglichkeiten des Content Management bringen allerdings einen erhöhten *Konfigurationsaufwand* mit sich. Je geringer der Nutzen aus den bereitgestellten Funktionen ist, desto weniger wird der Aufwand aufgewogen. Dies ist insbesondere bei kleineren und inhaltlich stabilen Informationsmengen der Fall, wie z.B. für kleinere Websites. Deshalb liegt angesichts dieser Kriterien die Annahme nahe, dass die Durchführung von Content Management in solchen Fällen sich im wirtschaftlichen Sinne nicht rentiert [Zschau (2003), S. 52].

Content Management Systeme haben die Aufgabe, die oben beschriebene Funktionalität zu implementieren. Gemeinhin wird davon ausgegangen, dass Content Management Systeme in einer auf den Technologiestandards des World Wide Web, d.h. unter Nutzung von HTTP und HTML, basierenden Umgebung agieren. In der Literatur spricht man dann von *Web Content Management Systemen* (WCMS) [Gersdorf (2003), S. 65]. In diesen WCMS kommen zur dynamischen Generierung der Anzeige von Content häufig Templates zum Einsatz.

Streng genommen erfüllt ein WCMS, wenn es sich ausschließlich auf die Internet-Umgebung ausrichtet, nicht die Anforderungen an die Integration von Content-Quellen [Gersdorf (2003), S. 66]. Um dies in vollem Umfang zu erreichen ist auch die Integration von Quellen notwendig, die nicht auf Internet-Technologien wie HTML basieren. Hierzu gehören u.a. Dateiformate wie PDF und DOC oder gescannte Papierdokumente [Thome / Böhn (2006), S. 544]. Auch Plone als zukünftige Zielumgebung der zu migrierenden Website ist ein WCMS. Es ist neben der Publikation von Webseiten auch in der Lage, PDF- und DOC-Dateien zu integrieren, stellt aber keine Funktionalität z.B. zur Texterkennung von gescannten Dokumenten zur Verfügung. Dennoch soll in dieser Arbeit mit dieser Einschränkung Plone als vollwertiges Web Content Management System angesehen werden. Zur Festlegung der Terminologie wird von Plone als Content Management System gesprochen.

1.5 Einführung in Plone

Der Abschnitt „Einführung in Plone“ dient dazu, ein besseres Verständnis des Zielsystems Plone zu vermitteln. Vor allem im Abschnitt 4 zum Ziel-Design wird auf diese Grundlagen zurückgegriffen und auch die Programmierung der Transformationen (siehe mitgelieferte CD) kann so besser nachvollzogen werden. Es werden die wesentlichen Elemente des Aufbaus von

Plone, die Formulierung von Templates sowie Content-Typen und das Rahmenwerk Archetypes zu deren vereinfachter Entwicklung näher behandelt.

Das Content Management System Plone baut auf den Web-Applikations-Server *Zope* und das darauf basierende *Content Management Framework* (CMF) auf. Alle diese Komponenten sind in der objektorientierten Programmiersprache Python geschrieben. Hierbei stellt Zope die grundlegenden Komponenten für Web-Applikationen zur Verfügung, wie u.a. einen Web-Server, ein Web-Interface und eine Objektdatenbank (siehe auch Abschnitt 1.3.1). Damit besitzt es aber von sich aus noch keine Funktionalität für Content Management. Es fehlen die typischen Elemente wie die Integration von Publikations-Workflows sowie eine saubere Trennung von Anwendungs- und Präsentationslogik. Diese kommt erst durch die Einbindung des CMF zu Stande. In Plone sind alle Elemente miteinander verbunden und so werden bei der Installation von Plone neben Zope auch das CMF automatisch mitinstalliert. Alle Angaben zum Zielsystem beziehen sich auf die zur Zeit der Erstellung dieser Abschlussarbeit aktuellen Versionen von Plone 2.5, Zope 2.9.4 und Python 2.4.3.

Bei Plone handelt es sich um Open-Source-Software und ist somit in der Anschaffung kostenlos. Durch das Open-Source-Konzept ist es auch möglich, die vorhandene Software zu verändern und weiterzuentwickeln.[4] In der Praxis kann das vorhandene Content Management System durch die Installation von so genannten *Produkten* in seiner Funktionalität erweitert werden. Diese Produkte sind modulare Erweiterungen des Content Management Systems und können selbst wiederum andere Produkte integrieren. Die Entwicklung kann selbst durchgeführt werden oder es können Produkte zu verschiedenen Anwendungsbereichen von der Website der Plone Stiftung heruntergeladen werden.[5]

Konzept von Plone

Das Zielsystem Plone dient nicht nur zur einfachen Verwaltung von Webseiten, sondern stellt zudem ein Portalkonzept dar. Dadurch wird das Content Management-Konzept realisiert und es kann eine Benutzerverwaltung und eine an die Nutzer angepasste Darstellung der Website eingerichtet werden.

Viele Aspekte bezüglich des Layouts sind in Plone bereits vordefiniert und können leicht abgewandelt werden. Durch die konsequente Anwendung von *Cascading Style Sheets* (CSS), von Internationalisierung und durch sein Template-System kann das Aussehen der Site an die eigenen Bedürfnisse angepasst werden.

Die wesentliche Arbeit mit Plone soll über die einfach zu bedienende Web-Benutzeroberfläche erfolgen. Und tatsächlich kann man eine Vielzahl der Operationen wie die Ordnerverwaltung, das Hinzufügen und Bearbeiten von Dateien, die Zuweisung der Zugriffsrechte als auch die für den Publikations-Workflow wichtige Statusverwaltung von Dokumenten darüber durchführen. Wenn aber z.B. das Erscheinungsbild der Website angepasst oder die Funktionalität durch die Installation von Produkten erweitert werden soll, muss direkt in Zope gearbeitet werden. Dafür steht die Zope-Benutzeroberfläche, das *Zope Management Interface* (ZMI) zur Verfügung. Das ZMI ist ein Werkzeug zur Verwaltung von Zope über eine Weboberfläche. Über den Browser kann auf das ZMI durch Anhängen von */manage* an eine gültige Adresse einer Plone-Website zugegriffen werden.

Generierung von Webseiten durch Plone

Wird eine Seite der Plone-Website abgerufen, kommen drei Elemente zum Einsatz. Zum einen generieren Vorlagen (*Zope Page Templates)* und darin integrierter *Python-Code* ein

[4] Vgl. zur Definition von Open Source Software http://www.opensource.org/docs/definition.php; nach der Definition der Open Source Initiative umfasst Kriterium 3 „Derived works“ die Möglichkeit zur Änderung und Weiterentwicklung von Open Source Software.

[5] Vgl. http://plone.org/products

HTML-Dokument, dessen endgültige Darstellung durch *CSS* festgelegt wird. Das Konzept der Verwendung von Templates zur dynamischen Erzeugung von Webseiten ist weit verbreitet. Allerdings unterscheidet sich die Architektur von Templates und deren Handhabung bei Plone erheblich von gängigen Technologien, wie ASP, Perl oder PHP. So ist in Plone jeder veröffentlichte Content der Website ein Objekt („*object publishing*“). Wie in objektorientierten Programmiersprachen üblich, besitzen diese Objekte Attribute und Operationen, die abgerufen werden können.

1.5.1 Templates

Templates dienen zur dynamischen Darstellung von Webseiten. Sie fügen Daten, die dem Template als Variablen übergeben werden, an bestimmten Stellen in die generierte Webseite ein. Damit sind Templates ein zentraler Bestandteil bei dem Design der Website und der Darstellung des darin enthaltenen Contents. Die Anwendung der hier beschriebenen Mechanismen kann in Abschnitt 4.3.2 zum Entwurf der Benutzungsschnittstellen nachvollzogen werden.

Template Attribute Language (TAL)
Für die Einbindung von Template-Variablen kann auf die *Template Attribute Language* (TAL) zurückgegriffen werden. Diese Ausdrücke werden nach den Regeln der *Templates Attribute Language Expression Syntax* (TALES) formuliert. Bereits im Standardquelltext des neu erstellten Templates kann man einige dieser TAL-Ausdrücke identifizieren, wie ausschnittsweise in Abbildung 6 dargestellt. Hier wird im Titel-Tag der Webseite über den *tal:content*-Aufruf festgelegt, dass, falls für das Template ein Titel eingetragen wurde, dieser statt *The title* ausgegeben wird. TAL-Ausdrücke werden in HTML-Tags eingefügt und führen bei Anzeige eines Templates abhängig von den angegebenen Attributen unterschiedliche Aktionen aus. So überschreibt das Attribut *tal:content* bei Vorhandensein des Wertes den durch den Tag eingegrenzten Text.

```
<title tal:content="template/title">The title</title>
```

Abbildung 6: Beispiel eines TAL-Ausdrucks im Template-Quelltext

In dem Template lassen sich neben einfachen Zuweisungen auch Kontrollstrukturen realisieren. Eine Verzweigung kann mit dem TAL-Attribut *tal:condition* eingefügt werden. Damit wird überprüft, ob die eingegebene Bedingung erfüllt ist oder nicht. Diese Überprüfung ist abhängig von den überprüften Werten und gibt auch dann *Falsch* zurück, wenn z.B. die Zahl *0* oder eine leere Zeichenkette („“) übergeben wird. Durch das Voranstellen des Präfixes *not:* für den überprüften Wert kann die Aussage invertiert werden. Abbildung 7 stellt eine beispielhafte Anwendung des *condition*-Attributs dar.

```
<p tal:condition="template/title">Es ist ein Titel vorhanden.</p>
<p tal:condition="not: template/title">Es ist kein Titel vorhanden.</p>
```

Abbildung 7: Beispiel der Verwendung des *condition*-Attributs

Neben der bedingten Anzeige durch *tal:condition* kann mit *tal:repeat* eine listenartige Template-Variable in einer Schleife durchlaufen werden. In dem Beispiel in Abbildung 8 wird die Liste des Portal-Katalogs durchlaufen und der Titel des Elements ausgegeben (näheres zum Katalog in Abschnitt 1.5.3). Durch Anwendung der Variable *roman* des *repeat*-Attributs (verwiesen in *tal:content=“repeat/iterationsvariable/roman“*) wird dem Titel jedes Elementes noch die laufende Zahl des Schleifendurchlaufs im römischen Zahlen vorangestellt. Dadurch, dass sich das *repeat*-Attribut im *p*-Tag befindet, werden alle

Anweisungen innerhalb dieses Tags während der Iteration ausgeführt und schließlich ein neuer Absatz gebildet.

```
<p tal:repeat="iterationsVariable context/portal_catalog">
<span tal:content="repeat/iterationsVariable/roman">Hier erscheint eine Nummerierung mit römischen Buchstaben</span>
<span>: </span>
<span tal:content="iterationsVariable/Title">Title</span>
</p>
```

Abbildung 8: Beispiel der Verwendung des *repeat*-Attributs

Mit der beschriebenen Funktionalität von TAL lassen sich bereits dynamisch Webseiten generieren. Problematisch verbleibt allerdings der Wiederverwendungsaspekt. Um die Ausgabe eines Templates mit TAL auf ein anderes Template zu übertragen, müsste man die verwendeten TAL-Blöcke in das neue Template kopieren. Dies kann aber in Hinsicht auf Wartbarkeit und die angestrebte leichte Wiederverwendung nicht zielführend sein. Hierfür stellt Plone einen Mechanismus namens *Macro Expansion Attribute Language* (METAL) zur Verfügung.

Makros und METAL

Schlüssel zur Nutzung von METAL sind die komplementären Funktionen *define-macro* / *use-macro* und *define-slot* / *fill-slot*. Über *define-macro* wird ein *Makro*, also ein später in anderen Templates wieder zu verwendender Abschnitt, definiert. *Use-macro* wiederum ermöglicht die Wiederverwendung des in einem anderen Template definierten Makro-Blocks unter Angabe des korrekten Pfades. *Define-slot* beschreibt einen *Slot*, also einen Bereich, in den bei Aufruf des Makros eigener Text oder ein weiteres Makro eingefügt werden kann. Die Einfügeoperation erfolgt analog zum Makro mit *fill-slot*. Ein Slot muss immer zu einem Makro gehören und wird deshalb innerhalb eines Makro-Blocks definiert. Die folgenden Abbildungen stellen eine beispielhafte Anwendung dar. Die Fehlermeldungen in den Templates werden überschrieben, wenn der Slot befüllt wird. Kommentare sind durch # gekennzeichnet.

```
<html>
        <body>
                # Definition des Makros “templateTitel”
                <div metal:define-macro="templateTitel">
                        # Definition des Slots “titelText”
                        <div metal:define-slot="titelText">
                                Beschreibender Text zum Titel
                        </div>
                        <div tal:content="template/title">
                                Fehler: Es konnte kein Titel gefunden werden.
                        </div>
                </div>
        </body>
</html>
```

Abbildung 9: Quellcode des definierenden Templates *makroTest*

```
<html>
        <body>
                # Aufruf des Makros “templateTitel”
                <div metal:use-macro="here/makroTest/macros/templateTitel">
                # Ausfüllen des Slots “titelText”
                <div metal:fill-slot="titelText">
                        Der Titel dieses Templates lautet:
                </div>
                        Fehler: Es konnte kein Titel gefunden werden.
                </div>
        </body>
</html>
```

Abbildung 10: Quellcode des aufrufenden Templates *Testtemplate*

Wenn *Testtemplate* aufgerufen wird, dann erscheint im Browser folgender Text:

```
Der Titel dieses Templates lautet:
Testtemplate
```

Abbildung 11: Ausgabe des Templates *Testtemplate* im Browser

In Abbildung 10 wird das Makro *templateTitel* über den Ausdruck *here/makroTest/macros/templateTitel* aufgerufen. Das Ende des Pfadausdrucks verweist auf das Template *makroTest*, wobei sich in den darin definierten Makros auch das Makro *templateTitel* befindet. Das vorangestellte *here* steht dabei für den so genannten *Kontext*. Dieser beschreibt, in welcher Umgebung ein Objekt in Plone aufgerufen wird und ist ein Pfadausdruck [McKay / da Silva (2006), S. 105]. In diesem Bespiel bedeutet *here*, dass das Makro zuerst in der Umgebung gesucht wird, in dem sich auch das aufrufende Template befindet. Pfadausdrücken in Plone werden häufig die Ausdrücke *here* oder das äquivalente *context* vorangesetzt.

Kombination von Makros und Slots

Die Verschachtelung von Templates über Makros und Slots wird in Plone fast immer verwendet. Um sich das entwickelte Template anzeigen zu lassen, muss es im ZMI gespeichert werden, z.B. in einem Benutzer-Ordner unter dem Dateinamen *index_html*. Durch diese Bezeichnung wird sichergestellt, dass sich das Template bei Aufruf der Ordneradresse automatisch als Startseite öffnet. Würde man nun tatsächlich das Template mit diesem Quellcode ablegen und es über den Browser öffnen, wird statt dem gewohnten Portal-Layout ausschließlich der in Abbildung 11 aufgeführte Text erscheinen. Die Ursache ist, dass das Portalgrundgerüst einer Plone-Website ebenfalls über die beschriebenen Definitionen und Aufrufe der Makros und Slots entsteht. Diese werden im so genannten *main-Template* beschrieben. Um also den Portal-Charakter der Website bei Aufruf des Templates zu erhalten, muss man die Struktur des main-Templates in das aufzurufende Template integrieren. Dies erfolgt normalerweise über das Master-Makro, das in diesem Beispiel durch *use-macro="here/main_template/macros/master"* zu erreichen ist. Nun können die Slots des main-Templates befüllt werden. Da die Ausgabe des Test-Templates in der Mitte erscheinen soll, muss der *main*-Slot befüllt werden. Eine Variante für diese Implementierung ist in Abbildung 12 dargestellt.

```
<html>
        <body>
                <div metal:use-macro="here/main_template/macros/master">
                        <div metal:fill-slot="main">
                                <div metal:use-macro="here/makroTest/macros/templateTitel">
                                <div metal:use-slot="titelText">
                                        Der Titel dieses Templates lautet:
                                </div>
                                        Fehler: Es konnte kein Titel gefunden werden.
                                </div>
                        </div>
                </div>
        </body>
</html>
```

Abbildung 12: *Testtemplate* mit Einbindung in die Portalstruktur der Website

Struktur des Plone-Portals

Die Portalstruktur, die Plone bereits standardmäßig zur Verfügung stellt, ist komplex. Neben den Makros und Slots, die bei Nutzung sichtbar werden, definiert man auch Slots, die auf Einstellungen zur Darstellung und Funktionalität der Seite verweisen (Slots für den HTML-Head, CSS und JavaScript Head). Eine interessante Konstruktion stellen die Slots für die linke und die rechte Spalte der Portalseite dar. Hier werden in den definierten Slots bereits Makros benutzt, die angezeigt werden, wenn der Slot durch ein anderes Template nicht befüllt wird. Ansonsten erscheinen die verwendeten Makros, üblicherweise Portlets (kleine Boxen mit Inhalt), z.B. für die Navigation, den Login oder den Kalender. Nach der Nummerierung in

Abbildung 13 sind im Folgenden die in der Portalstruktur von Plone genutzten bzw. definierten Makros und Slots kurz aufgeführt. Die Rahmen in der Abbildung heben in schwarz aufgerufene Makros, in dunkelgrau durch das aufrufende Template befüllte Slots und in hellgrau die bereits im main-Template mit aufgerufenen Makros gefüllten Slots hervor.

1. das Logo der Website (aufgerufen durch das Makro *portal_logo*)
2. Portal-Reiter (Makro *portal_tabs*)
3. Pfadleiste (Makro *path_bar*)
4. Website-Aktionen (Makro *site_actions*)
5. Schnellsuche (Makro *quick_search*)
6. Persönliche Leiste (Makro *personal_bar*)
7. Linke Spalte (Slots column_one_slot und portlets_one_slot, Makro left_column)
8. Hauptbereich (Slot main)
9. Rechte Spalte (Slots column_two_slot und portlets_two_slot, Makro right_column)
10. Sichten auf den Content (Makro *content_views*)
11. Aktionen auf den Content (Makro *content_actions*)
12. Nachricht (Makro *portal_message*)
13. Fußzeile (Makro *footer*)
14. Kolophon (Makro *colophon*)
15. die gesamte Seite (Makro *master*)

Auf dieser Abbildung nicht sichtbar sind neben den oben genannten Slots der Header-Slot (*header*) und Sub-Slot (*sub*) über, bzw. unter dem Hauptbereich.

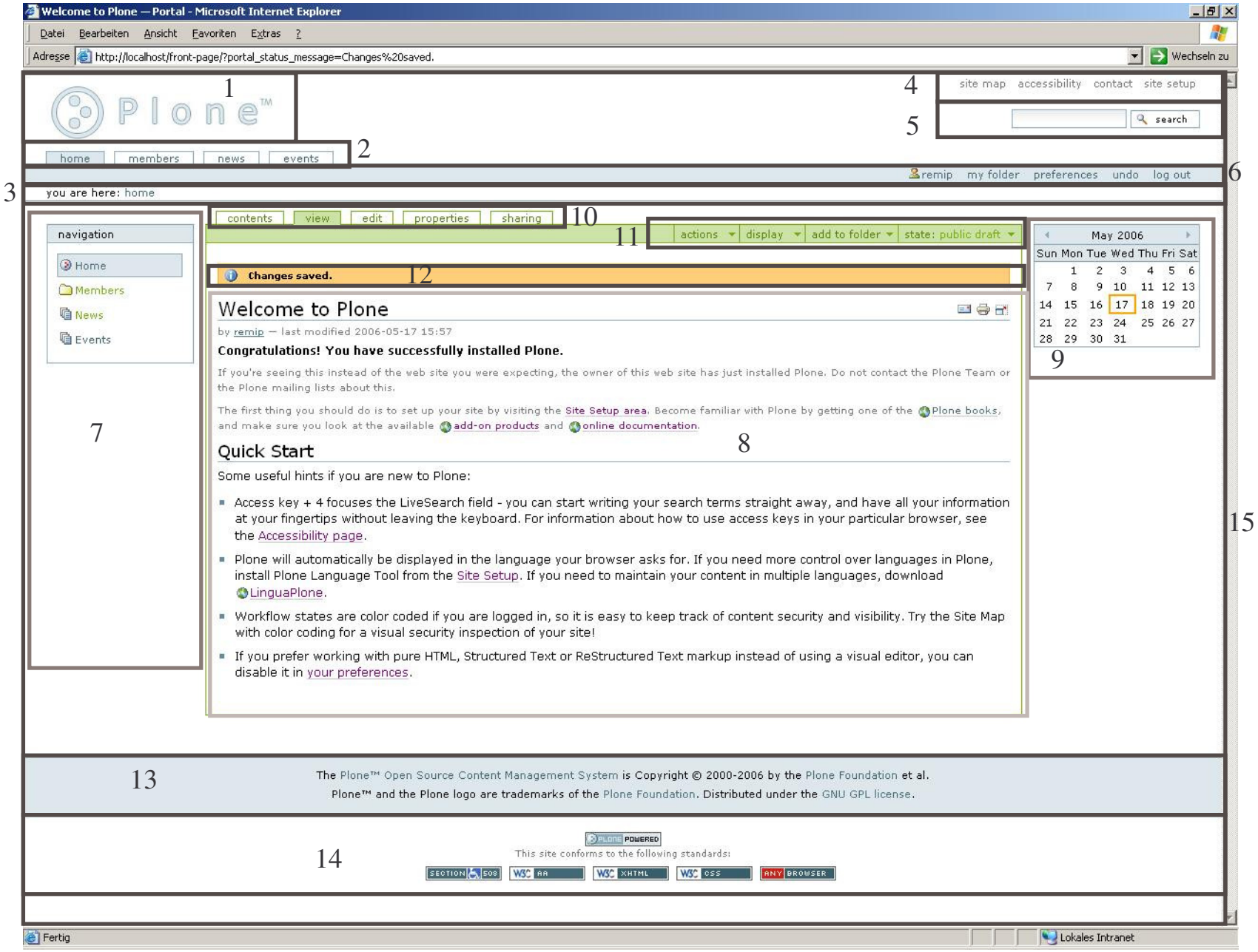

Abbildung 13: Portalstruktur mit Makros und Slots

Ganz alleine mit den beschriebenen Mechanismen TAL und METAL können allerdings keine komplexen Inhalte in den Templates erzeugt werden. Gerade wenn Objekte nach bestimmten Kriterien ausgewählt und Operationen auf ihnen durchgeführt werden sollen, greifen die einfachen Verweise auf ihre Attribute zu kurz. Hierfür kann man sich in Plone des *Scriptings* in Python bedienen. Es muss lediglich, ähnlich wie bei den Templates, ein entsprechendes Scripting-Objekt über das ZMI erzeugt werden und der Code kann eingegeben werden. Durch die Auslagerung der Anwendungslogik ins Script kann eine bessere Trennung zwischen der Funktionalität und der Präsentationslogik im Template erreicht werden.

Durch die Verwendung von METAL können Templates in Makro-Blöcke und Slots unterteilt werden, die ihrerseits durch TAL-Ausdrücke oder auch durch Scripting dynamisch befüllt werden. Wie nun die einzelnen Templates und Makros in Plone organisiert werden, bleibt dem Benutzer weitestgehend selbst überlassen. Empfehlenswert ist die Nutzung des *portal_skins/custom*-Verzeichnisses in der ZMI zur Sammlung selbstentwickelter Templates und Scripts. Auch wenn die von Plone standardmäßig angebotenen Elemente verändert werden sollen (über das Anklicken des Buttons *customize*), ist dies über das ZMI nur durch das Erstellen einer Kopie im *custom*-Ordner möglich.

1.5.2 Content-Typen in Plone

Bereits in der Definition des Begriffs Content wurde herausgestellt, dass die Identifizierung von Content-Typen im Content Management eine zentrale Rolle spielt. Auch in Plone wird die Formalisierung von Content-Typen unterstützt. Man kann nicht nur auf die bereits vorhandenen Content-Typen zurückgreifen, sondern auch eigene, dem Bedarf angepasste Content-Typen schreiben. Eine Übersicht über die installierten Content-Typen findet man im ZMI im Verzeichnis *portal_types*. Darüber lassen sich für den jeweiligen Content-Typ die Konfiguration, systeminterne Methoden-Alias-Zuweisungen, die vorgenannten Content-Typ-eigenen Aktionen und Sicherheitseinstellungen bedienen.

Die gesamte Beschreibung eines Content-Typen ist in einer Python-Datei (.py) im Dateisystem von Plone gespeichert. Darin befindet sich auch in Form eines Dictionaries die Konfiguration des Content-Typen, die so genannte *Factory Type Information*, die die wichtigsten Eigenschaften eines Content-Typen beschreibt.[6] Die Definition Eigenschaftsfelder der *Factory Type Information* sind bei allen Content-Typen gleich. Es handelt sich um ein Meta-Schema. Nur die Belegung der Eigenschaftswerte unterscheidet sich zwischen den Content-Typen. Hierzu gehören u.a. die ID, der angezeigte Name, die zugewiesenen Aktionen (siehe Abschnitt 1.5.4), ebenfalls in Form eines Dictionaries, und die zur Erstellung einer Instanz notwendige Factory-Methode, der Konstruktor. In der *Factory Type Information* befinden sich außerdem die Methoden-Aliase, die z.B. auf Standardmethoden für Ansicht und Bearbeitung verweisen (siehe Abschnitt 1.5.4). In Abbildung 14 ist die leicht veränderte *Factory Type Information* einer regulären Webseite (*Document*) abgebildet:

[6] Vgl. McKay / da Silva (2006), S. 336; Ein Dictionary ist ein Python-Datentyp, der eine Auflistung von Schlüssel- / Wertepaaren darstellt. Begrenzt wird ein Dictionary durch geschweifte Klammern, in denen die Schlüssel- / Wertepaare durch Doppelpunkte einander zugeordnet werden und die verschiedenen Paare durch Kommata getrennt werden.

```
factory_type_information = (
 { 'id'          : 'Document'
 , 'meta_type'   : 'Document'
 , 'description' : """\
Documents contain text that can be formatted using 'Structured Text.'
They may also contain HTML, or "plain" text.
"""
 , 'icon'      : 'document_icon.gif'
 , 'product'    : 'CMFDefault'

# Factory-Methode zur Erzeugung eines Objekts des Typs Document
 , 'factory'    : 'addDocument'
 , 'immediate_view' : 'metadata_edit_form'
 , 'aliases'    : {'(Default)':'document_view',
            'view':'document_view',
            'gethtml':'source_html'}

# Document zugewiesene Aktionen
# view ist der Verweis auf das Ansichts-Template
 , 'actions'    : ( { 'id'      : 'view'
             , 'name'     : 'View'
             , 'action': 'string:${object_url}/document_view'
             , 'permissions'  : (View,)
             }
# edit ist der Verweis auf das Bearbeiten-Template
            , { 'id'      : 'edit'
             , 'name'     : 'Edit'
             , 'action': 'string:${object_url}/document_edit_form'
             , 'permissions'  : (ModifyPortalContent,)
             }
# metadata ist der Verweis auf das Metadaten-Template
            , { 'id'      : 'metadata'
             , 'name'     : 'Metadata'
             , 'action': 'string:${object_url}/metadata_edit_form'
             , 'permissions'  : (ModifyPortalContent,)
             }
            )
 }
,
)
```

Abbildung 14: Factory Type Information von Document.py

Neben der notwendigen Konstruktor-Methode (in diesem Fall *addDocument)*, befindet sich in der Datei des Content-Typs auch die Klassenbeschreibung. Dort sind neben Methoden auch deren Sicherheitseinstellungen definiert. Damit diese von Plone übernommen werden, muss die Klasse *Document* initialisiert werden. Die erfolgt über den Aufruf *InitializeClass(Document)* [McKay / da Silva (2006), S. 340].

In Plone gibt es Objekte, andere Objekte enthalten können. Diese Objekte werden in diesem Zusammenhang *Container* genannt. Die in den Containern befindlichen Objekte, übernehmen nicht nur von ihren Klassen Eigenschaften, sondern auch von ihren Containern [McKay / da Silva (2006), S. 106].

1.5.3 Verwaltung und Zugriff auf Objekte in Plone

Die im CMS erstellten Instanzen von Content-Typen werden in Plone im Katalog *portal_catalog* erfasst, der von Zope zur Verfügung gestellt wird. Durch Aufruf von *portal_catalog* innerhalb des ZMI können sie unter dem Reiter *Catalog* vollständig eingesehen werden. Der Katalog ist eine erweiterte Version des ZCatalog-Werkzeugs von Zope [McKay / da Silva (2006), S. 315]. In einer Plone-Site erfüllt er drei wichtige Aufgaben. Zum einen indiziert er den Content und macht ihn so durchsuchbar. Außerdem speichert er die Content-Metadaten. Zuletzt stellt er eine Schnittstelle für die Suche nach Content zur Verfügung.

Metadaten in Plone

Metadaten werden häufig zur Suche und Kategorisierung von Content in Plone verwendet [McKay / da Silva (2006), S. 50]. Sie umfassen optionale Daten für jedes Content-Objekt innerhalb von Plone und orientieren sich überwiegend am Metadatenschema *Dublin Core*. Im Vergleich mit der Auflistung von Metadaten im Abschnitt 1.4.1 gehören sie vor allem zu den

Verwaltungsmetadaten. In Plone werden vor allem folgende Metadaten von Content-Objekten verwendet (entnommen aus McKay / da Silva (2006), S. 50 und aus der Betrachtung des Katalogs):

- *id*: eindeutige Identifizierung eines Objekts innerhalb von Plone (Dublin Core)
- *title*: der Titel des Content-Objekts (Dublin Core)
- *description*: eine kurze Beschreibung des Inhalts des Content-Objekts (Dublin Core)
- *keywords*: Angabe von Schlagworten, die den Inhalt des Content-Objekts gut beschreiben (Dublin Core)
- *creator*: Angabe der für die Erstellung des Content-Objekts verantwortlichen Personen (Dublin Core)
- *exclude from navigation*: Festlegung, ob das Content-Objekt in der Navigation angezeigt wird

Indizierung und Suchen

Die *Indizierung* dient der effizienten Bearbeitung von Suchabfragen. Die Indizes, die von Plone standardmäßig zur Verfügung gestellt werden, sind im ZMI im Verzeichnis *portal_catalog* unter dem Reiter *Indexes* abrufbar. Hierbei handelt es sich fast ausschließlich um Verwaltungsmetadaten (vgl. Abschnitt 1.4.1). Jeder Index verweist wiederum auf einen Indextyp, der die Art der in ihm gehaltenen Daten angibt.[7] Bei Bedarf können weitere Felder zur Indizierung über die ZMI hinzugefügt werden. Hierbei muss beachtet werden, dass mit steigender Anzahl von Indizes, sich auch die Durchsuchungsgeschwindigkeit im Katalog verringert.

Wurden Änderungen im Quellcode von Plone gemacht, die die Indizierung betreffen, wie z.B. die Installation von neuen Produkten, sollte eine Reindizierung durchgeführt werden. Dies kann für einzelne Indizes durch den Button *Reindex* oder aber für den gesamten Katalog durch den Button *Update Catalog* unter dem Reiter *Advanced* erfolgen.

Soll der Katalog auf bestimmte Begriffe durchsucht werden, so kann auf Programmierebene eine *searchResults*-Abfrage formuliert werden.[8] Beim Durchführen der Suchabfrage werden nur die Indizes nach Kriterien durchsucht. Möchte man allerdings nach einer bestimmten Eigenschaft eines Content-Typen suchen, muss dieses explizit in der Liste der Indizes vermerkt sein und vor der Suche eine Aktualisierung der Indizes durchgeführt worden sein.

Unter den Indizes hat der Index *SearchableText* eine Sonderstellung. In dieses Feld werden Werte aus Eigenschaftsfeldern von Content-Typinstanzen eingetragen, die die Anweisung *searchable = 1* enthalten. Dadurch kann nach Werten, die in einem Eigenschaftsfeld vorkommen, gesucht werden, ohne für die Eigenschaft explizit einen Index zu definieren. Allerdings kann nicht präzise nach Content gesucht werden, der die Daten in diesem bestimmten Eigenschaftsfeld enthält. Es werden lediglich die Werte im Metadatenfeld *SearchableText* eingetragenen und damit innerhalb der Website durchsuchbar gemacht.

In bestimmten Fällen kann die explizite Indizierung der Eigenschaftsfelder von Content-Typen sinnvoll sein. Dies ist dann der Fall, wenn gezielt nach den Inhalten der Felder gesucht werden soll. Das oben beschriebene Vorgehen mit der Anweisung *searchable* ist hier keine Lösung, da die Inhalte mehrerer Eigenschaften im Metadatenfeld *SearchableText* vermischt werden. Hierfür muss ein Index eingerichtet werden, der den Namen der Akzessormethode des Eigenschaftsfelds (siehe weiter unten in diesem Abschnitt) besitzt.

[7] Zur Übersicht der verfügbaren Indextypen vgl. McKay / da Silva (2006), S. 315 / 316.
[8] In der vorliegenden Plone-Version ist der dafür notwendige vollständige Funktionsaufruf „context.portal_catalog.searchResults“

Die Funktionsweise einer *searchResults*-Abfrage ist in Abbildung 15 beispielhaft dargestellt. Die direkte Suche nach Attributwerten erfolgt hier über *title = „Plone“*. Festgehalten werden muss, dass es sich nicht um eine exakte Suchabfrage handelt. Es werden auch Titel ausgegeben, die *Plone* nur als Teilzeichenkette enthalten. Über die Abfrage nach Date (dies ist das Erstellungsdatum des Contents bzw. das Datum der letzten Änderung) werden die Metadaten der Objekte zurückgegeben, die nach dem 01.05.2006 erstellt wurden. Dieser Ausdruck, der wieder ein Dictionary darstellt, ist etwas komplexer. Er besteht aus einer Liste *query* und einer Bereichsvergleichoperation *range*. Im Beispiel wird der Variablen *query* das Datum in Form einer Liste aus einem Element übergeben, das mit *range:'min'* kombiniert wird, also „Datum mindestens 01.05.2006“ vergleicht. Am Schluss wird noch die Ausgabe der Abfrage nach *title* aufsteigend sortiert.

```
context.portal_catalog.searchResults(
        title = "Plone",
        Date={“query”:[DateTime(‘2006/05/01’)],
              “range”:”min”},
        sort_on="title"
)
```

Abbildung 15: Beispiel einer searchResults-Abfrage

Erzeugung und Löschen von Objekten

Um innerhalb eines Skripts Instanzen von Content-Typen zu erzeugen, kann man sich der Methode *invokeFactory* bedienen. Viele Objekte in Plone können über die Verwendung dieser Methode Instanzen neuer Objekte erzeugen. Abbildung 16 stellt dies exemplarisch an einem normalen Webseiten-Objekt (*Document*) dar.[9] Dort steht die Variable *context* für ein Objekt, das über seine *invokeFactory*-Methode neue Objekte des Typs *Document* generieren kann.

```
# Erzeugung einer neuen ID
newId = context.generateUniqueId('Document')
# Aufruf der Konstruktor-Methode
context.invokeFactory(id=newId, type_name='Document')
# Zuweisung der erzeugten ID zu einem Nachrichtenobjekt
newDocument = getattr(context, newId)
return "Done"
```

Abbildung 16: Skript zur Erzeugung eines Objekts des Typs *Document*

Ebenso wie die Erzeugung ist auch das Löschen von Objekten von Belang. In Plone kann dies mit der Methode *manage_delObjects* erreicht werden, wie Abbildung 17 an einem *Document*-Objekt veranschaulicht.[10] Auch hier ist es Voraussetzung, dass das durch *context* repräsentierte Objekt in der Lage ist, eine Löschoperation für Objekte des Typs *Document* durchzuführen.

```
# documentID ist die ID eines Objekts vom Typ Document
context.manage_delObjects(["document"])
```

Abbildung 17: Löschen eines Objektes

Zudem existieren bei jedem Content-Typen Zugriffsmethoden, die das Lesen und Schreiben des Werts eines Eigenschaftsfelds erlauben. Diese Zugriffmethoden werden im Zusammenhang mit Plone Akzessor- und Mutator-Methoden genannt. Plone vergibt für diese Methoden standardmäßige Namen, die folgende Syntax besitzen:

- Akzessor: get[Attributname]()
- Mutator: set[Attributname]()

[9] Dieses Beispiel wurde verändert aus McKay / da Silva (2006), S. 313 entlehnt.
[10] Vgl. http://plone.org/documentation/how-to/manipulating-plone-objects-programmatically/

Wenn man dennoch andere Namen für die Zugriffsmethoden definieren will, kann man dies durch Veränderung der Einstellungen der Content-Typen (näheres in den Abschnitten 1.5.2 und 1.5.5) erreichen.

1.5.4 Aktionen in Plone

Aktionen (Actions) in Plone definieren Operationen, die der Nutzer *über die Benutzeroberfläche der Website* durchführen kann. Sie stellen also eine Teilmenge aller Methoden dar, die ausgeführt werden können. Besonders wichtige Aktionen im Zusammenhang mit Content-Typen sind *view* und *edit*. Sie rufen Templates für eine Ansicht, bzw. ein Bearbeitungsformular der Content-Typ-Instanz auf. Diese Templates werden bei der Verwaltung von Content häufig genutzt. Für die Darstellung werden häufig Reiter verwendet, wie sie in den Makros *content_actions* und *site_actions* auf der Startseite des Portals zu finden sind.

Aktionen werden hierbei entweder von der Website selbst, von den Content-Typen oder auch von installierten Produkten zur Verfügung gestellt. Dementsprechend sind die verfügbaren Aktionen in Plone dezentral verteilt. Bei *portal_actions* unter dem Reiter *Actions* finden sich Aktionen, die für die gesamte Website gelten. Bei Auswahl des Reiters *Action Providers* werden alle Werkzeuge angezeigt, die Aktionen definieren. Dazu gehört auch das Werkzeug *portal_types*, wo nach Auswahl des jeweiligen Content-Typs unter dem Reiter *Actions* die nur für diesen Content-Typ definierten Aktionen angezeigt werden.

Die Operationen, die die Aktionen ausführen, sind im Aktionseigenschaftsfeld *URL (Expression)* festgelegt. Hierüber kann eine einfache Verlinkung zu einer anderen Seite erfolgen, dynamisch ein TALES-Ausdruck ausgewertet (siehe Abschnitt 1.5.1 zu Templates) oder auch ein Python-Skript aufgerufen werden. Nach diesem Vorbild stellt die Weboberfläche von Plone dem Nutzer mit durch Anklicken verfügbaren Aktionen eine Vielzahl für das Content Management relevanter Operationen zur Verfügung.

Im Ansichtsmodus wird Content angezeigt, was häufig die Standardaktion für Content-Typen ist. Die *view*-Aktion entspricht der standardmäßigen Anzeige der Webseite. Dem hingegen wird durch Aufruf der *edit*-Aktion das Bearbeitungsformular angezeigt, das schnelle Änderungen der Instanz des Content-Typs ermöglicht. Ein beispielhaftes Formular zur Bearbeitung einer Webseite über Plone stellt Abbildung 18 dar.

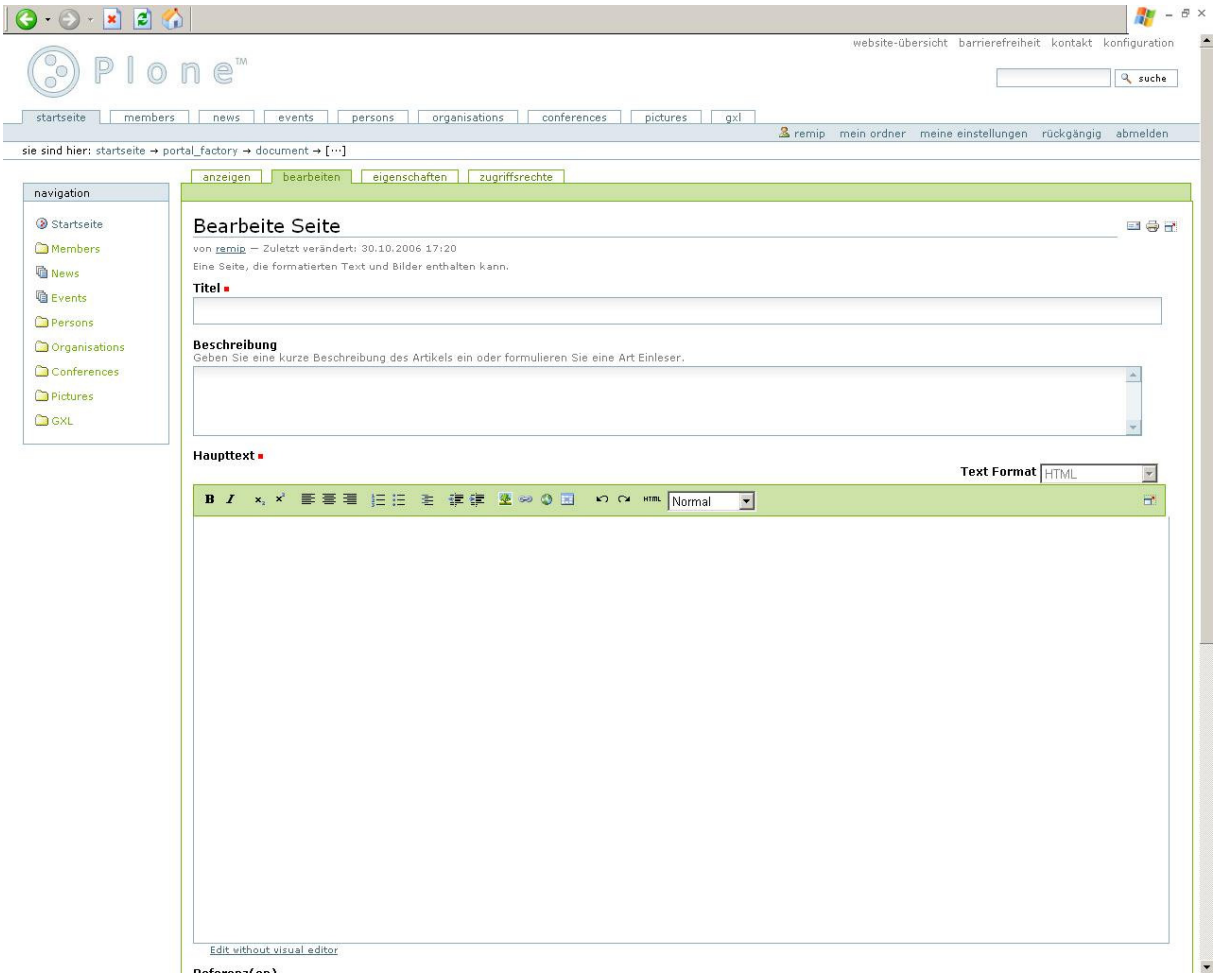

Abbildung 18: Formular zur Bearbeitung einer Webseite

1.5.5 Archetypes

Tatsächlich ist Programmierung eines Content-Typen mitunter recht kompliziert und zeitaufwändig. Hierzu gehört nicht nur die Definition des Content-Typen in der PY-Datei mit allen Angaben der *Factory Type Information*, den zugewiesenen Aktionen und den Sicherheitseinstellungen, sondern auch die Generierung der Sichten auf die Objekte, also die Erstellung der Templates. Auf Grundlage des CMF wurde zur Unterstützung der Entwicklung von Content Typen das Framework *Archetypes* entwickelt. Hierbei werden die Informationen ebenfalls als PY-Dateien im Dateisystem abgespeichert, besitzen aber eine abgewandelte Struktur, auf die später eingegangen werden wird.

Ein wesentlicher Vorteil bei der Entwicklung über Archetypes ist die Bereitstellung diverser Dienste, die nach Definition des Content-Typen vom Framework geleistet werden. An Hand der Beschreibung in Archetypes kann Plone dynamisch *view*- und *edit*-Templates erzeugen. So kann Content direkt nach der Erstellung der Beschreibung des Content-Typs bearbeitet und betrachtet werden. Diese Templates sind nicht als Template-Code abgelegt und können bei Bedarf durch die Definition eines neuen Templates übergangen werden. Durch Archetypes ist außerdem eine einfache Implementierung von Referenzen zwischen Objekten zu bewerkstelligen. Dies kann vor allem bei komplexeren Content-Typen eine große Hilfe bei der Verwaltung der Objekte sein. Außerdem stellt Archetypes eine Schnittstelle zur Speicherung von Daten zu Content-Typen in externen relationalen Datenbanken zur Verfügung, so dass man nicht ausschließlich auf die von Zope bereitgestellte Objektdatenbank angewiesen ist.

Ein abschließendes Argument für die Nutzung von Archetypes ist die Möglichkeit, automatisch Content-Typen auf Archetype-Basis aus UML-Klassendiagrammen entwickeln zu lassen. Hierfür wird zunächst ein UML-Modellierungsprogramm benötigt, das das erzeugte Klassendiagramm unter Nutzung von XML Metadata Interchange (XMI) exportieren kann. In dieser Arbeit wird hierfür *ArgoUML* verwendet, ein Open-Source-Produkt.[11] Diese Datei wird schließlich mit dem Plone-Produkt *ArchGenXML* importiert, das davon ausgehend den neuen Content-Typ generiert. Leider ist es derzeit nicht möglich, in Plone vorhandene und auf Archetypes basierende Content-Typen in XMI-Dateien zu exportieren und somit wieder in Diagramme zu konvertieren.[12]

Die Entwicklung von Content-Typen mit Archetypes wird auch als schema-basierter Ansatz beschrieben [Ritz (2005), S. 28]. Die Idee dahinter ist, dass für einen Content-Typ ein *Schema* definiert wird und dieses an ein bereits existierendes angehängt wird, ähnlich dem Prinzip der Vererbung in der objektorientierten Programmierung. Auf diese Weise erbt es die Eigenschaften des angehängten Schemas und die dafür zur Verfügung gestellten Automatismen, wie sie oben beschrieben wurden.

Als *Grundschemata* („stock schemas", [Limi (2005)]), von denen per Archetypes entwickelte Content-Typen abgeleitet werden können, kann auf *baseSchema*, *baseFolderSchema* und *baseTreeFolderSchema* zurückgegriffen werden. Hierbei wird *baseSchema* für gewöhnliche Content-Typen verwendet und *baseFolderSchema* bzw. *baseTreeFolderSchema* für Content-Typen, die selbst Instanzen von anderen Content-Typen enthalten.[13] Die letzteren Content-Typen werden auch *Folderish Objects*, ordnerartige Objekte, genannt. Unabhängig von ihren sonstigen Eigenschaften stellen alle genannten Schemata die für Content-Typen in Plone wichtigen Eigenschaften *id* und *title*.

Ein Schema besteht aus den *Feldern* eines Content-Typs und den zu den Feldern gehörenden *Widgets*. Ein Widget beschreibt eine Komponente der Benutzeroberfläche, wie ein Textfeld, Drop-Down-Boxen oder Buttons. Ein Schema umfasst 0 bis beliebig viele Felder, die wiederum genau ein Widget zugewiesen bekommen [McKay / da Silva (2006), S. 365 / 366].

Bei den Feldern können verschiedene *Eigenschaften* angegeben werden. So kann festlegt werden, ob man nach den Inhalten dieses Felds suchen kann (*searchable*, siehe Index *searchableText*) oder ob die Eingabe zwingend vorgeschrieben ist (*required*). Um die Gültigkeit von Benutzereingaben hierfür zu überprüfen, kann auch ein oder mehrere so genannte *Validatoren* angegeben werden. Hierfür stellt Plone einige Standardvalidatoren zur Verfügung, wie z.B. zur Syntaxüberprüfung von URLs und Email-Adressen. Weitere Validatoren können selbst definiert werden. Die Zuordnung eines Widgets erfolgt durch die Zuweisung der Eigenschaft *widget*. Innerhalb der Widget-Zuweisung können weitere Werte für das Widget gesetzt werden, die sich auf die Erscheinungsform auswirken, wie die Größe oder das Label des Widgets). Die Widget-Eigenschaft eines Felds ist deshalb so wichtig, da auf Grundlage dieser Informationen das *edit*-Template zur Bearbeitung automatisch erstellt wird.

Die Datei für mit Archetypes implementierte Content-Typen unterscheidet sich von der bereits beschriebenen Variante. Zu Beginn stehen die üblichen Import-Anweisungen zu den Klassenbibliotheken, wie auch ggf. der Verweis auf eventuell zu verwendende Validatoren. Anstatt der Factory Type Information folgt darauf in der PY-Datei die Schemadefinition. Hierin werden nun alle Felder mit ihren Eigenschaften und zugehörigen Widgets aufgelistet.

[11] Vgl. http://argouml.tigris.org/
[12] Vgl. http://plone.org/documentation/tutorial/archgenxml-getting-started/intro
[13] Es wird empfohlen, bei besonders großen Mengen von Instanzen das höher performante Grundschema baseTreeFolder zu verwenden. Vgl. u.a. [Limi (2005)].

Abbildung 19 veranschaulicht einen Ausschnitt aus der Schemadefinition des auf Archetypes basierenden *Document*-Content-Typs.

```
# zunächst wird das Schema festgelegt
# als Basis-Schema wird ATContentTypesSchema gewählt, das erweitert wird
ATDocumentSchema = ATContentTypeSchema.copy() + Schema((

# Zuweisung eines Felds vom Typ TextField für die Eingabe des Haupttexts des Dokuments
 TextField('text',
    required=True, # Festlegung, dass das Feld belegt warden muss
    searchable=True, # Festlegung, dass der Inhalt des Feld durchsuchbar ist
    primary=True,
    storage = AnnotationStorage(migrate=True),
    validators = ('isTidyHtmlWithCleanup',), # Festlegung des Validators
    default_content_type = zconf.ATDocument.default_content_type,
    default_output_type = 'text/x-html-safe',
    allowable_content_types = zconf.ATDocument.allowed_content_types,

# ab hier Festlegung des Widgets, also der Darstellung auf der Benutzeroberfläche
# die Zuweisung RichWidget legt fest, dass in dieses Feld HTML-Code eingetragen wird
    widget = RichWidget(
      description = "",
      description_msgid = "help_body_text",
      label = "Body Text", # die label-Zuweisung erscheint als Überschrift auf der
Benutzeroberflächer
      label_msgid = "label_body_text",
      rows = 25, # das Widget stellt ein 25 Zeilen großes Textfeld zur Verfügung
      i18n_domain = "plone",
      allow_file_upload = zconf.ATDocument.allow_document_upload)),
 ), marshall=RFC822Marshaller()
 )
```

Abbildung 19: Ausschnitt aus der Schema-Definition des *Document*-Content-Typs auf Archetypes-Basis

Im Anschluss an die Schema-Definition kann die *finalizeATCTSchema*-Methode aufgerufen werden, wodurch in Plone bei der Generierung der Bearbeitungsformulare Routinen zur Anzeige weiterer Felder aufgerufen werden.[14] In der darauf folgenden Klassendefinition wird auf das Schema verwiesen und weitere Informationen, die in der *Factory Type Information* zu finden sind, angegeben. Ansonsten unterscheidet sie sich nicht wesentlich von Klassendefinitionen herkömmlicher Content-Typen. Die *registerATCT*-Anweisung schließt die Content-Typ-Beschreibung mit Archetypes ab.

1.5.6 Entwicklung mit Archetypes und ArchGenXML

Ein wichtiger Bestandteil dieser Arbeit ist die Anwendung des Plone-Produkts *ArchGenXML*. Wie in Abschnitt 1.5.5 beschrieben, dient es zur automatisierten Erzeugung von auf Archetypes basierenden Content-Typen aus UML-Klassendiagrammen. ArchGenXML unterstützt zwar auch die Implementierung von Workflows aus Aktivitätsdiagrammen, diese werden hier allerdings nicht betrachtet. Dieser Abschnitt beschreibt die Erstellung von Klassendiagrammen zur späteren Umwandlung in Content-Typen auf Archetypes-Basis.

Der wichtigste Gegenstand der Modellierung ist zunächst der Content-Typ selbst. ArchGenXML verfährt so, dass jede *Klasse* eines Klassendiagramms zu einem Content-Typ umgewandelt wird. Durch die Zuweisung eines *Stereotyps* zu einer Klasse kann festgelegt werden, von welchem Archetypes-Schema der neue Content-Typ abgeleitet werden soll. Wenn kein Stereotyp gesetzt wird, so wird davon ausgegangen, dass es sich um einen einfachen Content-Typen handelt, der sich vom Schema *BaseSchema* ableitet (siehe Abschnitt 1.5.5).

Die in den Klassen definierten *Attribute* werden später auf die Eigenschaftsfelder der Content-Typen abgebildet. Hierbei kommt den so genannten Eigenschaftswerten (*tagged values*) eine besondere Bedeutung zu. Dort können die zu den Feldern gehörenden Parameter wie Sichtbarkeit (*visible*), Default-Werte oder erforderliche Eingabe (*required*) eingegeben

[14] Vgl. http://plone.org/documentation/tutorial/richdocument/extending-atct

werden. Ebenso kann dort festgelegt werden, ob ein Feld im Katalog indiziert wird oder in die Metadaten aufgenommen wird. In engem Zusammenhang mit der Wahl der Datentypen für die Attribute steht die Zuweisung der Widgets. ArchGenXML weist jedem aus den Attributen abgeleiteten Feld eines Content-Typen ein Default-Widget zu.

Aus den einer Klasse zugewiesenen *Operationen* leitet ArchGenXML Methoden für Content-Typen ab. Diese werden schließlich gemäß den angegebenen Sichtbarkeitswerten als öffentlich, geschützt oder privat ausgewiesen. Wichtig für Plone sind auch die den Methoden ähnlichen Aktionen. Diese werden durch die Zuweisung von Stereotypen zu den Operationen deklariert. So definiert der Stereotyp <<*action*>> eine frei definierbare Aktion, während <<*view*>> den Namen für die Aktion zur Anzeige von Content festlegt.

Beziehungen zwischen Content-Typen

Zudem werden Beziehungen zwischen Klassen von ArchGenXML berücksichtigt. Sind Klassen durch *Aggregationen* verbunden, sind die Komponenten einer aggregierenden Klasse als erlaubte Content-Typen zugewiesen. Die dabei als Unterelemente definierten Content-Typen sind allerdings von anderen Content-Typen einfügbar. Bei einer *Komposition* hingegen können zugewiesene Content-Typen lediglich von der komponierenden Klasse generiert werden.

Über die Verwendung von Generalisierungen können Methoden und Schemadefinitionen der generalisierenden Klassen übertragen werden. Dabei ist nicht nur einfache, sondern auch mehrfache Vererbung möglich. Sollen Spezialisierungen von Content-Typen eingetragen werden, die nicht im Modell vorhanden sind, so kann über das Setzen des Eigenschaftswerts *additional_parents* eine durch Kommas getrennte Liste von Klassen hinzugefügt werden.

Eine besonders wichtige Rolle bei der Übertragung von Klassendiagrammen zu Content-Typen in Plone spielen *gerichtete Assoziationen*. Durch dieses Modellierungselement ist es möglich, Referenzen zwischen Klassen herzustellen. Z.B. wird durch die gerichtete Assoziation von der Klasse *Organisationseinheit* zu *Person* festgelegt, dass in Plone vom Content-Typ *Organisationseinheit* auf Instanzen des Content-Typs *Person* referenziert werden kann.

Tabelle 1 stellt in einer Übersicht die bei Nutzung von ArchGenXML wichtigen Elemente eines Klassendiagramms ihren Entsprechungen in Content-Typen gegenüber (vgl. auch [Aune (2005), S. 16]).

Klassendiagramm	Auf Archetypes basierender Content-Typ
Paket	Produkt
Klasse	Content-Typ
Operation	Methode
Attribut	Feld
Eigenschaftswert (tagged value)	Eigenschaft
Klassen-Stereotyp	Sub-Schema
Operations-Stereotyp	Aktion

Tabelle 1: Entsprechungen zwischen Klassendiagramm und Archetypes

1.6 Durchführung der Migration

Im Abschnitt 1.2 dieser Arbeit wurde das Referenz-Prozessmodell für Migrationen ReMiP von Ackermann als zentraler Gegenstand dieser Arbeit vorgestellt. Es wurde dargelegt, dass innerhalb dieses Prozessmodells die Durchführung einer Migration in Phasen, Kernbereiche und Basisbereiche unterteilt ist. Zudem wurde die Migration der GXL-Website des Instituts für Softwaretechnik an der Universität Koblenz in das Zielsystem Plone als konkrete Anwendung für die Erprobung des ReMiP herausgestellt.

In Abschnitt 1.3 wurde das Zielsystem für die Migration erläutert und sowohl allgemein durch relevante Internet-Technologien als auch spezifisch durch die Präsentation des Content Management Systems Plone vermittelt. Somit sind die Grundlagen für die Durchführung der Migration gelegt.

In den folgenden Abschnitten dieses Bereichs wird die Anwendung des ReMiP für die Migration der GXL-Website dokumentiert. Zur besseren Orientierung sind die Schritte nach den Kern- und Basisbereichen des ReMiP strukturiert. Die beschriebenen Aktivitäten sind nicht sequenziell zu verstehen (vgl. Abschnitt 1.2.2). Vielmehr handelt es sich um Aktivitäten, die über das gesamte Projekt in unterschiedlicher Intensität durchgeführt wurden (siehe Abschnitt 13.1.2). Ebenso interagieren sie miteinander, so dass an manchen Stellen Verweise zu Abschnitten, die erst später in der Arbeit erscheinen, unvermeidlich sind.

In welcher Form die Kernbereiche ausgeführt wurden und wie sich die Phasen und Basisbereiche der Migration in das Gesamtbild einfügen, wird in Abschnitt 13 zur Evaluation des ReMiP beschrieben. Die Abschnitte 2 bis 12 dienen der Dokumentation der migrationsspezifischen Aktivitäten, die ansonsten durch separate Artefakte erfasst worden wären.

2 Anforderungsanalyse

Die Anforderungsanalyse leistet unter Anwendung der in Abschnitt 1.2.2 angesprochenen Aktivitäten *Problem analysieren*, *Anforderungen ermitteln*, *System definieren* und *Anforderungen verwalten* die Ermittlung, Analyse, Dokumentation, Validierung, Verabschiedung und Verwaltung der Anforderungen an das Zielsystem in einer konsistenten und überprüfbaren Form [Ackermann (2005), S. 167]. Zur Systematisierung der Anforderungen werden diese in *funktionale* und *nicht-funktionale* Anforderungen unterschieden. Funktionale Anforderungen sprechen die Funktionalität des Zielsystems an, also was das System leisten soll. Dem hingegen legen nicht-funktionale Anforderungen qualitative Aspekte fest, wie das System die Funktionalität erbringen soll.

Innerhalb einer Migration sind die funktionalen Anforderungen weitestgehend vom Legacy-System vorgegeben. Zwar kann entschieden werden, dass bestimmte Funktionalität nicht in das Zielsystem zu übernehmen ist, die bewusste Erweiterung der Funktionalität widerspricht allerdings dem Grundprinzip der Migration (vgl. Ackermann (2005), S. 167 / 168). Im Rahmen der Anforderungsanalyse einer Migration ist statt der Neudefinition funktionaler Anforderungen die Legacy-System-getreue Abbildung derselben das Ziel.

Bei der Betrachtung der nicht-funktionalen Anforderungen hingegen kann auch auf Erweiterungswünsche des Auftraggebers eingegangen werden [Ackermann (2005), S. 167]. Dabei ist vor allem die Definition der technischen Anforderungen an die Zielumgebung und das Migrationsumfeld relevant, da hier viele Vorgaben nicht vom Legacy-System bestimmt sind.

2.1 Problem analysieren

Um möglichst vollständige Informationen für die Migration zu sammeln, ist zunächst die *Identifikation der Stakeholder* von Belang [Ackermann (2005), S. 172]. Durch gemeinsame Bearbeitung der Problemanalyse soll sichergestellt werden, dass die Anforderungen und Rahmenbedingungen der Migration richtig erfasst werden. Im Fall der zu migrierenden Website wurden folgende Stakeholder identifiziert: der Website-Betreiber der GXL-Website als Auftraggeber, der Website-Verantwortliche des relevanten Fachbereichs, der Redakteur der GXL-Website, der Domänenexperte, der Legacy-Experte, die zukünftigen Betrachter der Website, der Designer und der Programmierer.

Der Betreiber der zu migrierenden Website, oder auch Auftraggeber, wird repräsentiert durch einen Beteiligten des auf der Website dargestellten Projekts, Herr Dr. Winter. Auf diesen Stakeholder wird im weiteren Verlauf der Arbeit als *Website-Betreiber* verwiesen. Ebenso vertritt Herr Dr. Winter die Stakeholder-Rolle des *Redakteurs*. Er hat in der Vergangenheit bereits Inhalte auf der Website veröffentlicht und wird auch in Zukunft neuen Content einfügen. Schließlich übernimmt er noch die Rolle des *Domänenexperten*. Im Verlauf des Website-Betriebs hat er inhaltliche Anpassungen an der Website vorgenommen und kann in Bezug auf dort abgelegte Inhalte Hinweise geben. Ebenso steht er eingeschränkt auch als *Legacy-Experte* zur Verfügung. Herr Dr. Winter war zwar bei der technischen Erstellung der Website nicht intensiv beteiligt, dennoch kennt er deren technische Grundlagen.

Der Verantwortliche der Website des relevanten Fachbereichs ist Herr Dr. Gipp. Er hat die Umstellung der Webseiten des gesamten Fachbereichs in ein Content Management System auf Basis von Zope durchgeführt, das dem Zielsystem ähnlich ist (vgl. Abschnitt 1.5). Damit kann er bei der Migration technische Hinweise geben und auf seine Erfahrungen bei der Verwaltung der Website des Fachbereichs zurückgreifen. Auf diesen Stakeholder wird mit *Website-Verantwortlicher* verwiesen.

Die Zielperson einer Website ist ihr Betrachter. Hierfür kommen natürlich sehr viele Personen in Frage, die diese Position einnehmen könnten, aber nicht alle in das Migrationsvorhaben eingebunden werden können. In diesem Fall wird der Verfasser der Arbeit die Perspektive des Website-Betrachters übernehmen. Auf diesen Stakeholder wird im Folgenden als *Website-Betrachter* verwiesen.

Schließlich verbleiben noch die Stakeholder Designer und Programmierer. Diese Rollen werden vom Verfasser der Arbeit ausgefüllt, der naturgemäß diese Stakeholder repräsentiert. Auf sie wird im weiteren Verlauf mit *Website-Designer* bzw. *Website-Programmierer* verwiesen.

Durch die Erstellung einer *Vision* werden in Zusammenarbeit mit den Stakeholdern des Migrationsprojekts die zentralen Problemstellungen wie Ziele, Anforderungen und Randbedingungen auf recht abstraktem Niveau beschrieben [Ackermann (2005), S. 172]. Im vorliegenden Fall dient der folgende Absatz zur Formulierung der Vision:

> Zweck dieses Projekts ist die Migration der GXL-Website in das Content Management System Plone. Dabei sollen alle relevanten Daten, die auf der GXL-Website gehalten werden, in einer ähnlichen Form in das Zielsystem übertragen werden. Ziel dieser Migration ist es auch, die Verwaltung der Website entscheidend zu vereinfachen. Diese Vereinfachung umfasst sowohl die Verwaltung von Daten als auch die Schaffung von Grundlagen zur Weiterentwicklung des Designs. Am Ende der Migration soll die Website des neuen Zielsystems über das Internet verfügbar sein.

Um eine unmissverständliche Kommunikation zwischen den Projektbeteiligten zu gewährleisten, ist die Festlegung einer *einheitlichen Terminologie* von Bedeutung. Diese

Terminologie wird in einem Glossar formuliert und während des Projektverlaufs ständig aktualisiert [Ackermann (2005), S. 172 / 173]. Im Rahmen der vorliegenden Arbeit zählen zu dieser Terminologie auch Begriffe wie „zu migrierende Website“ oder „Legacy-Website“, die bereits öfter verwendet wurden. Da eine Aufführung aller Begriffe an dieser Stelle teilweise zu weit auf spätere Migrationsphasen voraus greifen würde, befindet sich im Anhang dieser Arbeit ein Glossar, dass die wichtigsten Begriffe definiert.

2.2 Anforderungen ermitteln

In der vorliegenden Arbeit wurden die Anforderungen iterativ erhoben. In Abständen von zwei bis drei Wochen wurden Treffen der Vertreter aller Stakeholder anberaumt, um den Fortschritt der Migration zu erläutern. Im Rahmen dieser Besprechungen wurden auch die Bedürfnisse der einzelnen Stakeholder diskutiert und durch Rückfragen verifiziert. Somit konnte im Verlauf der Zeit die Anzahl der Anforderungen vervollständigt werden.

2.2.1 Funktionale Anforderungen

Die funktionalen Anforderungen an das Zielsystem leiten sich, dem Grundprinzip der Migration folgend, aus der Funktionalität des Legacy-Systems ab [Ackermann (2005), S. 39]. Die genaue Kenntnis dieser Funktionalität leitet sich vor allem aus der Legacy-Analyse ab. Zum Nachvollzug der funktionalen Anforderungen sei damit auf Abschnitt 3 verwiesen.

2.2.2 Nichtfunktionale Anforderungen

Im Gegensatz zu den funktionalen Anforderungen, die durch das Legacy-System vorgegeben sind, können im Rahmen einer Migration die nichtfunktionalen Anforderungen recht frei erarbeitet werden. Die Grenzen bei der Erhebung dieser Anforderungen sind nur durch die organisatorischen Rahmenbedingungen des Migrationsprojekts gegeben. Für die zu migrierende Website ergaben sich nach mehreren Gesprächen mit dem Website-Betreiber verschiedene Anforderungen.

Integration der Literatur und der Werkzeuginformationen: Die zu migrierende Website enthält u.a. einen wissenschaftlichen Text mit Literaturverweisen, sowie eine Publikationsliste zum Thema GXL. Es besteht Interesse daran, die Anzeige der Literaturverweise zu vereinheitlichen und möglichst in einer Literaturdatenbank dynamisch abrufbar und verwaltbar zu machen. Hierfür bietet sich insbesondere die Integration mit der Literaturdatenbank *LitDB* des Fachbereichs 4 der Universität Koblenz-Landau an.[15] (Bedürfnis von Seiten des Website-Betreibers).

Layout: Da in Zukunft der Internetauftritt universitätsweit integriert werden soll, ist die Einbindung der GXL-Website ebenfalls von Belang. Hierbei sollte die zu migrierende Website sich in das Erscheinungsbild der bereits bestehenden Website des Fachbereichs 4 der Universität einfügen (Bedürfnis von Seiten des Website-Verantwortlichen der Universität). Aber die neue GXL-Website soll charakteristische eigene Merkmale besitzen, die die Beschreibung der Inhalte der zu migrierenden Website unterstützen (Bedürfnis von Seiten des Website-Betreibers). Dabei soll in Rücksicht auf den damit verbundenen Aufwand zunächst nur ein Basis-Design erarbeitet werden. Die Weiterentwicklung des Layouts wird als ein von der Migration ausgelagertes Projekt definiert.

Wartbarkeit: Nach der Migration der Website sollen die Inhalte und die realisierte Funktionalität für einen möglichst großen Kreis von Entwicklern wartbar sein. Wenn möglich soll auf leicht erlernbare, ggf. als Standardprodukt vorhandene Software zurückgegriffen

[15] Siehe unter https://www.uni-koblenz.de/~litdb/secure/index.php.

werden. Es soll vermieden werden, dass ein Programmierer Software entwickelt, die später von anderen Entwicklern nur schwer nachvollziehbar und veränderbar sein wird. Hierfür muss eine hinreichend detaillierte Dokumentation des Zielsystems existieren, die auch durch visuelle Modellierung unterstützt wird (Bedürfnis von Seiten des Website-Betreibers).

Support: Auf Grundlage des Bedürfnisses, möglichst ein Standardprodukt zur Website-Verwaltung zu verwenden, ist es wichtig, dass für das installierte System über einen längeren Zeitraum Support in irgendeiner Weise geleistet werden kann. Dieser muss nicht direkt erfolgen, allerdings sollte ausreichend Dokumentation vorhanden sein. Ebenso sind Internetforen, die sich mit der Entwicklung in dieser Zielumgebung beschäftigen, für die Lösung zukünftiger Probleme nützlich (Bedürfnis von Seiten des Website-Betreibers).

Anpassbarkeit: In Zukunft werden Änderungen in der Gestaltung und Funktionalität der Website nicht ausgeschlossen. Deshalb soll es möglich sein, Elemente der Website einfach zu bearbeiten und zu erweitern. Das umfasst nicht nur das Hinzufügen von Content, sondern auch die Veränderbarkeit der Content-Strukturen (Bedürfnis von Seiten des Website-Betreibers).

Software-Evolution: Für gewöhnlich wird Software einmal installiert und schließlich vor Ort genutzt und im Betrieb angepasst. Im Rahmen der Software-Wartung gibt es aber mit der Zeit immer wieder notwendige Änderungen an der eigentlichen Software, die verschiedener Art sein können.[16] Diese werden für die in Betrieb befindliche Software mittels Updates übertragen. Mit zunehmender Zeit werden auch umfangreichere Anpassungen notwendig, die schließlich zu einer Überarbeitung der Software in Form einer neuen Version führen können. Für diese Entwicklungstätigkeiten ist eine Gruppe von Software-Entwicklern notwendig, die für längere Zeit das System anpassen können. Die Entwicklungsarbeit soll nicht oder wenn nur in einem geringen Maße von den Website-Betreibern geleistet werden (Bedürfnis von Seiten des Website-Betreibers)

Verbesserung der Website-Qualität: Die Website wurde zwecks schneller Publikation seinerzeit eher durch „Code and Fix“ erstellt. Dadurch ist es zu kleineren Fehlern im Quellcode der HTML-Dateien gekommen. Diese sollen vor Durchführung der Migration im Rahmen einer Sanierungsmaßnahme korrigiert werden (Bedürfnis von Seiten des Website-Betreibers).

Lauffähigkeit auf Browsern: Die zu migrierende Website soll nicht ausschließlich auf einen Browser ausgerichtet sein. Stattdessen soll sie auf möglichst vielen Browsern fehlerfrei angezeigt werden. Damit wird der Verbreitung vieler verschiedener Browser-Produkte im Internet Rechnung getragen werden (Bedürfnis des Website-Betreibers).

Der Vergleich aller im Umlauf befindlichen Browser ist nicht praktikabel. Deswegen wird eine Beschränkung der untersuchten Browser vorgenommen. Um dem größten Anteil der verwendeten Browser abzudecken, wird die Lauffähigkeit bei *Internet Explorer, Firefox, Mozilla und Opera* getestet. Gemäß öffentlich verfügbarer Statistiken deckt der Internet Explorer etwa 60 %, Mozilla Firefox knapp 25 %, Mozilla 2 % und Opera 1,5 % der Internet-Nutzer weltweit ab.[17] Somit wird gewährleistet, dass die Website für knapp 90% der potenziellen Nutzer lauffähig ist.

Einhaltung der zeitlichen und budgetbedingten Restriktionen: Die Durchführung der Migration wird im Rahmen der Erstellung einer Abschlussarbeit geliefert. Der Verfasser

[16] Gemeint sind die verschiedenen Arten von Softwarewartung: adaptiv, korrektiv, präventiv und perfektiv. Vgl. z.B. [Masak (2005), S. 160 ff.].
[17] Vgl. hierfür u.a. http://www.browser-statistik.de/ und http://www.w3schools.com/browsers/browsers_stats.asp.

dieser Arbeit unterliegt dabei vor allem zeitlichen Restriktionen. Zur Anfertigung der Arbeit stehen ihm nach Vorgabe der Prüfungsordnung sechs Monate Zeit zur Verfügung. Diese zeitliche Vorgabe soll eingehalten werden (Bedürfnis des Website-Programmierer und -Designers). Zudem existieren für die Durchführung der Migration keine unbegrenzten finanziellen Mittel. Einerseits soll der Verfasser der Arbeit sich finanziell nicht verausgaben (Bedürfnis des Website-Programmierer und –Designers). Andererseits sollen von Seiten des Website-Betreiber und des Website-Verantwortlichen keine übermäßigen Aufwändungen für die Universität zur Realisierung der Migration entstehen

2.3 Anforderungen verwalten

Die Anforderungsverwaltung umfasst auch die Festlegung der Konventionen, wie Anforderungen zu erfassen sind [Ackermann (2005), S. 176]. In den in Abschnitt 2.2 ermittelten Anforderungen ist eine solche Systematik erkennbar. Zunächst wurden die unterschiedlichen Anforderungen an das Zielsystem einzeln aufgelistet. Innerhalb der Beschreibung der Anforderungen finden sich detaillierende Angaben zur Art der Anforderung, der Ausprägung der Anforderung und zu der Person, von der die Anforderung ausgeht. Somit ist es gewährleistet nachzuvollziehen, was eine Anforderung umfasst und erleichtert bei Abschluss der Migration die Überprüfung der Erfüllung der Anforderungen.

Im Fall der zu migrierenden Website ist die Menge der Anforderungen noch überschaubar. Deshalb wurde auf ein aufwändiges Anforderungsmanagementsystem verzichtet. Durch die ständige Aktualisierung und Erweiterung der im Abschnitt 2.2 erstellten Liste ist für jeden Stakeholder dennoch ersichtlich, wie sich die Anforderungen zueinander verhalten.

3 Legacy-Analyse / Aufbereitung

Im Kernbereich „Legacy-Analyse / Aufbereitung“ wird der genaue Aufbau der Legacy-Website untersucht und festgestellt, ob und in welchem Umfang Sanierungsbedarf bei den Webseiten besteht. Hier entscheidet sich auch, ob das Projekt überhaupt durchzuführen ist und welche Maßnahmen getroffen werden müssen, um die Migration zu ermöglichen.

3.1 Design & Verhalten des Legacy-Systems grob rekonstruieren

Durch die Aktivitätsgruppe „Design & Verhalten des Legacy-Systems grob rekonstruieren“ wird das Legacy-System zunächst oberflächlich beschrieben, um daraus das weitere Vorgehen bei der Untersuchung planen zu können [Ackermann (2005), S. 182]. Im Fall der zu migrierenden Website werden genauer ihre die technologischen Grundlagen, die Navigation und die auf der Website gehaltenen Daten untersucht.

Der eigentliche *Hauptbereich* der Website beinhaltet eine Leiste zur Navigation und stellt innerhalb seiner Struktur die meisten Inhalte der Website dar. Der *Beispielbereich* kann innerhalb des Navigationsbereichs ausgewählt werden. Wird dieser Link betätigt, so öffnet sich ein neues Fenster zur Darstellung von Beispielen des Themenbereichs GXL. Zwar ist das Erscheinungsbild des Beispielbereichs dem Hauptbereich sehr ähnlich, allerdings verfügt dieser über eine eigene Navigation und ist strukturell anders unterteilt.

Ebenfalls über die Navigation des Hauptbereichs kann der Download-Bereich angesteuert werden. Markant ist an dieser Stelle, dass der Download-Bereich nicht zur GXL-Website selbst gehört, sondern sich auf einer anderen befindet, die sich mit dem Projekt GUPRO der Universität Koblenz-Landau beschäftigt. Die GUPRO- und GXL-Websites stehen miteinander in direktem Zusammenhang, da GXL ein Unterprojekt von GUPRO darstellt. Im

Download-Bereich Software-Downloads allgemein zum Projekt GUPRO und dabei auch zum Themenbereich GXL verfügbar gemacht und verwaltet.

3.1.1 Technologische Grundlagen der Website

Die zu migrierende Website verzichtet weitestgehend auf Programmierung zur Anzeige der Webseiten und der Interaktion mit dem Benutzer. Dem Betrachter der Website erscheint sie als großes Dokument, in dem man über Nutzung von Hyperlinks bequem navigieren kann. Die Vermittlung von Information in Form von einführenden Texten und listenartigen Übersichten steht im Vordergrund.

Bei der Legacy-Website handelt es sich um eine überwiegend statische Website. Alle in diesem System veröffentlichten Seiten liegen in abgespeicherter Form auf dem Server zum Abrufen bereit. Zum Zeitpunkt der Anfrage beim Server werden keine Skripte ausgeführt, welche eine Interaktion mit dem Client ermöglichen. Es existiert lediglich ein Verweis auf einen Bereich der übergeordneten GUPRO-Website für die Verfügbarmachung von *Downloads*. Dieser Bereich ist mit dem Web-Applikationsserver Zope realisiert worden und erzeugt aus Templates bei jedem Aufruf die angezeigte Website. Da die Verwaltung dieser Downloads sich nicht direkt im Bereich der zu migrierenden Website befindet, wird sie *nicht in die Migration mit aufgenommen*.

Obwohl die Website statisch ist, kann für ihre Bearbeitung auf ein ausgeklügeltes System zur Verwaltung ihrer Inhalte zurückgegriffen werden. Über die Nutzung des Programms *make* können die einzelnen Webseiten nach Bedarf über festgeschriebene Routinen generiert werden. Diese Befehle, die in einem so genannten *Makefile* notiert sind, werden außerdem zur Erzeugung eines einheitlichen Website-Layouts und zur Generierung von Webseiten auf Grundlage von XML-Dokumenten verwendet. Die genaue Vorgehensweise wird in Abschnitt 3.3 erläutert. Festzuhalten ist an dieser Stelle, dass die Veränderung einiger Webseiten und des Layouts der gesamten Website von zentraler Stelle unabhängig von der später angezeigten Webseite durchgeführt werden kann.

3.1.2 Navigation und Ordnerstruktur

Durch die statische Struktur der GXL-Website ist in Hinblick auf die Navigation die Betrachtung der Ordnerstruktur im Dateisystem sinnvoll. Tatsächlich lässt sie sich sehr gut aus der Navigationsleiste ablesen. Dokumente für die einzelnen Unterthemen wurden fast ausnahmslos in separate Ordner gespeichert. So lässt sich die physische Ordnerstruktur der Website mit der Navigationsleiste abbilden. Oberbegriffe innerhalb der Navigation werden als *Navigationsbereich*, während die in ihnen enthaltenen Unterthemen als *Navigationspunkte* bezeichnet werden.

Navigationsbereich *Overview*

- Navigationspunkt *Background*: Ordner *Introduction*
- Navigationspunkt *Introduction*: Ordner *Introduction*
- Navigationspunkt *FAQ*: Ordner *faq*
- Navigationspunkt *Examples*: Ordner *examples*
- Navigationspunkt *Publications*: Ordner *Publications*

Navigationsbereich *Definition*

- Navigationspunkt *DTD*: Ordner *dtd*
- Navigationspunkt *XML Schema*: Ordner *xmlschema*

Navigationsbereich *Schemas*:

- Navigationspunkt *Graph Model*: Ordner *GraphModel*

- Navigationspunkt *Metaschema*: Ordner *MetaSchema*

Navigationsbereich *Tools*:
- Navigationspunkt *Tool Catalogue*: Ordner *tools*
- Navigationspunkt *Downloads*: Hyperlink zu *Download*-Bereich → eigener Teilbereich der Website

Navigationsbereich *Advance*:
- Navigationspunkt *Change Requests*: Ordner *changeRequest*
- Navigationspunkt Future: Ordner *future*
- Navigationspunkt *GXL 1.1*: Ordner *dtd*

Wie in dieser Gegenüberstellung zu erkennen ist, sind für fast alle Navigationspunkte eigene Ordner auf dem Server erzeugt worden. Lediglich bei *Background* und *Introduction* sowie *DTD* und *GXL 1.1* teilen die jeweiligen Navigationspunkte sich denselben Ordner.

3.1.3 Daten der Legacy-Website

Im Rahmen der Legacy-Analyse auf abstrakter Ebene ist auch die erste Erfassung der im Alt-System gehaltenen Daten von Bedeutung. Zwar erfolgt an dieser Stelle keine ausführliche Modellierung, dennoch stellt die Identifizierung der Inhalte auf den Webseiten eine Grundlage für die spätere Detailuntersuchung dar. Die auf den einzelnen Webseiten vorgefundenen Inhalte auf einer etwas abstrakteren Ebene werden nachfolgend aufgelistet:
- Webseite *Background*: Projekt, Tagung, Organisation, Organisationseinheit, Person
- Webseite *Introduction*: Artikel, Bild
- Webseite *FAQ*: FAQ
- Webseite *Links*: Link
- Webseite *Examples*: Beispiel, Bild, Datei
- Webseiten *Definition*: Quelltext, Datei
- Webseiten *Schemas*: Seite, Bild
- Webseite *Tool Catalogue*: Werkzeug, Organisation, Organisationseinheit, Person
- Webseite *Change Requests*: Änderungsanfrage, Person, Tagung, Organisation, Organisationseinheit
- Webseite *Future*: Seite
- Webseite *GXL 1.1*: Quelltext, Datei
- Webseite *Change Log*: Änderung

Die im Kontext von Webseiten nächstliegenden Inhalte sind *Seite*, *Bild* und *Datei*. Diese drei Inhalte beschreiben die wesentlichen Elemente, die eine Webseite ausmachen. Zum einen beschreibt *Seite* allgemein den HTML-Code einer regulären Webseite, der Referenzen zu anderen im Internet verfügbaren Dokumenten und Dateien macht. Wenn es sich um Referenzen zu Bildern handelt, greift *Seite* auf eine Instanz des Typs *Bild* zurück. *Bild* fasst Dateien zusammen, die als Bilder gespeichert wurden und ein entsprechendes Format aufweisen. Hinzu kommt noch *Datei*, wodurch im Internet verfügbare Dateien beschrieben werden. Diese lassen sich nicht wie Bilder in ein HTML-Dokument einbinden und werden vor allem zum Download angeboten.

Die Inhalte der Webseite *Background* umfassen die Beschreibung eines *Projekts*. Die Projektbeschreibung verfügt u.a. über eine kurze Beschreibung und Kontaktinformationen. Das Projekt wurde bei *Tagungen* oder Präsentationen diskutiert, die auf der Webseite aufgelistet sind. Zu der Projektarbeit bestehen Beziehungen zu *Organisationen* und *Organisationseinheiten*, die das Projekt in partnerschaftlicher Zusammenarbeit unterstützen. Diese Organisationen sind vor allem Unternehmen und Universitäten, während

Organisationseinheiten je nach Organisation Abteilungen, Institute oder Arbeitsgruppen sind. Zuletzt befinden sich auf dieser Webseite auch Informationen zu Einzelpersonen. Diese *Personen* sind am Projekt direkt als Teilnehmer oder indirekt über Organisationseinheiten beteiligt.

Unter dem Navigationspunkt *Introduction* befindet sich ein *Artikel*, der eine einführende Beschreibung zum Thema GXL leistet. Der Artikel besteht aus mehreren Webseiten und weist Eigenschaften auf, die über die Beschreibung einer regulären Seite hinausgehen. Zwischen den einzelnen Webseiten existieren nämlich Navigationslinks, die sie als ein eigenständiges Dokument kennzeichnen. Innerhalb der Webseiten des Artikels werden auch *Bilder* verwendet.

Der Navigationspunkt *FAQ* fasst die beliebten Frequently Asked Questions (*FAQ*) zusammen. Bei der vorliegenden Webseite handelt es sich um wiederkehrende Fragen rund um den Themenbereich GXL.

Unterhalb des Navigationspunkts *Examples* befindet sich umfangreiches Material mit *Beispielen* zur Verdeutlichung der Funktionsweise von GXL. Diese Beispiele sind thematisch geordnet und verfügen neben GXL-Quelltextbeispielen auch über *Bilder* von Graphen und UML-Klassendiagrammen. Zusätzlich werden unter diesem Navigationspunkt auch *Dateien* als komprimierte Ordner angeboten, die eine Sammlung aller Beispiele enthalten.

Der Navigationsbereich *Definition* umfasst insbesondere *Quelltext* von DTDs und XML-Schemata. Zu jedem Quelltext werden eine Copyright-Information sowie eine kommentierte Quelltextversion bereitgestellt. Da diese Quelltexte von anderen Webseiten referenziert werden (z.B. bei Document Type Definitions), wird zusätzlich noch eine *Datei* angeboten, die unkommentierten Quelltext enthält.

Die Webseiten unter dem Navigationsbereich *Schema* sind relativ einfach gehalten. Die *Seiten* enthalten vor allem *Bilder*, die den Aufbau von GXL beschreiben.

Im Navigationspunkt *Tool Catalogue* befinden sich zahlreiche Informationen und eine Übersicht zu Software-*Werkzeugen*, die dem GXL-Projekt nahe stehen. Die Werkzeuge sind bezüglich ihrer Art thematisch geordnet. Bei der Werkzeugsbeschreibung existieren weiterhin auch Verweise auf *Organisationen*, *Organisationseinheiten* und *Personen*.

Innerhalb des Navigationsbereiches *Advance* werden unter *Change Requests Änderungsanfragen* zur Version GXL 1.1 veröffentlicht. Innerhalb der Änderungsanfragen gibt es auch Verweise auf *Organisationen*, *Organisationseinheiten*, *Tagungen* und *Personen*.

Die Webseite *Future* beinhaltet einfachen HTML-Text und ist somit eine *Seite*. Der letzte Navigationspunkt der zu migrierenden Webseite, *GXL 1.1*, präsentiert die Definition von GXL 1.1. Hier handelt es sich, ähnlich zum Navigationsbereich *Definition*, um *Quelltext* mit Copyright-Informationen, kommentierten Quelltext und durch *Dateien* verfügbar gemachten unkommentierten Quelltext

Von jeder Webseite des Hauptbereichs kann die Webseite *Change Log* aufgerufen werden, die eine Übersicht der durchgeführten *Änderungen* auf der GXL-Website seit der Veröffentlichung gibt.

3.2 Legacy-System global bewerten

Die globale Bewertung des Legacy-Systems dient als Ausgangspunkt zur Wahl einer Migrationsstrategie [Ackermann (2005), S. 184]. Hierfür wird die Verwendung einer Portfolio-Analyse angeraten, die die Kriterien der technischen Qualität und der betriebswirtschaftlichen Bedeutung des Alt-Systems untersucht. Zunächst werden die

Bewertungsstruktur und die Ausprägungen der Kriterien detailliert, bevor nach der Durchführung der Bewertung und dem Einbezug organisatorischer Rahmenbedingungen eine Empfehlung für die Migrationsstrategie gegeben werden kann.

3.2.1 Erstellen der Bewertungsstruktur

Zur Erstellung der Bewertungsstruktur müssen zunächst die Bewertungskriterien zu den Sammelkriterien der technischen Qualität und der betriebswirtschaftlichen Bedeutung zusammengetragen werden. Hierzu gehört neben der Definition des Untersuchungsgegenstandes auch eine Beschreibung des Bewertungsvorganges und der Übertragung der Messergebnisse in ein Bewertungssystem.

Um die einzelnen Unterkriterien allgemein vergleichbar zu machen, wird eine ordinale Bewertungsskala von 1 (sehr gut) über 2 (gut), 3 (befriedigend), 4 (ausreichend) bis 5 (schlecht) eingeführt. Jedes Kriterium muss innerhalb dieser Bewertungsskala bewertet werden. Da für das Software-System nicht alle Kriterien die gleiche Wichtigkeit besitzen, ist es sinnvoll, jedes einzelne Kriterium zu gewichten. Diese Gewichtungen wirken sich nur innerhalb eines Kriteriumssystems aus, also werden technische Kriterien nur in Verhältnis zu technischen Kriterien gewichtet, analog wird mit den betriebswirtschaftlichen Kriterien verfahren. Die Sammelkriterien untereinander werden jeweils gleich gewichtet. Es wird also angenommen, dass beide gleich wichtig sind.

3.2.2 Technische Qualität

In dem Sammelkriterium für technische Qualität werden die Kernbereiche von Software-Design betrachtet, nämlich die Programmierung, die Benutzungsschnittstellen und die Daten. Ausgehend von diesen elementaren Punkten kann eine umfassende Beurteilung der technischen Qualität erfolgen.

Programmierung
Bezüglich der Programmierung können die Aspekte der *Programmierung mit make* (über das *Makefile*) sowie die *Standardkonformität* des auf den Webseiten befindlichen HTML- und XML-Codes isoliert werden. Es soll geklärt werden, wie gut die *Makefiles* ihre Aufgabe der Aktualisierung der Webseiten erfüllen (Effektivität) und wie vorteilhaft sie dabei programmiert sind (Effizienz). Auf der anderen Seite sollen bei Prüfung der Standardkonformität des HTML- und XML-Codes der Webseiten die Abweichungen als Fehlerrate erfasst werden.

Als Bewertungsmethode bezüglich der Programmierung mit *make* wird eine Expertenmeinung verwendet. Hierbei kann auf den Website-Betreiber und seine Erfahrungen mit der Aktualisierung von Inhalten zurückgegriffen werden. Die Verwendung von *make* ermöglicht die zentrale Generierung von statischen Websites, wobei alle Abhängigkeiten zwischen den Webseiten vollständig in den *Makefiles* erfasst sind. Als Ergebnis wird die *Programmierung mit make* mit *gut* bewertet und erhält eine 2 auf der Bewertungsskala.

Zur Bewertung der HTML-Konformität bietet sich die Nutzung eines Validators für HTML und XML an, der Fehler innerhalb des Quellcodes zählt und dokumentiert. Zur Bewertung kann die Fehlerzahl und die Art des Fehlers herangezogen werden. Eine genaue Auswertung der Standardkonformität ist weiter unten in diesem Abschnitt aufgeführt. Bei der Überprüfung der HTML-Konformität wurden zahlreiche Fehler entdeckt. Dennoch sind diese nicht schwerwiegend und deuten auf systematische Fehler hin, die relativ leicht zu beheben sind. Zugleich sind die XML-Dokumente vollkommen fehlerfrei. Deshalb wird die *Standardkonformität* mit einem Wert von 2 als *gut* bewertet.

Benutzungsschnittstellen
Die Benutzungsschnittstellen einer Website sind vor allen die dort angezeigten Webseiten. Die Qualität einer Benutzungsschnittstelle wird deshalb an den Attributen einer Webseite ausgerichtet. Einerseits muss untersucht werden, wie gut und intuitiv die *Navigation* der Website realisiert wurde. Andererseits muss berücksichtigt werden, wie das *Design* auf den Benutzer wirkt und ihn bei der Suche nach Informationen unterstützt. Eine umfassende Untersuchung der Usability in ihrer Tiefe erfolgt allerdings nicht.

Zur Bewertung der Aspekte Navigation und Design ist es schwierig quantifizierbare Metriken zu finden. Stattdessen soll die Bewertung durch die subjektive Einschätzung bei der Betrachtung der Seite erfolgen. Diese Aufgabe wird von dem Verfasser dieser Arbeit übernommen.

Die Navigation ist insgesamt sehr übersichtlich und ermöglicht durch die Einbindung einer statischen Navigationsleiste eine gute Orientierung. So ist gewährleistet, dass man, mit Ausnahme des Navigationsbereichs *Examples*, auf jeder Webseite die anderen Webseiten ansteuern kann. Negativ auf die Navigation wirkt sich das Öffnen eines neuen Browser-Fensters für den angesprochenen Teilbereich sowie die Verwendung von Frames im *Examples*-Bereich (siehe Abschnitt 1.3.1 zu Internet-Technologien) aus. Deshalb wird die *Navigation* dennoch nur mit *zufrieden stellend* bewertet.

Das Design ist stilistisch recht einfach gehalten, erfüllt aber dennoch seinen Zweck. Durch die klare Strukturierung jeder Webseite in einer Tabellenstruktur findet sich der Betrachter leicht zurecht. Programmcode-Beispiele sind farblich hervorgehoben und Abschnitte innerhalb von Webseiten durch Farbwahl gut zu unterscheiden. Für das *Design* erhält die alte GXL-Website deshalb die Bewertung *gut*.

Daten
Bei Betrachtung der Daten einer Website werden zwei Aspekte getrennt. Zunächst geht es um die *Datenhaltung* innerhalb der Website. Im einfachsten Fall befinden sich die Daten direkt im HTML-Text der Webseiten. Sie könnten aber auch von externen Datenquellen wie Datenbanken stammen. Außerdem, und dabei handelt es sich bei Webseiten um einen sehr wichtigen Punkt, muss die *Aktualität der Inhalte* geprüft werden. Es wäre sinnlos eine Website zu migrieren, deren Inhalte bereits obsolet sind und keinen weiteren informativen Wert mit sich tragen.

Bei der Datenhaltung soll untersucht werden, wie viele der auf der Website veröffentlichten Daten entweder nur auf den Webseiten vorkommen, aus einer Datenbank oder sonstigem Dokument unabhängig von der Präsentation entnommen werden. Hierfür genügt eine genauere Untersuchung der Inhalte der Webseiten. Besteht eine Webseite überwiegend aus Daten, die nur im HTML-Code vorkommen, so wirkt sich das negativ auf die Qualität der Datenhaltung aus, während eine Webseite, die überwiegend aus generierten Daten besteht für eine gute Datenhaltung spricht. Diese Untersuchung wird von dem Verfasser der Arbeit durchgeführt.

Die Datenhaltung der zu migrierenden Website ist zweigeteilt. Es existieren Webseiten, die lediglich als HTML-Code vorliegen. Außerdem gibt es auch Bereiche, bei denen die Webseiten aus XML-Code generiert wurden. Im Hauptbereich der Website befinden sich acht Navigationspunkte, die nur in HTML-Format gespeichert sind. Dem stehen vier Webseiten gegenüber, deren Daten ursprünglich aus einem XML-Dokument stammen. Bei diesen vier handelt es sich im Vergleich zu den anderen Webseiten um eher lange Dokumente. Deswegen werden die beiden Gruppen als gleich gewichtet angesehen. Die *Datenhaltung* wird mit *zufrieden stellend bewertet*.

Die Beurteilung der Aktualität der Inhalte ist Aufgabe des Domänenexperten des Legacy-Systems. Er kann mit seiner Expertenmeinung am besten bestimmen, ob die Inhalte der GXL-Website weiterhin Gültigkeit besitzen. Tatsächlich sind die meisten Daten auf der Website nach wie vor aktuell. Eine Ausnahme bildet der Navigationsbereich *Advance*. Die dort gespeicherten Inhalte mit Änderungsanfragen zu GXL 1.0 und der Weiterentwicklung der Version GXL 1.1 werden als nicht mehr aktuell eingestuft. In einigen Bereichen wurde Sanierungsbedarf festgestellt, insbesondere bei den Hyperlinks zu anderen Seiten und dem Einführungsartikel zum Thema GXL. Insgesamt wird die *Aktualität des Inhalts* dennoch mit *gut* bewertet.

Bei der Gewichtung der einzelnen Kriterien muss auf deren Bedeutung für das Migrationsvorhaben Rücksicht genommen werden. Würde man jedes Kriterium gleich stark gewichten, so hätte jedes einen Faktor von über 0,16. Für die Übertragung von Webseiten ist insbesondere die Strukturierung der Daten, also die Datenhaltung, von Bedeutung. Je besser die Daten strukturiert sind, desto leichter lassen sich diese in die Zielumgebung migrieren. Gleichzeitig ist eine Migration dieser Daten nur dann gerechtfertigt, wenn die Aktualität der Daten gewährleistet ist. Dementsprechend werden für beide Kriterien mit dem Faktor 0,25 gewichtet.

Ebenfalls von Bedeutung sind die Benutzungsschnittstellen. Für die Programmierung von Transformationsroutinen in der Migration kann es erforderlich sein, dass man sich an vorhandenen Links auf den Webseiten orientiert, um Inhalte anderer Webseiten auszulesen. Hierfür ist es notwendig, dass die Navigation auf den einzelnen Seiten lückenlos vorhanden ist. Weniger bedeutend für das Migrationsvorhaben ist das Design, da die genaue Wiedergabe des Designs auf der Ziel-Website nicht gewünscht ist. Es wird die Navigation mit dem Faktor 0,2 und das Design mit 0,1 gewichtet.

Weiterhin kommen noch die Aspekte der Programmierung hinzu. Allerdings macht die Programmierung auf der Legacy-Website nur einen geringen Teil aus. Zudem muss die Struktur von *Makefiles* in der Form auch gar nicht in das Zielsystem übernommen werden. Viel wichtiger hingegen ist die Untersuchung der Standardkonformität des vorhandenen Quellcodes für HTML und XML. Wenn Webseiten in das Zielsystem übertragen werden, dann sollte eine ausreichende Quellcode-Qualität gewährleistet sein. So wird die Standardkonformität mit dem Faktor 0,15 und das Kriterium der eigentlichen Programmierung nur mit 0,05 gewichtet.

Werden alle Bewertungen nun entsprechend gewichtet erhält man das Ergebnis 2,45.[18] Damit kann die technische Qualität des Legacy-Systems leicht besser als zufrieden stellend eingeschätzt werden. Damit ist zu erwarten, dass der technische Teil der Migration zwar einige Herausforderungen mit sich bringen wird (vor allem in Hinsicht auf die Migration der Daten). Dennoch ist die technische Qualität noch so gut, dass das Risiko kontrollierbar ist.

3.2.3 HTML- und XML-Konformität der Website

Nach den Angaben des Website-Betreibers ist zu vermuten, dass die Webseiten der zu migrierenden Website fehlerhaften HTML-Code enthalten und somit Prüfungen auf Einhaltung geltender HTML-Standards nicht standhalten wird. Neben Webseiten, die lediglich in HTML geschrieben sind, existieren auch solche, die aus XML-Dateien generiert wurden. Auf Grund der strukturierten Inhalte von XML-Dokumenten ist davon auszugehen, dass die Migration der Daten einfacher mit den XML-Daten durchzuführen ist. Deshalb muss

[18] Rechnung: (2 * 0,05 + 2 * 0,15) + (3 * 0,25 + 2 * 0,25) + (3 * 0,2 + 2 * 0,1) = 2,45;
Die Klammerung wurde zur Gruppierung der Bereiche Programmierung, Benutzungsschnittstellen und Daten vorgenommen.

sowohl eine Validierung der Standardkonformität der HTML- als auch separat der XML-Dokumente erfolgen.

HTML-Konformität
Um dem Umfang der Abweichung der Website von geltenden HTML-Standards zu bestimmen, wurden einzelne Webseiten unter Nutzung des Validierungsdienstes des World Wide Web-Konsortiums auf Validität überprüft.[19] Von mehreren ähnlichen Seiten wurde nur eine exemplarisch überprüft und schließlich wurden die Fehler in Bezug auf die Anzahl der gleichförmigen Webseiten hochgerechnet. Bei diesen Seiten findet sich ein „ja“ in der Spalte „Expl.“. Tabelle 2 zeigt das quantitative Ergebnis der Untersuchung.

#	Webseite	Fehler	Expl.	Bemerkung
1	http://www.gupro.de/GXL	1		
2	http://www.gupro.de/GXL/Introduction/background.html	40		
3	http://www.gupro.de/GXL/Introduction/intro.html	60		
4	http://www.gupro.de/GXL/Introduction/section1.html	40		
5	http://www.gupro.de/GXL/Introduction/section2.html	145		
6	http://www.gupro.de/GXL/Introduction/section3.html	100		
7	http://www.gupro.de/GXL/Introduction/section4.html	84		
8	http://www.gupro.de/GXL/Introduction/section5.html	45		
9	http://www.gupro.de/GXL/Introduction/section6.html	43		
10	http://www.gupro.de/GXL/Introduction/references.html	236		
11	http://www.gupro.de/GXL/dtd/dtd.html	35		
12	http://www.gupro.de/GXL/xmlschema/xmlschema.html	42		
13	http://www.gupro.de/GXL/GraphModel/graphModel.html	46		
14	http://www.gupro.de/GXL/MetaSchema/metaSchema.html	87		
15	http://www.gupro.de/GXL/future/gxl1.1.html	30		
16	http://www.gupro.de/GXL/dtd/gxl-1.1.html	54		
17	http://www.gupro.de/GXL/examples/instance/graph/simpleExample/navigation.html	2	ja	Navigationsframe Examples
18	http://www.gupro.de/GXL/examples/schema/gxl/simpleExample/simpleExampleSchema.html	11	ja	GXL-Beispiel Examples
19	http://www.gupro.de/GXL/examples/schema/graph/simpleExample/simpleExample.html	13	ja	Graph-Beispiel Examples
20	http://www.gupro.de/GXL/examples/schema/uml/simpleExample/simpleExample.html	13	ja	Klassendiagramm-Beispiel Examples
21	http://www.gupro.de/GXL/Publications/publications.html	39		
22	http://www.gupro.de/GXL/faq/faq.html	69		
23	http://www.gupro.de/GXL/tools/tools.html	51		
24	http://www.gupro.de/GXL/changeRequest/changeRequest.html	58		
25	http://www.gupro.de/GXL/changeLog/changeLog.html	31		

Tabelle 2: Auflistung der Fehler der Validitätsprüfung des HTML-Codes

Bei der Betrachtung dieser Validitätsprüfung des HTML-Codes fällt auf, die Anzahl der Fehler hoch ist. Ohne Berücksichtigung des *Examples*-Bereichs der Website und der Webseiten, die aus XML-Code generiert wurden, sind es insgesamt 1119. Zudem schwankt die Anzahl der bei der Validitätsprüfung festgestellten Fehler erheblich (von 1 bis zu 236). Die Fehler des *Examples*-Bereichs der zu migrierenden Website lassen sich nur schätzen (siehe Punkt 17 bis 20) und liegen bei etwa (65 * 2) + (21 * 11) + (21 * 13) + (11 * 13) ~ 777.[20]

Hierbei handelt es sich allerdings nur um eine quantitative Auflistung der Fehler. Zur Analyse und zur Vorbereitung weiterer Schritte ist auch die Betrachtung der *Art der Fehler* von

[19] Zu finden unter http://validator.w3.org/.
[20] Rechnung: 64 Navigationsseiten mal 2 Fehler, 21 Graph-Beispiele mal 11, 21 GXL-Beispiele mal 13 und 11 UML-Beispiele mal 13.

Bedeutung. Auf der Ergebnisseite des Validierungsdienstes wird für jeden Fehler eine genaue Beschreibung angegeben, so dass genau nachvollzogen kann, welche Fehler vorkommen.

Die große Mehrzahl der Fehler hängen mit dem *a*-Tag zusammen, der an der Stelle, wo er im Quelltext steht, nicht zulässig ist. Als mögliche Ursache gibt der Validierungsdienst die Verwendung von XHTML-Syntax in als HTML ausgewiesenen Dokumenten an. Und tatsächlich entfallen sämtliche Fehler dieser Art, wenn die Webseite auf den Standard *XHTML 1.0 Transitional* getestet wird. Das Ergebnis der Überprüfung auf XHTML-Validität ist allerdings ähnlich schwach, da bei anderen Tags neue Fehler gefunden wurden, weil sie gegen den XHTML-Standard verstoßen.

Außerdem wurden, wenn auch nicht in dieser Menge, nicht geschlossene Tags gefunden, die auch nach dem weniger strikten *HTML 4.01 Transitional*-Standard nicht zulässig sind. In einzelnen Fällen (bei der Verwendung des *img*-Tags) wurde statt des existierenden Attributs *alt* für die Anzeige eines Alternativtextes, das nach Standard nicht erfasste Attribut *alr* gesetzt. Hierbei handelt es sich wahrscheinlich um einen einfachen Tippfehler im Quelltext der Webseite.

Abschließend beurteilt ergibt sich bezüglich der Qualität des HTML-Quelltextes der Website ein Sanierungsbedarf. Die hohe Anzahl der Fehler ist alleine kein guter Indikator für die Qualität des Quelltextes. Wie die Untersuchung herausgestellt hat, handelt es sich hierbei nur im geringen Maße um Flüchtigkeitsfehler. Vielmehr sind es systematische Fehler, die durch die Verwendung automatisierter Verfahren leicht zu beheben sind. Das Vorgehen der Sanierung des Quelltexts wird in Abschnitt 3.4.1 erläutert.

XML-Konformität
Ebenso wie bei der Beurteilung der HTML-Konformität von Webseiten wird bei der Beurteilung der Qualität des Quellcodes in den XML-Dokumenten verfahren. Auch für die Untersuchung der XML-Validität kann auf den Validierungsdienst des World Wide Web-Konsortiums zurückgegriffen werden. Statt der Untersuchung der Validität nach offiziellen Standards wie *XHTML 1.0 Transitional* und *HTML 4.01 Transitional* wird hier die Validität des Dokuments ausgehend von den referenzierten Document Type Definitions (DTD) gemessen. Die Ergebnisse der Überprüfung der XML-Validität der Dokumente veranschaulicht Tabelle 3.

#	Web-Seite	Fehler
1	http://www.gupro.de/GXL/faq/faq.xml	0
2	http://www.gupro.de/GXL/Publications/publications.xml	0
3	http://www.gupro.de/GXL/tools/tools.xml	0
4	http://www.gupro.de/GXL/changeRequest/changeRequest.xml	0
5	http://www.gupro.de/GXL/changeLog/changelog.xml	0

Tabelle 3: Auflistung der Fehler der Validitätsprüfung des XML-Codes

Im Vergleich zum HTML-Code der Webseiten, die ausschließlich in HTML vorliegen, weisen die XML-Dokumente keine Fehler auf. Sie müssen nicht saniert werden und können unverändert in der Migration verwendet werden.

3.2.4 Betriebswirtschaftliche Bedeutung

Der Bewertungsbereich „Betriebswirtschaftliche Bedeutung“ wurde in zwei Aspekte aufgespalten. Natürlich ist die *wirtschaftliche Bedeutung* der Website für die Universität und die dafür zuständige Arbeitsgruppe ein Kriterium. Aber da es in der universitären Forschung vor allem um das Schaffen und Vermitteln von Wissen geht, besitzen wirtschaftliche Aspekte nur eine sekundäre Bedeutung. Eine ausschließlich wirtschaftliche Betrachtung der Website

in der Portfolio-Analyse muss zwangsläufig zu kurz greifen. Deshalb wird zur wirtschaftlichen Bedeutung noch die *wissenschaftliche Bedeutung* als Kriterium hinzugefügt.

Zur Bewertung der beiden Unterpunkte ist die Meinung des Website-Betreibers und des Domänenexperten gefragt. Auf dem Gebiet der Einschätzung der wirtschaftlichen Bedeutung verfügt der Website-Betreiber über das notwendige Wissen, während der Domänenexperte als direkter Projektbeteiligter am besten abschätzen kann.

Auf wirtschaftlicher Ebene erfüllt die zu migrierende Website keine *wirtschaftliche Funktion*. Durch sie werden keine Einnahmen gemacht und sie wird lediglich zur Bereitstellung von Informationen betrieben. Deshalb muss die wirtschaftliche Bedeutung mit *schlecht* eingeschätzt werden. Dennoch dient sie als Website des organisationsübergreifenden GXL-Projekts zur Sammlung und Verbreitung von Wissen. Damit erfüllt sie einen wichtigen *wissenschaftlichen Zweck*, weswegen dieser Aspekt mit *sehr gut* bewertet wird.

Bei der Gewichtung der beiden Kriterien wird davon ausgegangen, dass im Rahmen einer Hochschule stets die wissenschaftliche Bedeutung im Vordergrund steht. Deshalb wird dieses Kriterium mit 0,9 und die wirtschaftliche Bedeutung mit 0,1 bewertet. Bei Einbezug der Bewertungen für die Teilbereiche ergibt sich somit der Wert 0,1 * 5 + 0.9 * 1 = 1,4. Bezüglich der wissenschaftlichen Funktion ist die zu migrierende Website sehr wichtig.

3.2.5 Portfolio-Analyse

Aus der Bewertung der Sammelkriterien wurden für die technische Qualität der Wert 2,85 und für die betriebswirtschaftliche Bedeutung der Wert 1,4 ermittelt. Zur Visualisierung wird in Abbildung 20 in Anlehnung an die Darstellung in [Ackermann (2005), S. 54] die Positionierung des Legacy-Systems innerhalb der Portfolio-Matrix dargestellt.

Wie ersichtlich, befindet sich die zu migrierende Website knapp innerhalb des Felds für Kapselungs- oder Konversionskandidaten. Unter Konversion wird die mehr oder weniger automatisierte Transformation des Alt-Systems in die Zielumgebung verstanden, während Kapselung die Erhaltung des Alt-Systems durch die Definition von Schnittstellen mit dem Zielsystem bedeutet [Ackermann (2005), S. 51]. Je nach steigendem Anteil des Wiederverwendbarkeitspotenzials des Alt-Systems, wird entweder eine Kapselung oder eine Konversion favorisiert.

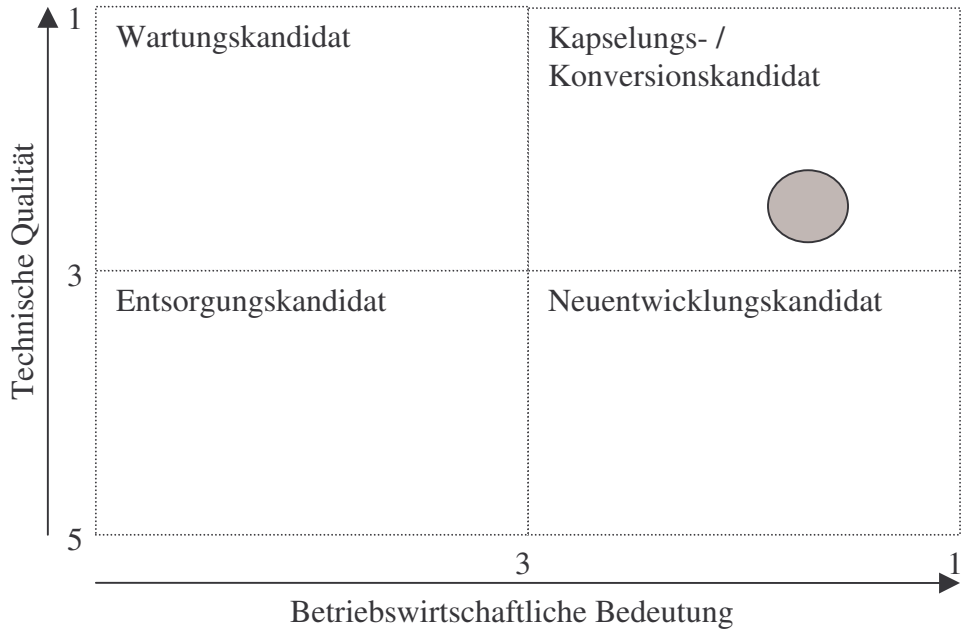

Abbildung 20: Portfolio-Matrix für die alte GXL-Website

Mit dem Ergebnis der Gegenüberstellung von technischer Qualität und betriebswirtschaftlicher Bedeutung der zu migrierenden Website lässt sich feststellen, dass grundsätzlich eine Migration sinnvoll ist. Da innerhalb der Website fast alle Daten unverändert im Zielsystem wieder verwendet werden sollen, ergibt sich bezüglich der zu wählenden Transformationsstrategie eine Empfehlung für eine Konversion.

3.2.6 Einbezug organisatorischer Faktoren

Neben den vorgenannten Kriterien zur Erstellung einer Portfolio-Analyse, ist zur Risikovermeidung innerhalb des Migrationsprojekts auch die Berücksichtigung organisatorischer Faktoren notwendig [Ackermann (2005), S.185]. Hierzu gehören z.B. die Anzahl der Mitarbeiter, technische Kompetenzen und Budget.

Bezüglich der Anzahl der Mitarbeiter kann eingeschränkt werden, dass lediglich der Verfasser der Arbeit sich mit den wesentlichen Problemstellungen der Migration beschäftigt. Er erhält zwar Unterstützung von den anderen Projektbeteiligten, muss allerdings die hauptsächliche Arbeit leisten. Zudem soll der gesamte Aufwand der Migration innerhalb der üblichen Frist zur Anfertigung einer Master-Abschlussarbeit, also in 6 Monaten durchführbar sein. Bezüglich der technischen Kompetenzen muss miteinbezogen werden, dass der Verfasser vor Beginn des Migrationsprojekts kaum über Kenntnisse des Zielsystems Plone verfügte. Im Gegensatz dazu waren dem Verfasser die Grundlagen des Legacy-Systems größtenteils bekannt. Bezüglich des Budgets sollten neben der reinen Arbeitszeit und den kalkulierbaren Kosten für Papier und Kommunikation zwischen den Projektbeteiligten keine weiteren Kosten hinzukommen.

Aus den organisatorischen Bedingungen ergeben sich einige Restriktionen. Die Dauer der Migration beträgt maximal 6 Monate und während dieser Zeit muss der Arbeitsaufwand von einer Person geleistet werden. Zudem muss der Umfang darauf abgestimmt werden, dass das Zielsystem zu Beginn der Migration nur unzureichend bekannt ist. Außerdem sind nur begrenzt finanzielle Mittel vorhanden.

3.2.7 Bewertung des Legacy-Systems

Aus der Portfolio-Analyse der wesentlichen Faktoren der technischen Qualität und betriebswirtschaftlichen Bedeutung sowie der Einbeziehung organisatorischer Faktoren lässt sich nun eine Aussage bezüglich der Migrationseignung des Legacy-Systems machen. Gemäß der Positionierung des Alt-Systems innerhalb der Matrix lässt sich ablesen, dass es eher ein Kandidat für eine Konversion darstellt. Der Abstand zur Empfehlung einer Neuentwicklung ist allerdings nicht sehr groß. Von diesen Erkenntnissen und unter Einbezug der Argumente, die im Abschnitt 5 zur Wahl der Migrationsstrategie aufgeführt werden, wird *Konversion* als Migrationsstrategie abgeleitet.

Die gegebenen organisatorischen Bedingungen unterstützen die Durchführung einer Migration. Es existieren gemessen am Umfang der Maßnahmen ausreichende, aber begrenzte Ressourcen an Arbeitskraft und Budget. Deswegen muss zu jedem Zeitpunkt der Migration der Projektstand gemessen an diesen Kriterien überprüft werden, um riskante Abweichungen von den Vorgaben zu vermeiden.

3.3 Legacy-System detailliert reverse-engineeren

Das Legacy-System wird in der Aktivitätsgruppe „Legacy-System detailliert reverse-engineeren“ gemäß der gewählten Migrationsstrategie in einem angemessenen Detaillierungsgrad untersucht. Im vorliegenden Fall einer Konversion soll Daten und Programme des Alt-Systems möglichst vollständig in das Zielsystem überführt werden. Durch die Untersuchung des Legacy-Systems in Abschnitt 3.1 wurde bereits herausgestellt, dass das Alt-System vor allem aus Daten in Form von Webseiten und XML-Dokumenten besteht. Damit liegt der Fokus der Reverse-Engineering-Aktivitäten auf der Analyse der Legacy-Daten. Bei der zweckbezogenen Beschreibung interessiert vor allem, wie das Alt-System beschrieben werden kann, um die Überführung in das Zielsystem zu erleichtern [Ackermann (2005), S. 186 / 187].

Bei der Bewertung des Legacy-Systems wurde bei der Aktualität der Daten festgestellt, dass der Navigationsbereich *Advance* nicht mehr den Anforderungen genügt. In Absprache mit dem Website-Betreiber wurde ausgemacht, dass die Inhalte dieses Navigationsbereiches nicht migriert werden sollen. Dennoch ist erfolgt in dieser Aktivitätsgruppe auch die genaue Analyse der dort verwalteten Inhalte. So können eventuell vorhandene Beziehungen zum Rest der Website entdeckt und in das Ziel-Design einbezogen werden. Ohne diese Untersuchung ist die Legacy-Analyse unvollständig.

3.3.1 Betrachtung der zu migrierenden Website

Bezüglich der Inhalte und Navigation stellt sich die zu migrierende Website überwiegend als Menge einfacher Hyperlink-Dokumente dar. So steht dessen Content direkt im HTML-Quellcode und wird nicht dynamisch erzeugt. Auch die Navigation ist relativ einfach und erfolgt über Verlinkung zu anderen Seiten der Website, zu Ankern innerhalb der Seiten und zu externen Seiten. Auf keiner der Seiten kommen Skripts, Applets oder sonstige Server- oder clientseitige Programmierung zum Einsatz.

Bei genauerer Betrachtung des internen Aufbaus wird allerdings klar, dass es sich bei der alten GXL-Website keinesfalls nur um HTML-Dokumente handelt. Stattdessen wurde bei Erstellung der Website auf diverse Hilfsmittel zurückgegriffen, die die Verwaltung der Inhalte der Webseiten erheblich vereinfachen.

3.3.2 *Makefiles*

Innerhalb aller Ordner, die im Internet veröffentlich werden, befinden sich so genannte *Makefiles*. Dieses Makefile wird von dem Programm *make* ausgeführt. *Make* wird vor allem in der Software-Entwicklung zum gezielten Kompilieren von Software-Komponenten eingesetzt. Die Beziehungen, die dabei zwischen den einzelnen Dateien einer Software bestehen, sind in dem Makefile definiert [Herold (2003], S. 37].

Wird der *make*-Befehl ausgeführt, werden die in dem Makefile beschriebenen Abhängigkeiten und damit verbundenen Befehle ausgeführt. Der typische Aufbau eines Eintrags im Makefile wird in Abbildung 21 dargestellt (in Anlehnung an Herold (2003), S. 39]). Die Abhängigkeiten des Ziels zu anderen Objekten sind in der ersten Zeile zu finden, während der Befehl in der zweiten Zeile mit einem Tabulator-Zeichen eingerückt wird.

```
Ziel: Objekt1, Objekt2 # Abhängigkeitsbeschreibung
 Befehl # Kommandozeile
```

Abbildung 21: Eintrag innerhalb eines *Makefiles*

Da man auch Beziehungen und Zusammenhänge zwischen Dokumenten wie z.B. Webseiten definieren kann, wurde im Fall der zu migrierenden Website die Funktionalität von *make* zur Generierung von Webseiten genutzt. Abbildung 22 zeigt den Inhalt eines einfachen *Makefiles* aus dem Ordner „Future“ der zu migrierenden Website. Hierin legt der Befehl *cat* fest, dass die Dateien *head.html*, *gxl1.1_body.html* und *foot.html* konkateniert werden und schließlich in die Zieldatei *gxl1.1.html* ausgegeben werden. *TARGETDIR* ist eine lokale Variable, die zur Verkürzung der Pfadangabe eingetragen wurde. Hierbei handelt es sich um die Zusammensetzung der Webseite *Future* aus einer Kopfzeile, dem eigentlichen Inhalt und einer Fußzeile.

```
# TARGETDIR = /home/user/www/GXL

All:
        cat $(TARGETDIR)/head.html $(TARGETDIR)/future/gxl1.1_body.html
$(TARGETDIR)/foot.html >> $(TARGETDIR)/future/gxl1.1.html
```

Abbildung 22: Makefile aus dem Ordner "Future" der zu migrierenden Website

Der Vorteil des *Makefiles* liegt darin, dass sobald der Befehl *make* ausgeführt wird, auch die Anweisungen im Makefile umgesetzt werden. So muss bei Änderungen innerhalb einer dieser Dateien lediglich *make* ausgeführt werden, um die zur Veröffentlichung bestimmte Zieldatei *gxl1.1.html* zu aktualisieren. Durch die konsequente Verwendung von *Makefiles* auf der gesamten Website ist die Verwaltung der Webseiten erheblich vereinfacht worden.

3.3.3 XML-Transformatoren

Neben der durchgängigen Verwendung des Befehls *cat* innerhalb der *Makefiles* zur Generierung von einheitlichen Kopf- und Fußzeilen im Dokument, wurde noch auf eine andere Funktionalität zurückgegriffen, die bei der Generierung der statischen Webseiten große Bedeutung hat. Tatsächlich beinhaltet die Website nicht nur HTML-Dokumente, sondern auch XML-Dateien. XML wird zur Beschreibung von beliebigen Datenstrukturen verwendet [Meinel / Sack (2004), S. 944]. XML-Dokumente enthalten ein Wurzelelement, von dem aus sich weitere Elemente baumartig verzweigen. Dabei ist die Art der Formatierung der Elemente, wie sie bei HTML ein zentrales Anliegen darstellt, zunächst unwichtig. Von Bedeutung ist hingegen eine datenorientierte Beschreibung der Inhalte.

Allerdings können XML-Dokumente in der Form wie sie vorliegen nicht direkt publiziert werden, da sie keinerlei Formatierungsanweisungen enthalten. Für die Transformation von

XML-Dokumenten existieren aber Transformationswerkzeuge, die den XML-Code nach Transformationsanweisungen zur Anzeige in HTML umwandeln. Außerdem existieren Transformatoren, die nach Kriterien die Erzeugung dynamischer Dokumente ermöglichen, wie z.B. die Erstellung einer Liste. Hierfür ist häufig die Verwendung einer Programmiersprache notwendig [Meinel / Sack (2004), S. 989]. Tatsächlich sind beide Formen von Transformationen, also der Darstellung und dynamischen Restrukturierung, auf der alten GXL-Website in Anwendung.

Für die Festlegung der Transformationen wurde auf die Technologie der Extensible Style Language (XSL) und XSL Transformations (XSLT) zurückgegriffen. In einem XSL-Stylesheet werden die Anweisungen zur Transformation der Elemente des XML-Dokuments, die so genannten *Templates*, vorgeschrieben [Meinel / Sack (2004), S. 991]. Das XSL-Stylesheet und das XML-Dokument werden schließlich von einem XSLT-Prozessor verarbeitet und das transformierte Dokument ausgegeben.

Zur Verdeutlichung der Funktionsweise der XSL-Stylesheets befindet sich in Abbildung 23 ein Auszug aus dem Stylesheet für die Transformation von FAQs. Hierbei wird für jedes Element, das vom Typ *question* ist, das nachfolgende Template angewandt. Zunächst wird mittels des *a*-Elements ein Anker hergestellt, dessen Attribut *name* den Wert des Elementattributs „id“ erhält. Schließlich wird noch der Inhalt des Elementattributs *text* durch Einklammern in das *b*-Element fett gedruckt. Die Anweisung <*xsl:apply-templates*> besagt, dass an dieser Stelle die weiteren Template-Anweisungen ausgeführt werden sollen. Sie ist dann notwendig, wenn innerhalb eines Elements noch andere Elemente verschachtelt sind, die im Stylesheet eigene Formatierungsanweisungen besitzen.

```
<xsl:template match="question" name="question">
 <a>
 <xsl:attribute name="name">
  <xsl:value-of select="@id" />
 </xsl:attribute>
 </a>
 <b>
 <xsl:value-of select="@text' />
 </b>
 <br />
 <xsl:apply-templates />
 <p />
</xsl:template>
```

Abbildung 23: Auszug aus dem XSL-Stylesheet der Webseite FAQ

Auf der GXL-Website wird hierfür der Transformator *Xalan* verwendet.[21] Die Anweisung zur Transformation lässt sich in mehreren *Makefiles* nachvollziehen. Transformationen werden auf den folgenden Webseiten verwendet:

- *FAQ*
- *Publications*
- *Tool Catalogue*
- *Change Requests*
- *Change Log*

Bei diesen Seiten wird die Berücksichtigung der existierenden XML-Dokumente und der dazugehörigen XSL-Stylesheets für die Migration von Bedeutung sein. Ebenso existieren für die XML-Dokumente DTDs, die die Datenstruktur der Dokumente beschreiben. Zur Abbildung des Ist-Datenmodells der betroffenen Datentypen der zu migrierenden Website bieten sich diese Definitionen unbedingt an.

[21] Siehe http://xalan.apache.org/

Struktur der zu migrierenden Website

Auf der Ausgangsseite unter http://www.gupro.de/GXL/ erfolgt die globale Navigation über eine immer gleiche Leiste von Navigationslinks. Die Seiten dieses Bereichs der Webseite wurden nicht mit Frames realisiert. Tatsächlich bestehen die Seiten aus einer Tabellenstruktur, die 5 Zeilen mal 2 Spalten umfasst. Die Elemente sind nachfolgend aufgelistet und in Abbildung 24 veranschaulicht.

1. GXL-Logo (1. Zeile / 1. Spalte)
2. Kopfzeile mit Kontaktangaben (1/2)
3. Vertikallinie (2/1 und 2/2)
4. Navigationsangaben für die gesamte Website (3/1)
5. Anzeige für den aktuellen Inhalt (3/2)
6. Vertikallinie (4/1 und 4/2)
7. Letztes Änderungsdatum und Link zum Change Log (5/1)
8. Link auf druckbare Version der Seite (5/2)[22]

Abbildung 24: Homepage der zu migrierenden Website

Alle Elemente dieser Seite, bis auf die Zelle für den aktuellen Inhalt, bleiben bei Aufruf von neuen Seiten der Website gleich. Die Konsequenz aus der starren Tabellenstruktur der Hauptseite ist, dass für die als Bestandteil der Hauptseite angezeigten Seiten sämtliche Tabellenelemente in den Quelltext kopiert werden müssen. So findet sich in den HTML-

[22] Hier wird nicht tatsächlich ein Link zur Verfügung gestellt, der die druckbare Anzeige aller Dokumente dieser Website ermöglicht. Stattdessen wird hier auf eine Übersichtsseite verlinkt (background.html), die nur diesen Text außerhalb der Tabellenstruktur anzeigt.

Dateien der Website zunächst ein hohes Maß an Redundanz für die statischen Elemente der Website.

Dennoch muss festgehalten werden, dass die sich wiederholenden Elemente aller Webseiten vor Abruf des Webseitenbetrachters durch die Ausführung der Anweisungen in den *Makefiles* ähnlich wie in Abbildung 22 generiert wurden. Tatsächlich umfasst die Kopfzeile (*head.html*), Bezug nehmend auf Abbildung 24, die Tabellenabschnitte 1 bis 4, während die Fußzeile (*foot.html*) die Tabellenabschnitte 6 bis 8 abdeckt. Das heißt, dass alle Webseiten gleichermaßen diese Codeelemente enthalten, dennoch die Veränderung der Kopf- oder Fußzeile zentral vorgenommen werden kann und nicht in jedem Dokument nachvollzogen werden muss. Die Ausführung der Inhalte der Makefile reicht dafür aus.

Es werden allerdings nicht alle über die Navigationszelle erreichbaren Links im Layout der Hauptseite angezeigt. Ausnahmen bilden hier die Beispiele (*Examples*), die in einem neuen Fenster geöffnet werden. Zunächst wird der Bereich für die Beispiele erläutert.

Innerhalb des Hauptbereichs sind fast alle Daten für die Migration interessant. Eine Ausnahme bildet die Webseite *Change Log* mit der Übersicht der Änderungen, die in der Vergangenheit auf der Website durchgeführt wurden. Hierbei handelt es sich um Einträge, die um vier Jahre zurückliegen. Da sich die Struktur der Website durch die Migration ohnehin verändern wird, ist es nicht sinnvoll diese Informationen zu übertragen. Dieser Bereich wird aus diesem Grunde auch in der detaillierteren Beschreibung des Ist-Datenmodells der zu migrierenden Website nicht weiter untersucht.

3.3.4 *Examples*-Webseiten

Der *Examples*-Bereich unterscheidet sich von den anderen Seiten der Website. Neben anderen Navigationsinhalten greifen sie auch auf weitere Techniken in HTML zurück und benutzen Frames und Formularelemente. Die Webseiten sind unterteilt in Navigations-Frame und Content-Frame. Die darstellbaren Beispiele sind in Gruppen unterteilt. Für jedes Beispiel kann die Anzeige von Instanzen und Schemata ausgewählt werden. Instanzen umfassen ein Graph- und ein GXL-Beispiel, während Schemata zusätzlich noch ein Klassendiagramm-Beispiel anzeigen. Eine Ausnahme bildet das Metaschema. Dort können lediglich die Beispiele für Schemata abgerufen werden. Insgesamt gibt es 10 * (2 + 3) + 3 = 53 Beispiele (10 normale Beispiele mit je 2 Instanz- und drei Schema-Beispielen, sowie drei Metaschema-Beispiele).

Die Navigation für den Beispiel-Bereich ist statisch. Hier ist die lokale Navigation für diesen Bereich in einem Frame als statische HTML-Seite realisiert. Wenn dort ein Navigationslink angeklickt wird, wird eine neue Navigationsseite in den Frame geladen, die bei dem Nutzer den Eindruck einer dynamisch generierten Navigation erweckt. Für diesen Effekt ist für jeden Beispieltyp (z.B. einfach, komplex, hierarchisch) und seine 5 Beispiele eine eigene Navigation abgelegt, die bei Aufruf in das Navigationsframe geladen wird. Dabei wird das im Content-Frame angezeigte Beispiel in der Navigation rot markiert. Hinzu kommt noch eine neutrale Navigationsansicht für jedes Beispiel, indem keines markiert ist, aber alle Beispieloptionen sichtbar sind. Insgesamt handelt es sich hierbei um 1 + (10 * 5) + (1 * 3) + (11 * 1) = 65 Navigationsseiten.[23]

Zudem ist es nicht nur über die Navigationsleiste möglich, auf der Seite zu navigieren, sondern auch über Navigationselemente in der Kopfzeile der Webseiten. Diese Navigation verändert sich, ähnlich wie in der Navigationsleiste, abhängig von dem angezeigten Inhalt.

[23] Rechnung: Ausgangsnavigation, normale Beispieltypen, Metaschema-Beispiel und neutrale Navigation.

Die Ansteuerung der Seiten kann sowohl über Hyperlinks als auch mit Button mit Kombinationsfeldauswahl durchgeführt werden.

3.3.5 Navigationsmodell der zu migrierenden Website

Die Inhalte der Navigationsleiste, die die globale Navigation der Website gewährleistet, wurden zur Ableitung von Bereichen innerhalb der Website verwendet. Allerdings existiert zwischen den Webseiten selber eine Vielzahl von Verweisen durch Nutzung von Hyperlinks. Die Betrachtung dieser lokalen Navigationselemente ist für die Analyse der zu migrierenden Website ebenfalls von Bedeutung. Auf diese Weise kann festgestellt werden, zwischen welchen Webseiten Abhängigkeiten bestehen, die bei der Migration berücksichtigt werden müssen.

Für das Auffinden von Verweisen zwischen den Webseiten wurde das Werkzeug *WebLoupe* verwendet. Es handelt sich um ein Crawler-Werkzeug, das die Analyse von Websites an Hand der Hyperlinks innerhalb der Seiten ermöglicht. Hierfür wurde für jede Webseite der Legacy-Website ein Crawl-Vorgang gestartet.

Die Ergebnisse der Untersuchung mit WebLoupe wurden anschließend in ein Klassendiagramm übertragen. Eine Klasse entspricht dort einer Webseite und die Assoziationen zwischen den Klassen stellen Hyperlinks dar. Die Richtung einer Assoziation zeigt dabei die Aufrufsrichtung eines Hyperlinks. Geht von einer Klasse A eine gerichtete Assoziation zu einer Klasse B aus, bedeutet dies, dass sich auf Webseite A ein Hyperlink zu Webseite B befindet. Ungerichtete Assoziationen stehen für die Existenz von Hyperlinks sowohl von der einen, als auch von der anderen Webseite. Der Stereotyp <<*document*>> stellt eine Ansammlung von Webseiten mit ähnlichem Navigationsverhalten dar (siehe *Examples, IntroductionRest* und *GXL 1.1*). Es wurden nur die Links innerhalb der Website berücksichtigt. Das resultierende Navigationsnetz ist in Abbildung 25 zu sehen.

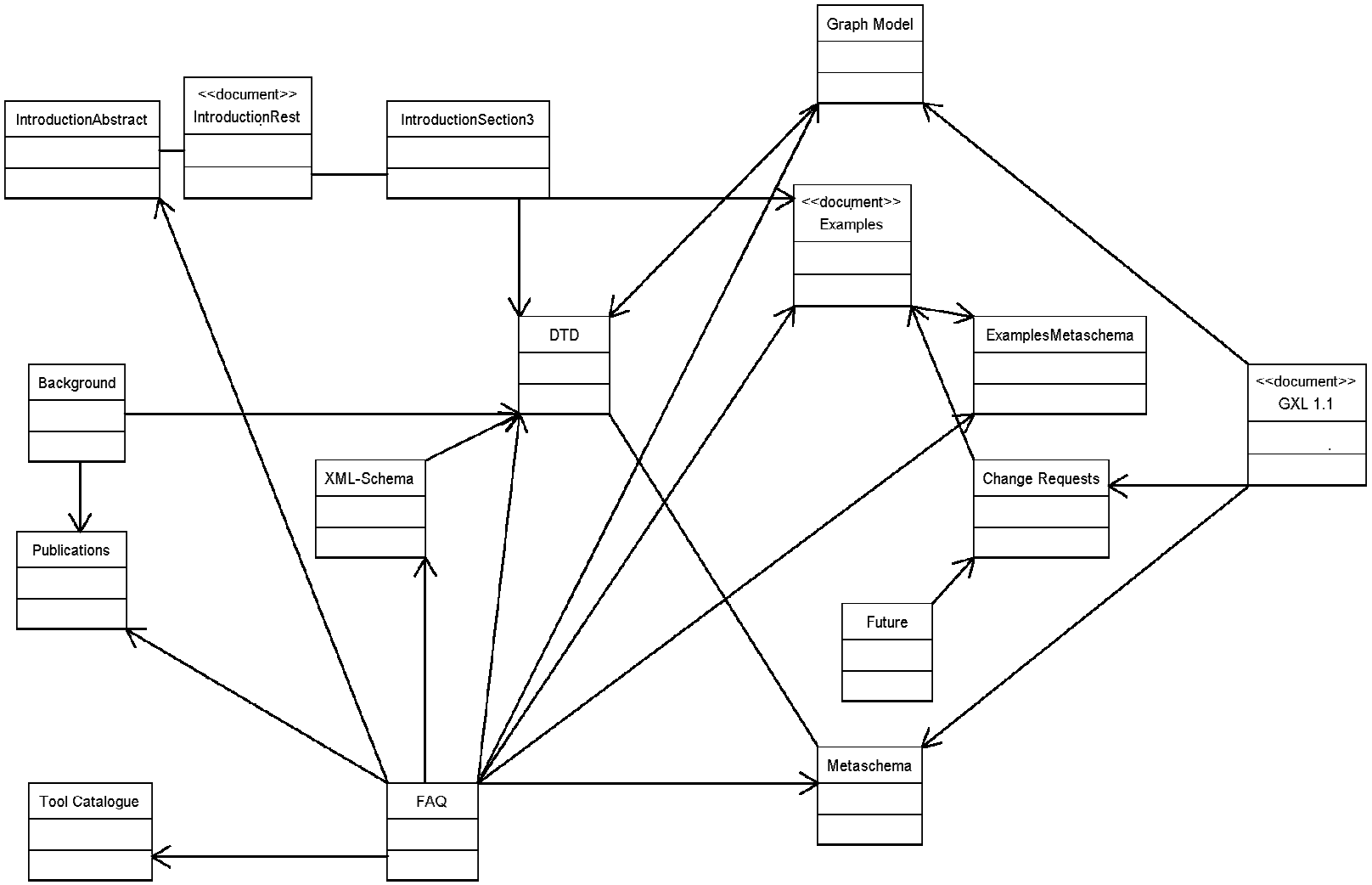

Abbildung 25: Netz der lokalen Navigation zwischen den Webseiten der Legacy-Website

In Abbildung 25 wird deutlich, dass die Webseiten miteinander intensiv verlinkt sind. Dabei gibt es deutliche Unterschiede, ob eine Webseite eher auf andere Webseiten referenziert, oder ob auf sie selbst referenziert wird. Besonders viele Verweise auf andere Webseiten macht *FAQ*. Auf der anderen Seite existieren viele Verweise auf die Webseite *DTD*. Dieser Tatbestand lässt sich damit erklären, dass *FAQ* vor allem Information zum Auffinden von Daten, während *DTD* als Referenzpunkt die eigentliche Information enthält.

Ebenso kann nachvollzogen werden, dass es Gruppen von Webseiten gibt, die vor allem untereinander referenzieren. Dies ist z.B. bei *IntroductionAbstract*, *IntroductionSection3* und *IntroductionRest* der Fall. Diese drei Webseiten, bzw. Ansammlungen von Webseiten, gehören schließlich zusammen zu einem Artikel. Die einzelnen Passagen der Artikel wurden aufgeteilt, um die Verlinkung der Webseiten besser darstellen zu können (wie z.B. von FAQ zu IntroductionAbstract). Das Navigationsnetz der zu migrierenden Website wird in Abschnitt 3.4 weiter beschrieben und strukturiert.

3.3.6 Ist-Datenmodell der zu migrierenden Website

Um ein ausführliches Bild über den auf der Website gehaltenen Content zu erhalten, ist eine detaillierte Betrachtung der darin enthaltenen Daten notwendig. Zur Modellierung des in der zu migrierenden Website enthaltenen Contents werden UML-Klassendiagramme verwendet. In der Datenmodellierung werden diese häufig neben Entity-Relationship-Diagrammen eingesetzt. Im Fall der Content-Modellierung handelt es bei den Klassen um die verschiedenen Arten von Content, deren Elemente wie z.B. Überschrift, Beschreibung und Links als Attribute in den Klassen abgebildet werden. In den Klassendiagrammen geben die gerichteten Pfeile der Assoziationen nicht die Navigierbarkeit zwischen den Klassen an (vgl. [Jeckle et al. (2004), S. 81]), sondern die Leserichtung.

Bei der Modellierung der Daten der Legacy-Website wurde bereits auf die spätere Umsetzung der Daten im Zielsystem Rücksicht genommen. Zwar werden die Daten in den Zusammenhängen, wie sie auf der Website auftreten, beschrieben, allerdings auf einem abstrahierten, problemspezifischen Detaillierungsniveau, wie auch durch den ReMiP gefordert [Ackermann (2005), S. 186]. Statt auf den genauen Aufbau XML- und vor allem der HTML-Dokumente einzugehen, wurde vielmehr darauf geachtet, welcher Content sich auf der Website befindet. Tatsächlich ähneln die hier vorgestellten Modelle dem System der Content-Typen. Diese abgeleiteten Modelle lassen sich später im Ziel-Design leichter auf das Zielsystem übertragen.

Innerhalb der zu migrierenden Website gibt es mehrere Quellen, die zur Extraktion des Ist-Datenmodells herangezogen werden können. Die offensichtlichste Quelle zur Extraktion ist der HTML-Code der einzelnen veröffentlichten Webseiten. Die Inhalte dieser Webseiten wurden bereits im Abschnitt 3.1 erläutert. Die in den HTML-Dokumenten gehaltenen Daten wurden mangels einer Datenbeschreibung im Legacy-System relativ frei auf die Datentypen des Ist-Datenmodells abgebildet.

Neben den Webseiten, deren Daten sich lediglich im HTML-Code befinden, existieren aber auch Webseiten, die auf Grundlage von XML-Dokumenten generiert wurden. Für diese Gruppe, die in diesem Abschnitt unter „XML-Transformatoren“ genannt wurde, sind auch DTDs vorhanden, die den genauen erlaubten Aufbau der XML-Dokumente definieren. Diese DTDs sind auf Grund der übersichtlichen Strukturierung als Quellen für das Datenmodell besonders geeignet. Zur Überprüfung wie die in den DTDs definierten Elemente in den Dokumenten angewandt wurden, werden allerdings auch die XML-Dokumente selbst in die Modellierung einbezogen. Abbildung 26 zeigt die Verteilung der unterschiedlichen Quellen zur Extraktion des Ist-Datenmodells.

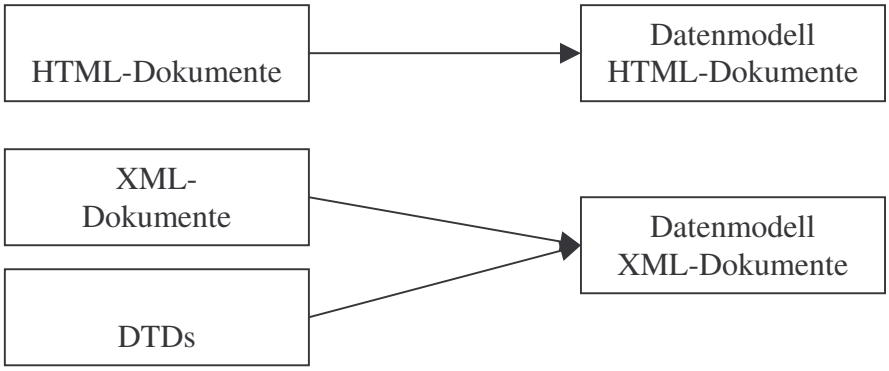

Abbildung 26: Verteilung der Quellen für das Ist-Datenmodell

Es wird ausdrücklich darauf hingewiesen, dass die Datenmodelle ohne Verwendung eines Werkzeugs erstellt wurden. Dies ist bei den HTML-Dokumenten auch verständlich, da dort keine im Quelltext nachvollziehbare Grundlage zur Extraktion von Datenstruktur besteht. Anders verhält es sich mit den XML-Dokumenten und den Einstellungen in Zope. Hier liegen in Form von DTD bzw. *Factory Type Information* Datenformate vor, die eine automatisierte Extraktion der Datenstruktur erlauben. Zur Modellierung der problemspezifisch wichtigen Elemente ist allerdings die Modellierung von Hand vorteilhaft. Auch unter Berücksichtigung der zeitlichen Ressourcen lohnt sich die Entwicklung eines automatisierten Werkzeugs zur Datenmodellierung im vorliegenden Fall nicht.

Die im Ist-Datenmodell vorgenommene Strukturierung soll bei der Übertragung in das Zielsystem Plone Hinweise zu Design-Entscheidungen geben. Bei der zu migrierenden Website handelt es sich überwiegend um HTML-Dokumente. Die darin enthaltenen Daten, die als Content identifiziert werden können, liegen in unaufbereiteter Form vor. Daten, die aus strukturierten XML-Dokumenten abgeleitet werden können, bieten für die Transformation eine bessere Grundlage. Bei der Beschreibung der Datentypen wird nochmals explizit darauf hingewiesen, ob die Daten ursprünglich aus einem XML-Dokument stammen. Anbei wird dieser in Form von Klassen im Datenmodell beschrieben, die aus der existierenden Website abgeleitet werden konnten. Es folgt die Beschreibung jeder einzelnen Webseite.

Webseite *Background*

Die zu migrierende Website wurde zur Präsentation der Ergebnisse eines Projekts aufgebaut, hier mit dem Datentyp *Project* erfasst. Zwar kommt diese Klasse von Content auf der Website nur einmal vor, dennoch stellt er eine sinnvolle Abstrahierung der dargestellten Website-Inhalte dar.

Von der Beschreibung des Datentyps *Project* kann direkt auf die nächste Art von Content verwiesen werden, die Person *(Person)*. Informationen zu Personen werden an vielen verschiedenen Stellen benötigt und sollten separat modelliert werden. Zwischen Projekt und Person existiert die Assoziation *consistsOf*. Damit wird die Beziehung zwischen einer Person dargestellt, die an einem Projekt beteiligt ist. Dementsprechend wurde der Assoziationsendpunkt mit der Bezeichnung *participant* versehen.

In engem Zusammenhang mit den Datentypen *Project* und *Person* stehen *Organisation* und *OrganisationalUnit*. *Organisation* beschreibt eine Organisation wie ein Unternehmen, eine Universität oder einen Verband. Diese Organisation besteht wiederum aus Organisationseinheiten, den *OrganisationalUnits*. Somit sind Organisationseinheiten vollständig in Organisationen enthalten, wie durch die Komposition zwischen den beiden Datentypen ausgedrückt.

In Zusammenhang mit Projekten können Organisationseinheiten unterstützend tätig sein. Diese Beziehung findet sich in Abbildung 27 in der Assoziation *isSupportedBy* wieder. Dabei wird die Organisationseinheit, die das Projekt unterstützt, mit *partner* gekennzeichnet.

Besonders vielfältige Beziehungen herrschen zwischen *OrganisationalUnit* und *Person*. Hierbei werden insbesondere Beschäftigungsverhältnisse und hierarchische Ordnungen repräsentiert. Die Assoziation *isLeadBy* deutet darauf hin, dass eine oder mehrere Personen als Leiter einer Organisationseinheit verantwortlich zeichnen. Der Leiter wird durch den Assoziationsendpunkt *head* bezeichnet. Neben dem Leiter sind auch andere Personen in irgendeiner Form in einer Organisationseinheit integriert. Diese Beziehung findet sich in der Assoziation *employs* wieder, wobei der Beschäftigte die Bezeichnung *employee* erhält. Aber nicht nur auf aktuell Beschäftigte soll verwiesen werden, sondern auch auf ehemalige Mitarbeiter. Die Assoziation *hadEmployed* mit dem Endpunkt *formerEmployee* bei Person beschreibt diese Beziehung.

Projekte und Projektergebnisse werden häufig auf Tagungen und Präsentationen vorgestellt. Für den wissenschaftlichen Austausch ist die Bezugnahme auf solche Ereignisse wichtig. Innerhalb des Klassendiagramms in Abbildung 27 wird deshalb der Datentyp *Conference* modelliert, der die wichtigsten Informationen rund um Tagungen zusammenfasst. Die Zuordnung einer Konferenz zu einem Projekt wird durch die Assoziation *isDiscussedAt* geleistet, wobei der Endpunkt der Assoziation bei *Conference* den Namen *projectConference* erhält.

Abbildung 27 gibt eine Übersicht der Datenmodellierung der Webseite *Background* und gibt die Eigenschaften der Datentypen detailliert wider.

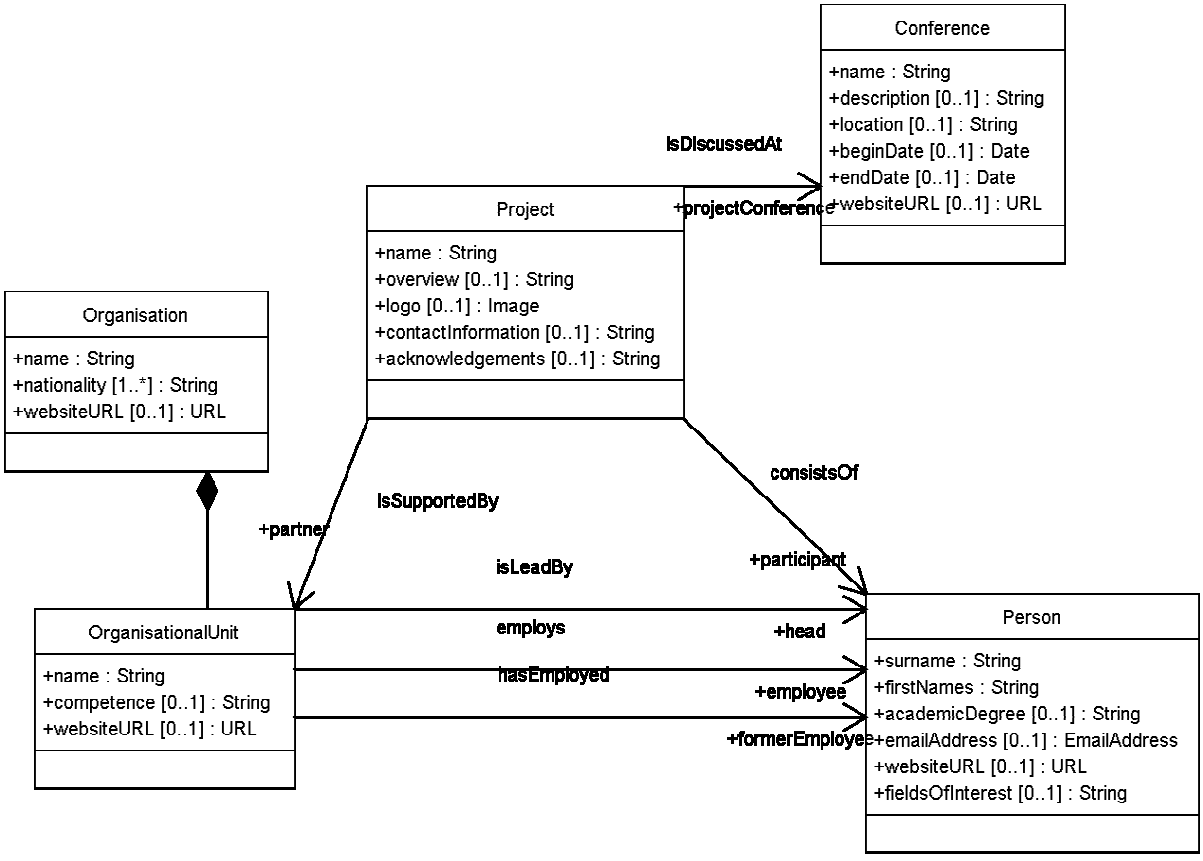

Abbildung 27: Klassendiagramm der Legacy-Webseite *Background*

Webseite *Introduction*

Die einzelnen Abschnitte des auf der Webseite *Introduction* befindlichen Artikels können mit normalen Webseiten abgebildet werden. Sie sind durch den Datentyp *Document* repräsentiert. Zur Zusammenfassung der Inhalte Dokuments dient der Datentyp *WebArticle*. Er gruppiert nicht nur die Abschnitte, sondern verfügt auch über Funktionen zur Erzeugung von Navigationslinks innerhalb des Dokuments. Durch die Funktionen *goToPrevious* und *goToNext* ist die lokale Navigation zwischen den Abschnitten nach hinten und nach vorne möglich. Ein Inhaltsverzeichnis wird durch die Funktion *createToc* erstellt. Das Datenmodell für diesen Bereich der Website ist in Abbildung 28 dargestellt.

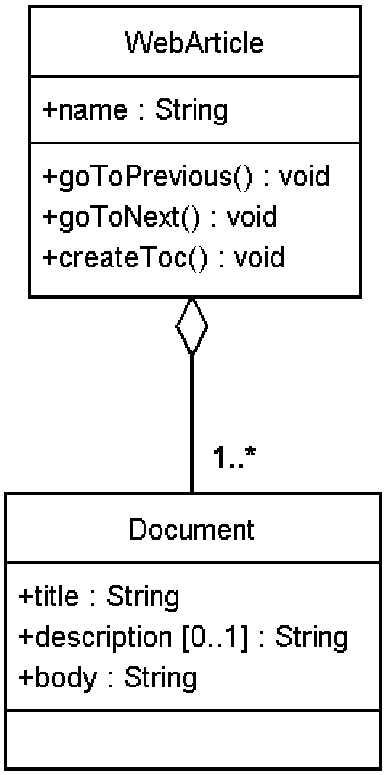

Abbildung 28: Klassendiagramm des Navigationspunkts *Introduction*

Webseite *Publications*

Die Webseite *Publications* mit einer Liste besonders lesenswerter Veröffentlichungen und Präsentationen zum Thema GXL wurde aus einem XML-Dokument generiert, wo die eigentlichen Daten gespeichert sind. Das Datenmodell der Publikationsliste kann in Abbildung 29 nachvollzogen werden.

Eine Veröffentlichung ist mit dem Datentyp *Publication* erfasst. Elemente dieses Typs werden innerhalb des Wurzelelements *Publications* gespeichert. In Publication wird lediglich der Titel der Veröffentlichung als Eigenschaft festgehalten. Die übrigen Informationen werden über Beziehungen gespeichert.

Zur Zuweisung eines oder mehrerer Autoren zu einer Veröffentlichung dient die Beziehung zwischen *Publication* und *Author*. Um Quellenangaben zu einem Literatureintrag zu machen, wird von *Publication* eine Beziehung zu *Description* hergestellt. Auf online verfügbare Dokumente kann wahlweise durch eine Beziehung zwischen *Publication* und *Link* oder *Source* verwiesen werden. Der Unterschied zwischen den beiden Datentypen ist das zusätzliche Attribut bei *Source* zur Eintragung der Größe des Dokuments.

Da einige Elemente von *Publication* bei Ereignissen wie Tagungen veröffentlicht wurden, existiert die Möglichkeit, diesen Sachverhalt durch eine Beziehung von *Publication* zu

Presented auszudrücken. Der Datentyp *Presented* enthält selbst keine Eigenschaften und wird ausschließlich durch seine Beziehungen definiert. Durch die Beziehung zwischen *Presented* und *Event* wird eingetragen, bei welchem Ereignis das Dokument präsentiert wurde. Für eine Beschreibung dieses Vortrags kann eine Beziehung zu *Description* hergestellt werden. Ebenso wie bei *Publication* können *Presented* Hyperlinks der Datentypen *Link* und *Source* zugewiesen werden.

Wenn es nicht möglich ist, eine Information zu einer Veröffentlichung in die genannten Datentypen einzuordnen, wird der Datentyp *OtherPublications* verwendet, der *Publication* über eine Beziehung zugeordnet werden kann. *OtherPublication* verfügt wie *Presented* über Beziehungen zu *Link*, *Source* und *Description*.

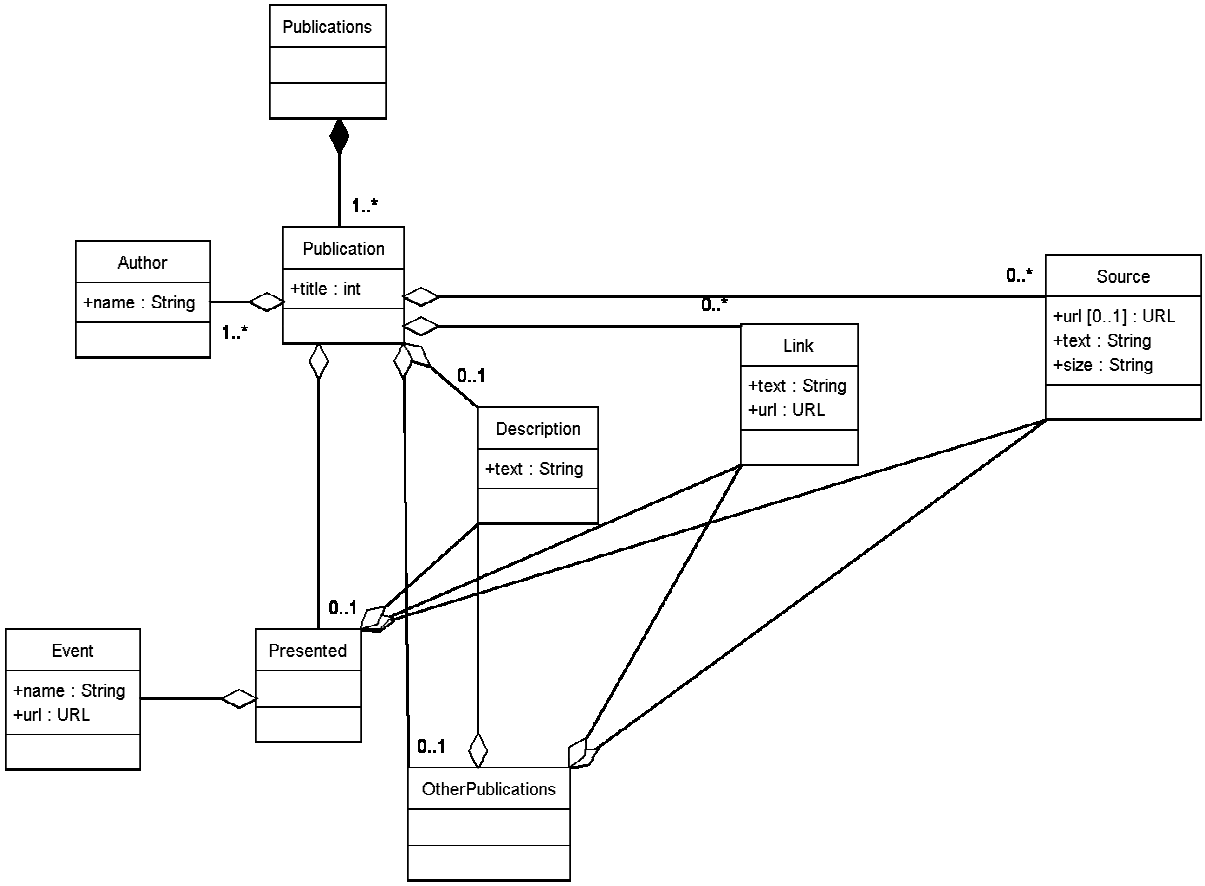

Abbildung 29: Klassendiagramm der Legacy-Webseite *Publications*

Webseite *Tool Catalogue*

Unter dem Navigationspunkt *Tool Catalogue* können Informationen zu inhaltlich mit dem Projekt verbundenen Werkzeugen angezeigt werden. Deren Daten liegen nicht nur auf der Webseite vor, sondern sind zudem noch in einem XML-Dokument gespeichert. Die Beschreibung des Datenmodells der Inhalte dieser Webseite orientiert sich deshalb wesentlich an dem im XML-Dokument vorgefundenen Datenmodell, das in seiner DTD festgeschrieben ist. Das in der Legacy-Analyse abgeleitete Datenmodell kann in Abbildung 30 nachvollzogen werden.

Ein Werkzeug wird durch den Datentyp *Tool* beschrieben. Mehrere Werkzeuge werden innerhalb einer Werkzeugliste zusammengefasst, einer *ToolList*. Besonders hervorzuheben unter den Eigenschaften von *Tool* ist *type*. Es beschreibt die Zugehörigkeit zu einer Gruppe von Werkzeugen, wie Konvertierer oder Visualisierungswerkzeuge. Neben seinen Eigenschaften wird *Tool* auch durch seine Beziehungen zu anderen Datentypen beschrieben.

Die Beziehung mit dem Datentyp *Platform* legt fest auf welchen Plattformen das jeweilige Werkzeug lauffähig ist. *Platform* verfügt hierfür über eine Eigenschaftsliste von Plattformen, bei denen durch das Setzen eines booleschen Werts festgelegt wird, ob das Werkzeug auf der Plattform läuft.

Der Datentyp *Links* fasst drei für ein Werkzeug wichtige Internet-Adressen zusammen. Es kann eine Adresse für die Homepage eines Werkzeugs, sowie die Internet-Adressen für den Download des Quellcodes oder der Installationsdatei des Werkzeugs angegeben werden.

Zur Eintragung der Beziehung zwischen einem Werkzeug und den bereitstellenden Personen oder Organisationen bzw. Organisationseinheiten dient der Datentyp *Supplier*. Bemerkenswert ist hier, dass Informationen zu Person und Organisation, bzw. Organisationseinheit, in einem Datentyp zusammengefasst sind. Diese Datentypen wurden bereits als eigenständige Daten beschrieben.

Um einen beschreibenden Text für jedes Werkzeug einfügen zu können, wird auf den Datentyp *Comment* zurückgegriffen. Dieser besteht aus XML-Elementen, die zur Eintragung von Text mit Hyperlinks entwickelt wurden. Sie wurden aus Übersichtsgründen in der Eigenschaft *text* zusammengefasst.

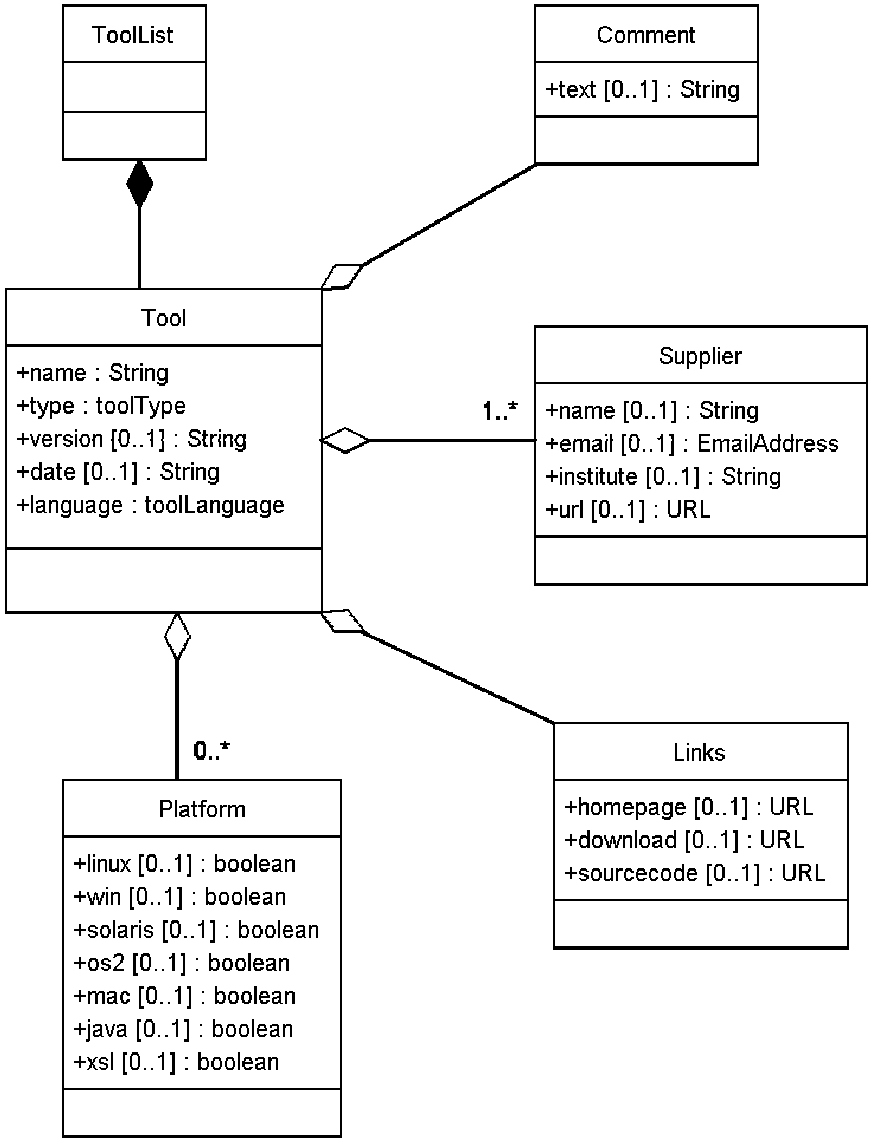

Abbildung 30: Klassendiagramm der Legacy-Webseite *Tool Catalogue*

Webseiten *Definition* und *GXL 1.1*

Im Navigationsbereich *Definition* und auf den Webseiten zu GXL 1.1 wird die Sprache GXL durch ein XML-Schema und mehrere Document Type Definitions (DTD) beschrieben. Diese Webseiten enthalten Quelltext in verschiedener Form und mit zusätzlichen Informationen. Neben den HTML-Dokumenten liegen keine weiteren Datenbeschreibungen vor.

Für die Darstellung der Quelltexte wird der Datentyp *Listing* definiert. Listing wird als logischer Datentyp beschrieben. Alle Quelltexte sind zwar nur als HTML-Dokumente gespeichert sind, weisen aber untereinander Ähnlichkeiten auf. Diese strukturellen Ähnlichkeiten wurden in Listing zusammengefasst.

Wichtig sind hier vor allem die Angaben der Version und der Sprache des Quelltextes sowie die Nennung von Urheberrechtsansprüchen. Der Code selbst liegt entweder als einfach kommentierter Text mit Formatierung und Hyperlinks, als Syntax hervorgehobener Text und als unkommentierter Quelltext vor. Da unkommentierter Quelltext auf der zu migrierenden Website immer als Datei zur Verfügung gestellt wird, wurde die Eigenschaft *sourceCodeUncommented* als Datei modelliert. Das Datenmodell von *Listing* ist in Abbildung 31 zu finden.

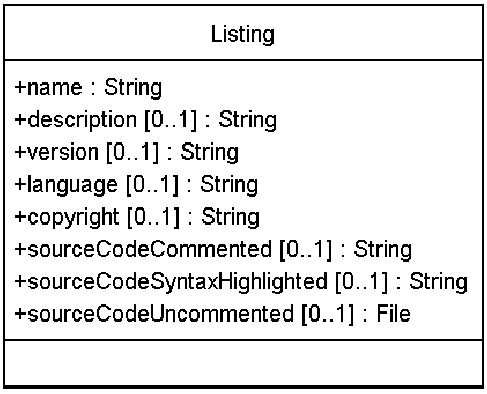

Abbildung 31: Klassendiagramm der Legacy-Webseiten *Definition* und *GXL 1.1*

Webseiten *Schemas* und *Future*

Auf den Webseiten *Schemas* und *Future* befinden sich einfache Inhalte, die schwer als eigene Datentypen zu modellieren sind. Vielmehr kann man sie dem Datentyp *Document* für einfache Webseiten zuordnen. Zur genaueren Beschreibung von *Document* siehe Abbildung 28 zur Modellierung der Webseiten *Introduction*.

Webseiten *Examples*

Im Bereich *Examples* gibt es eine umfangreiche Übersicht zu Graphbeispielen und deren Entsprechungen in GXL. Diese Beispiele sind zwar zusammenhängend auf Webseiten dargestellt, allerdings sind die Dateien, die zur Generierung dieser Webseiten verwendet werden, an separaten Stellen nach Bildern für Klassendiagramme und Graphen sowie GXL-Quelltext gespeichert.

Zur Beschreibung der auf den Examples-Webseiten vorhandenen Daten wurde der Datentyp *Examples* entwickelt. Er greift auf die Gemeinsamkeiten der verschiedenen Beispiele auf den Webseiten zurück und stellt die als eigenständige Dateien gespeicherten Elemente Klassendiagramm, Graph und GXL-Text als Eigenschaften *classDiagram*, *graph* bzw. *gxl* eines Beispiels heraus. Die Eigenschaft *classDiagram* ist als optional modelliert, da Instanzbeispiele nicht über eine Darstellung als Klassendiagramm verfügen. Zur Abbildung der Gruppierung der Beispiele auf der Webseite, erhalten sie die Eigenschaft *category*, mit deren Hilfe sie einer Gruppe zugeordnet werden können.

Zwischen einzelnen Beispielen gibt es eine hierarchische Ordnung, da Beispiele in verschiedenen Detaillierungsstufen dargestellt werden. Um eine solche hierarchische Beziehung zwischen Beispielen zu modellieren, existiert die rekursive Beziehung *instance*. Abbildung 32 zeigt das Datenmodell für den *Examples*-Bereich.

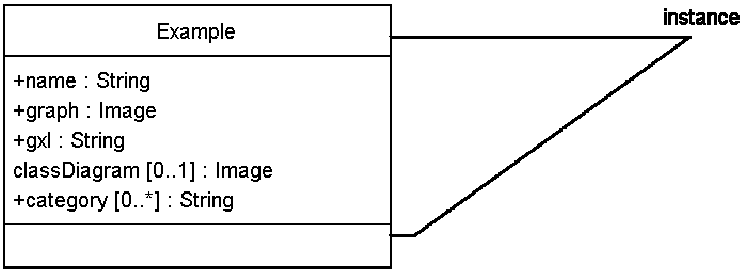

Abbildung 32: Klassendiagramm der Legacy-Webseiten *Examples*

Webseite *FAQ*

Zur schnellen Beantwortung häufig gestellter Fragen wurde eine Webseite *FAQ* erstellt. Neben deren Inhalten sind die Daten auch in einem XML-Dokument gespeichert. Das Datenmodell für die Webseite FAQ ist in Abbildung 33 dargestellt. Es wurde auf Grundlage der dazugehörigen DTD rekonstruiert, wobei deren Element-Beschreibungen in Datentypen der Legacy-Website abgebildet wurden.

Die Daten für die FAQs sind sehr verschachtelt. Zur Sammlung aller Elemente, die mit dem Bereich *FAQ* zu tun haben, existiert der Datentyp *FAQ*. Innerhalb von *FAQ* werden zur Zuordnung der FAQs zu einem bestimmten Thema Elemente des Datentyps *Topic* zugeordnet. Der Name des Themenbereichs wird in der Eigenschaft *name* eingetragen.

Topic wiederum umfasst Einträge des Datentyps *Question*. Dort wird in der Eigenschaft text die Frage für die jeweilige FAQ eingetragen. Innerhalb eines *Question*-Elements wird genau ein Element des Datentyps *Answer* zugeordnet. *Answer* besteht seinerseits aus einer Anordnung von Text und Elementen des Datentyps *LinkFAQ*. Er dient zur Eintragung von Hyperlinks und unterscheidet sich von ähnlichen Datentypen für Hyperlinks durch die Eigenschaft *extern*. Hier wird eingetragen, ob der Hyperlink auf ein Dokument innerhalb oder außerhalb der Website verweist.

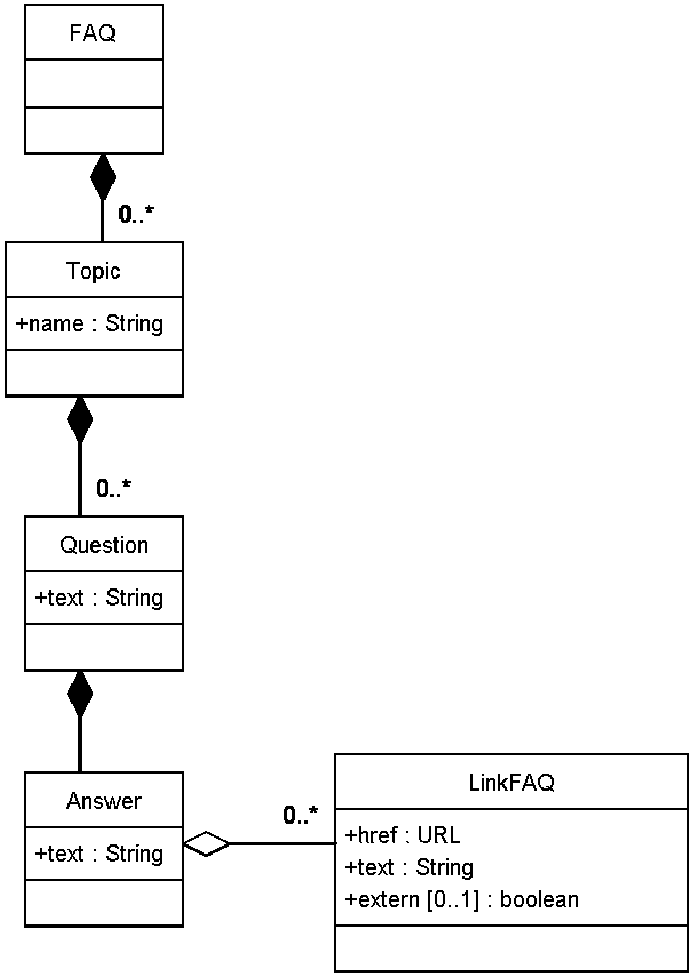

Abbildung 33: Klassendiagramm der Legacy-Webseite *FAQ*

Webseite *Change Requests*

Im Rahmen der Arbeit mit GXL und der Erstellung von darauf basierenden Werkzeugen gab es immer eine Diskussion über Änderungswünsche. Diese wurden auf eigener Webseite *Change Requests* zusammengefasst. Auch hier befinden sich die auf der Webseite veröffentlichten Daten innerhalb eines XML-Dokuments, das zur Herleitung des Datenmodells herangezogen wurde. Abbildung 34 zeigt das Ergebnis dieser Untersuchung.

Eine Änderungsanfrage wird durch den Datentyp *ChangeRequest* abgebildet und innerhalb des Datentyps *ChangeRequestList* verwaltet. *ChangeRequest* verfügt über keine Eigenschaften, sondern wird ausschließlich über seine Beziehungen zu anderen Datentypen beschrieben.

Zur Eintragung des Anfragenstellers existiert eine Beziehung zwischen ChangeRequest und *Applicant*. *Applicant* vermischt, ähnlich wie *Supplier* bei *Tool*, Informationen zu mehreren Datentypen. Einerseits werden dort Informationen zu Personen, andererseits allerdings auch zu Tagungen gespeichert. Das ist hier besonders schwerwiegend, als dass an Hand der Setzung der Eigenschaften von *Applicant* ohne nähere Betrachtung nicht festgestellt werden kann, ob es sich um eine Person oder eine Tagung handelt.

Die nächsten drei Datentypen, die in Beziehung zu *ChangeRequest* stehen, leiten sich alle von dem abstrakten Datentyp *Doc* ab. *Doc* steht für eine komplexe Struktur zur Beschreibung eines Dokuments durch Text, Tabellen, Bilder, Hyperlinks und mehr. Dieser Datentyp wird hier nicht näher beschrieben, da er lediglich ein Format für Daten darstellt. Die Datentypen

Description, *Status* und *Other* hingegen sind Spezialisierungen von *Doc* und erfüllen im Zusammenhang mit *ChangeRequest* eine bestimmte Funktion.

Durch *Description* kann eine Beschreibung der Änderungsanfrage gegeben werden, während *Status* wiedergibt, wie der Bearbeitungsstand von *ChangeRequest* ist. In dem Datentyp *Other* können Unterpunkte zu *ChangeRequest* erstellt werden, die sich nicht in die vorher genannten Kategorien einordnen lassen. Da bei *Other* der Betreff des Textes nicht bereits klar ist, kann dieser dort in der Eigenschaft *title* eingetragen werden.

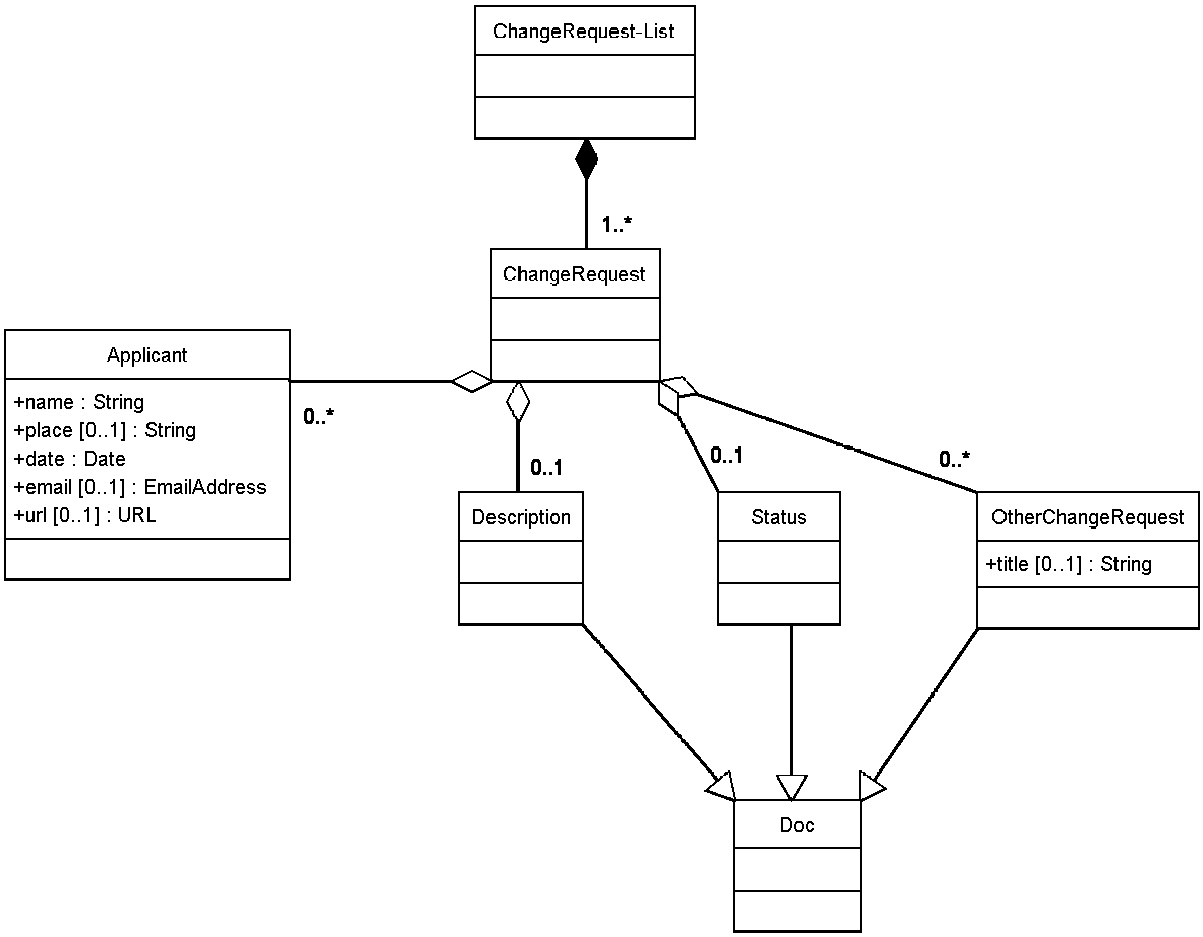

Abbildung 34: Klassendiagramm der Legacy-Webseite *Change Requests*

Sonstige Datentypen

Die restlichen Datentypen entsprechen den standardmäßigen Arten von Content, die auf vielen anderen Webseiten auch vorkommen. Dazu gehören Hyperlinks (*Link*), Email-Adressen (*EmailAddress*), Internet-Adressen (*URL*), Bilder (*Image*) und Dateien (*File*). Ihr Vorkommen kann bei vielen definierten Datentypen in den Eigenschaften nachvollzogen werden, in denen deren Bezeichnung als Datentyp der Eigenschaft auftaucht. Deshalb werden sie nicht in einem gesonderten Diagramm dargestellt und beschrieben.

3.4 Legacy-System vorbereiten

Die Aktivität *Legacy-System vorbereiten* ist neben der problemspezifischen Beschreibung und Bewertung in den vorhergegangenen Abschnitten ein essenzieller Ansatz zur Herstellung der Migrationsbereitschaft des Legacy-Systems. Die wesentlichen Workflows dieser Komponente umfassen die Sanierung des Alt-Systems und dessen Einteilung in isolierbare Migrationspakete. Denn tatsächlich lassen sich nur die wenigsten Legacy-Bestandteile in unvorbereiteter Form in das Zielsystem übertragen [Ackermann (2005), S. 187]. Wie genau

dabei vorgegangen wird, hängt von der gewählten Migrationsstrategie ab. Im vorliegenden Fall wird eine Konversion durchgeführt, bei der insbesondere die Daten in das Zielsystem übertragen werden sollen. Dabei ist deren hohe Qualität sehr wichtig. Deswegen muss eine Sanierung des Datenbestands vor allem auf korrektiver Ebene durchgeführt werden.

Im Bereich der Sanierung steht die Vorbereitung des Alt-Systems für die reibungsfreie Durchführung der Transformation im Mittelpunkt [Ackermann (2005), S. 188]. Für die vorliegende Migration trifft es, in Abwesenheit von umfangreicher Programmierung, insbesondere auf die im Legacy-System gehaltenen Daten und deren Beziehungen zueinander zu. Die *Daten* werden durch die *HTML- und XML-Dokumente* repräsentiert, deren *Beziehungen* durch die Verlinkung mit *Hyperlinks* innerhalb der Quelltexte.

3.4.1 Legacy-System sanieren

Bei der Untersuchung des Quellcodes der Webseiten, deren Daten lediglich in HTML-Dokumenten vorhanden sind, zeigte sich eine relativ geringe Qualität. Dennoch wurden die Fehler als systematisch eingestuft und sollten somit bei der Übertragung in das Zielsystem in großem Maße durch Werkzeugunterstützung zu beseitigen sein (siehe Abschnitt 3.2). Diese Einschätzung stellte sich als eingeschränkt richtig heraus. Zwar können die Fehler durch Korrekturmechanismen reduziert, allerdings nicht vollständig beseitigt werden.

Zur Sanierung des HTML-Quelltextes wird zur automatischen Verarbeitung ein geeignetes Konvertierungswerkzeug eingesetzt werden. Dieses Werkzeug soll den Quelltext in die von der Zielumgebung verwendete HTML-Version überführen (XHTML) und zudem die vorhandene Fehlerrate der Webseiten signifikant senken.

Zur Auswahl eines Werkzeugs wurden zwei Alternativen untersucht. Zum einen handelt es sich um *HTML Tidy*, einem Open Source-Produkt, das neben HTML- auch XML-Dokumente auf Validität untersucht und anpasst.[24] Zum anderen wird auf Funktionalität zurückgegriffen, die das zu Plone mitgelieferte Produkt *Kupu* leistet. Für die Bearbeitung von Content-Typen, die HTML enthalten, stellt Plone auf Ebene der Benutzungsschnittstellen standardmäßig den Editor *Kupu* zur Verfügung. Wird in der Bearbeitungssansicht HTML-Quelltext mit *Kupu* bearbeitet und abgespeichert, führt *Kupu* eine Überprüfung des HTML-Quellcodes durch und versucht, ihn in gültigen XHTML-Quellcode umzuwandeln.

Beide Konvertierungswerkzeuge wurden bezüglich ihrer Effektivität bei der Fehlerbeseitigung getestet. Dabei konnte keines der beiden Produkte eine vollständige XHTML-Validität auf allen Webseiten erreichen. Im Durchschnitt stellte sich HTML Tidy als etwas leistungsfähiger heraus. Somit wurde in Erwägung gezogen, HTML Tidy für die Überarbeitung des Quellcodes einzusetzen.

Es muss allerdings berücksichtigt werden, dass bei jeder Bearbeitung der Webseiten in Plone mit Kupu der Quellcode automatisch verändert wird. Diese Funktionalität ist grundsätzlich sinnvoll und soll auch belassen werden, denn so werden Änderungen im Quelltext direkt validiert. Das bedeutet aber ebenfalls, dass durch HTML Tidy korrigierter Quellcode durch Kupu zwangsläufig erneut verändert wird. Deshalb ist eine isolierte Betrachtung der Leistungsfähigkeit von HTML Tidy unzureichend. Es muss stattdessen noch untersucht werden, wie Kupu die Anpassungen von HTML Tidy verarbeitet. Verändert Kupu den korrigierten Quellcode in dem Maße, dass der positive Effekt von HTML Tidy aufgehoben wird, macht es keinen Sinn, Tidy als Konvertierungswerkzeug zu verwenden.

Bei Untersuchungen des Verhaltens von Kupu mit HTML Tidy wurde festgestellt, dass Kupu fast sämtliche Korrekturen von Tidy gemäß seiner eigenen Konvertierungsregeln verändert.

[24] Das Projekt HTML Tidy wird unter http://tidy.sourceforge.net/ verwaltet.

Das Resultat ist eine Webseite, die fast genauso viele Fehler hat, wie eine nur durch Kupu veränderte Webseite. Zusammengefasst bedeutet dies, dass die Bearbeitung mit Kupu von Quellcode, der mit HTML Tidy vorbereitetet wurde, keine Verbesserung in der Fehlerrate mit sich bringt, als nur mit Kupu bearbeiteter Quellcode. Eine Vorbereitung mit HTML Tidy kann deshalb entfallen.

Da im Verlauf der Testphase eine manuelle Überprüfung der Webseiten-Inhalte unerlässlich ist (vgl. Abschnitt 7), können diese Arbeitsschritte zur Validierung des Quelltexts mit Kupu (also das explizite Abspeichern in der Bearbeitungsansicht der Instanz des Content-Typs) ohne wesentlichen Mehraufwand durchgeführt werden. In einer Untersuchung der Effekte durch die Quellcode-Anpassung durch Kupu wurde festgestellt, dass der Quellcode zwar meist nicht vollständig fehlerfrei ist, aber dennoch überwiegend seine Qualität verbessert wird. Somit wird auf diese Funktionalität zur Verbesserung der Quellcode-Qualität im Zielsystem zurückgegriffen.

Bei der Umwandlung des Quelltexts handelt sich also weniger um eine Sanierung des Legacy-Systems im klassischen Sinne, sondern um eine Anpassung der überführten Legacy-Daten im Zielsystem. Es stellt dennoch eine Sanierung dar, da es irrelevant ist, zu welchem Zeitpunkt sie durchgeführt wird. Zudem ist die niedrigere Qualität des Quellcodes kein Hindernis, die Daten des Legacy-Systems nach Plone zu transferieren.

3.4.2 Migrationspakete bilden

Bei der Betrachtung der Zerlegung der Legacy-Website in Pakete müssen die Beziehungen zwischen den einzelnen Webseiten berücksichtigt werden, die in HTML als Hyperlinks realisiert sind. Die gesamte Website besteht aus einem Netz von Hyperlinks, bei denen eine Seite auf den Inhalt einer anderen verweist.

Ein zusammengehöriges Paket muss über die Eigenschaft verfügen, dass die darin enthaltenen Hyperlinks möglichst nur auf Webseiten innerhalb des Pakets verweisen. Eventuell vorhandene Verweise von externen Webseiten auf solche des Pakets sind bei der Bildung logischer Einheiten nicht relevant. Deshalb finden sich diese weder bei den Paketbeschreibungen noch bei deren Abbildungen. Das hierbei gewählte Vorgehen lässt sich mit dem Slicing von Programmen im Software-Reengineering vergleichen.

In Abschnitt 3.3 wurden in Abbildung 25 bereits die Zusammenhänge in der Navigation zwischen den einzelnen Webseiten in einem Navigationsnetz vorgestellt. Ausgehend von den dort beschriebenen Abhängigkeiten, werden die Pakete gebildet. Für die Zusammenfassung der Webseiten in Pakete gibt es mehrere Lösungen. Bei der gewählten Einteilung der Pakete wurde darauf geachtet, dass die Übersichtlichkeit und der thematische Zusammenhang erhalten bleiben. Die gebildeten Pakete sind wie folgt:

- Paket *Examples*
- Paket *Tool Catalogue*
- Paket *DTD*
- Paket *Background*
- Paket *Advance*
- Paket *Introduction*
- Paket *FAQ*

Es wurde bereits in den Abschnitten 3.2.2 und 3.3 erläutert, dass die Inhalte des Navigationsbereichs *Advance* nicht zu migrieren sind. Bei der Bildung der Migrationspakete ist es wichtig, die verbleibenden Inhalte der Webseite so zu Modularisieren, dass möglichst wenige Verweise auf diesen Navigationsbereich verbleiben. Gelingt es, *Advance* als von den

restlichen Inhalten unabhängiges Paket darzustellen, kann dieser Bereich gefahrlos von der Transformation ausgenommen werden.

Paket Examples
Der Bereich rund um die Beispiele zu GXL wird in dem Paket *Examples* zusammengefasst. Bei den betroffenen Webseiten verlinkt *Examples* auf *ExamplesMetaschema*. Ansonsten wird nur von außerhalb auf das Paket, genauer gesagt auf *Examples*, verwiesen. Abbildung 35 zeigt die Navigationsstruktur des Pakets *Examples*.

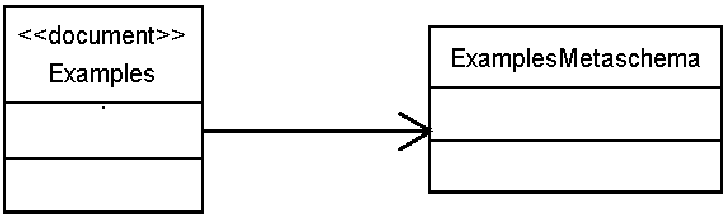

Abbildung 35: Navigationsstruktur des Pakets *Examples*

Paket Tool Catalogue
Das Paket *Tool Catalogue* umfasst lediglich die isolierte Webseite *Tool Catalogue*. Auf diese Webseite existieren zwar Verweise, sie selbst verlinkt allerdings auf keine andere Webseite innerhalb der Legacy-Website. Anders als bei *Future* kann hier kein direkter thematischer Zusammenhang mit anderen Webseiten gefunden werden. Da die Webseite *Tool Catalogue* selbst komplex ist, wird sie als eigenes Paket definiert. Abbildung 36 zeigt die Navigationsstruktur dieses Pakets. Da es aus einer einzelnen Webseite besteht und diese nicht auf weitere Webseiten verweist, besitzt sie keine Navigationslinks.

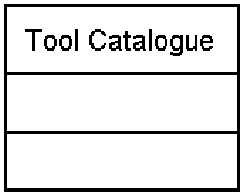

Abbildung 36: Navigationsstruktur des Pakets *Tool Catalogue*

Paket DTD
Zur Zusammenfassung der grundlegenden Informationen rund um die Beschreibung von GXL in Form von Beispielen sowie Bildern und Quelltexten wurde das Paket *DTD* definiert. Das zentrale Element ist die Webseite *DTD*. Sie ist ein wichtiger Referenzpunkt der Webseiten *Graph Model*, *XML-Schema* und *Metaschema*, auf die sie teilweise selbst wieder zurückverlinkt. Von *Metaschema* gibt es einen Verweis auf die *Examples*-Webseiten des Pakets *Examples*. Die Navigationsstruktur des Pakets *DTD* ist in Abbildung 37 nachzuvollziehen.

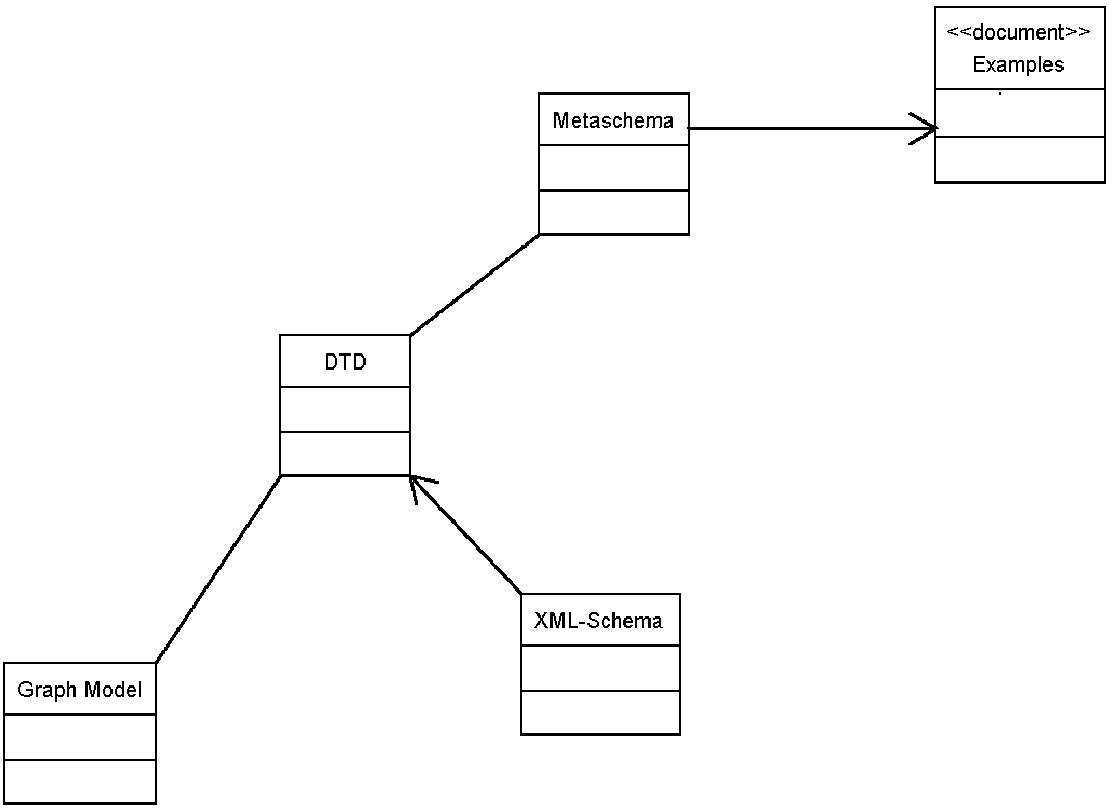

Abbildung 37: Navigationsstruktur des Pakets *DTD*

Paket Background

Die Webseite *Background* ist das zentrale Element des Pakets *Background*. Von dort gehen Verweise auf die Webseiten *Publications* und *DTD* aus, wobei die Webseite *DTD* wiederum ein Bestandteil des Pakets *DTD* ist. Der thematische Zusammenhang lässt sich durch den Zweck der Webseite *Background* erklären, da von dort aus erste Verweise auf die Struktur von GXL (in Form der Webseite *DTD*) und vorhandene Literatur zu GXL in der Webseite *Publications* ausgehen. Abbildung 38 verdeutlicht die Navigationsstruktur des Pakets *Background*.

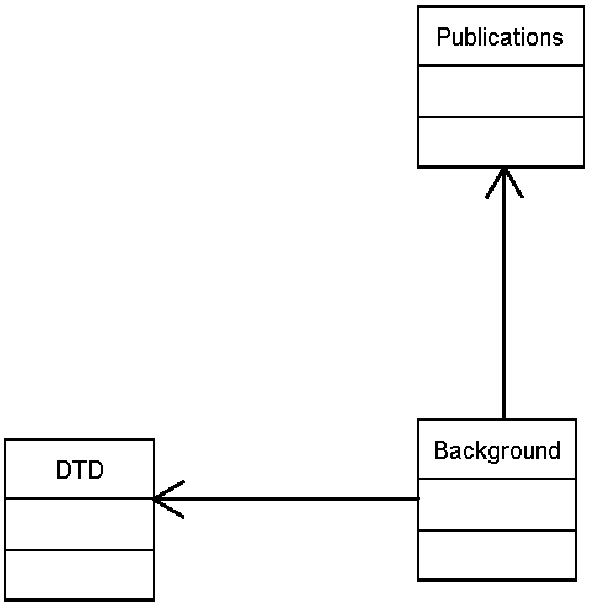

Abbildung 38: Navigationsstruktur des Pakets *Background*

Paket Advance
Das Paket *Advance* fasst die Webseiten zusammen, die sich in dem Navigationsbereich *Advance* befinden. Die Webseite *GXL 1.1* referenziert auf die Webseiten *Graph Model* und *Metaschema* und stellt damit eine Verbindung zur vorhergehenden Version von GXL her. Die Änderungsanfragen, die in der Version GXL 1.1 berücksichtigt wurden, sind durch die Verlinkung von der Webseite *GXL 1.1* zur Webseite *Change Requests* dargestellt.

Auf die Webseite *Change Request* selbst wird durch *Future* verwiesen. Innerhalb der Änderungsanfragen wurden Verweise zu Beispielen gemacht, die sich auf den *Examples*-Webseiten befinden. Dadurch und durch die Referenzen der Webseite GXL 1.1 bestehen vom Paket *Advance* aus Abhängigkeiten zu den Paketen *Examples* und *DTD*. Die Navigationsstruktur des Pakets *Advance* ist in Abbildung 39 dargestellt.

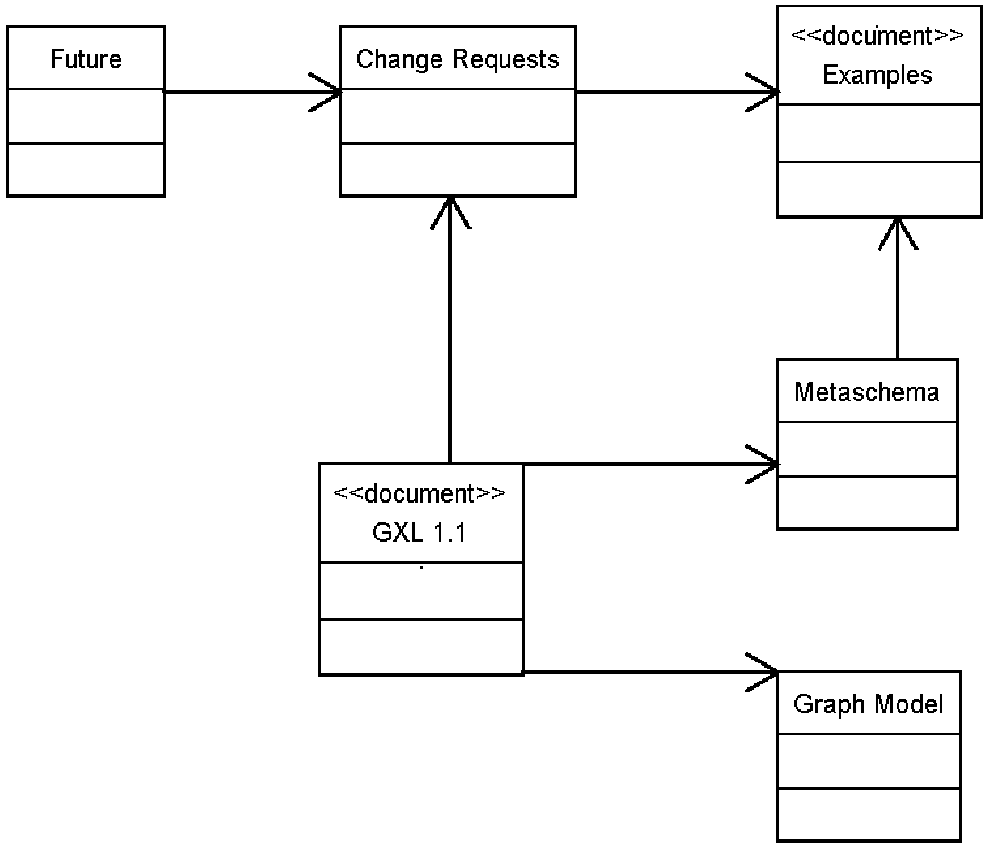

Abbildung 39: Navigationsstruktur des Pakets *Advance*

Paket Introduction
Die in dem einführenden Artikel zum Thema GXL enthaltenen Dokumente sind im Paket *Introduction* erfasst. Innerhalb der Webseiten des Artikels existiert eine Vielzahl von lokalen Navigationsreferenzen. Zudem wird zur Erläuterung des Inhalts auf einzelnen Webseiten des Artikels auf die Webseiten *DTD* und *Examples* verwiesen, die zu den Paketen *DTD* und *Examples* gehören. Abbildung 40 zeigt die Navigationsstruktur des Pakets *Introduction*.

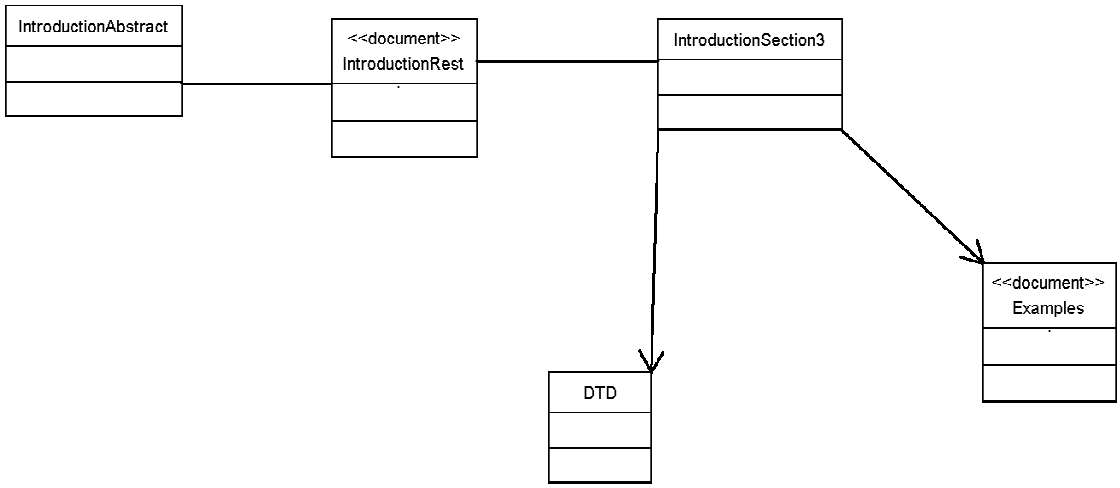

Abbildung 40: Navigationsstruktur des Pakets *Introduction*

Paket FAQ

Das Paket *FAQ* verfügt über besonders viele Verweise auf Webseiten, die bereits in anderen Paketen zusammengefasst wurden. Auf der Legacy-Website bestehen keine Hyperlinks von anderen Webseiten auf *FAQ*. Die von der Webseite *FAQ* referenzierten Webseiten sind Bestandteile fast aller bereits beschriebenen Pakete, bis auf das Paket *Advance*. Die komplexe Navigationsstruktur des Pakets *FAQ* ist in Abbildung 41 dargestellt.

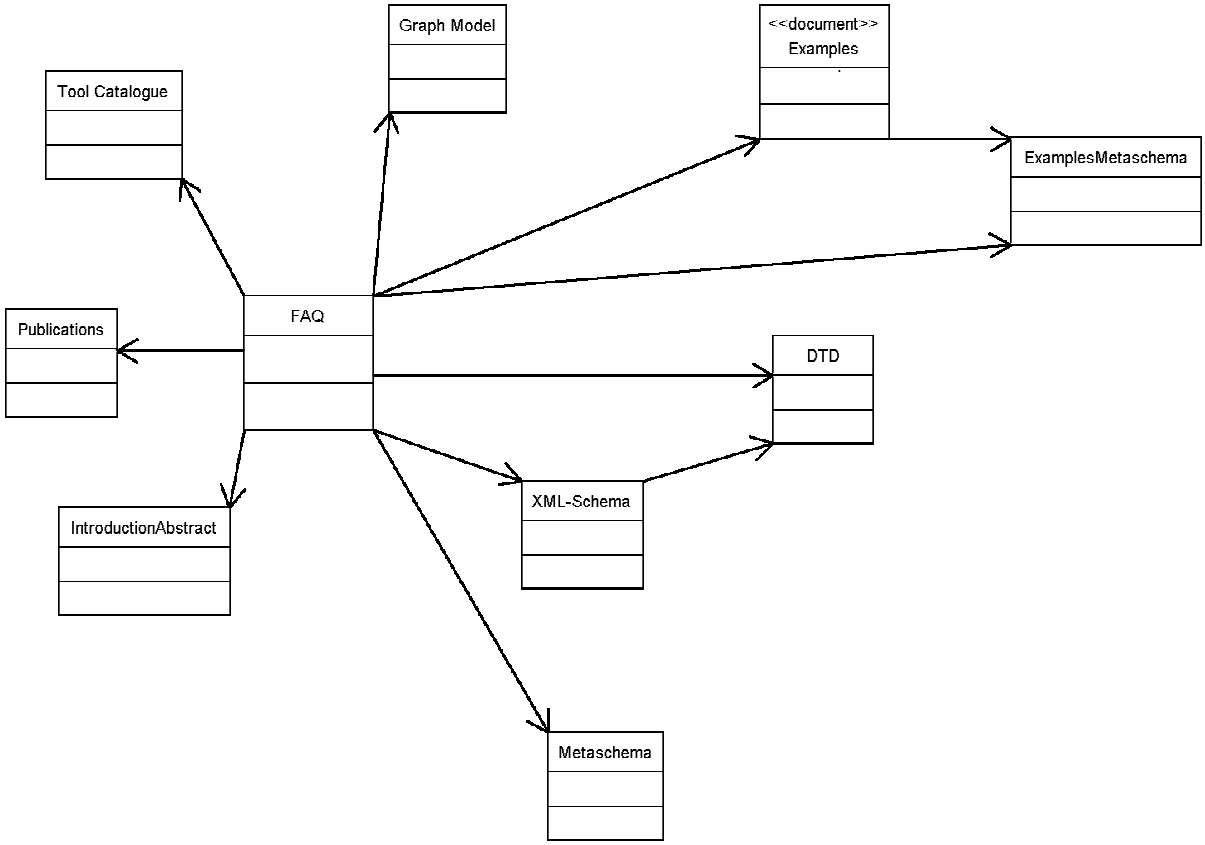

Abbildung 41: Navigationsstruktur des Pakets *FAQ*

Abhängigkeiten zwischen den Paketen

Bisher konnten aus der Navigationsstruktur der zu migrierenden Website zahlreiche Webseiten als Pakete isoliert werden. Dabei wurde der Fokus auf die Elemente der Pakete und deren Abhängigkeit zu anderen Webseiten gelegt. Für die Planung der Migration ist

allerdings auch die Betrachtung der Abhängigkeiten auf dem Niveau der Pakete wichtig. So kann z.B. eine Abhängigkeitshierarchie abgeleitet werden, die es ermöglicht, eine Reihenfolge der zu migrierenden Pakete festzulegen.

Die Pakete interagieren sehr unterschiedlich miteinander. Kandidaten für eine frühzeitige Migration sind diejenigen Pakete, die nur von anderen aufgerufen werden, selbst aber über keine weiteren Abhängigkeiten verfügen. Dazu gehören die Pakete *Tool Catalogue* und *Examples*. Sie sind Migrationspakete der Rangordnung 1.

Im weiteren Vorgehen können Pakete betrachtet werden, die zwar Abhängigkeiten zu den oben genannten besitzen, ansonsten aber keine weiteren referenzieren. Diese Pakete der Rangordnung 2 können migriert werden, sobald die Migration aller Pakete der Rangordnung 1 durchgeführt wurden. Zu dieser Gruppe gehört das Paket *DTD*. Analog wird bei der Bestimmung weiteren Rangordnungsstufen verfahren. Pakete der Rangordnung 3 sind *Background* und *Introduction*, bis zuletzt FAQ und *Advance* Pakete der Rangordnung 4 sind. Abbildung 42 stellt die Abhängigkeiten zwischen den Paketen der zu migrierenden Website dar und hebt die Pakete einer Rangordnung durch Schattierung hervor. Der Stereotyp <<*call*>> steht für einen Aufruf einer Webseite eines Pakets durch einen Hyperlink. Die Schattierung der Pakete stellt die Zugehörigkeit zu einer Rangordnung dar. Dunkel steht für Stufe 1, dunkelgrau für 2, hellgrau für 3 und weiß für Stufe 4.

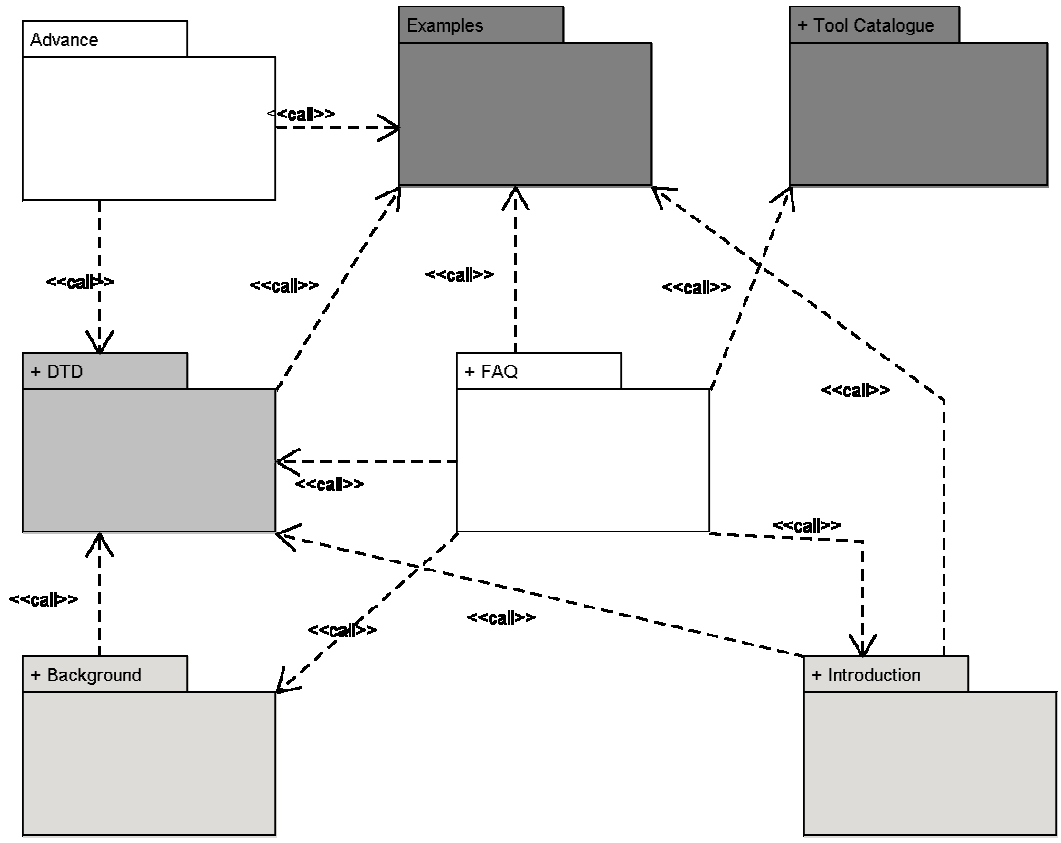

Abbildung 42: Abhängigkeiten zwischen den Paketen der Legacy-Website

Nachdem die einzelnen zusammenhängenden Bereiche der Website in Pakete zusammengefasst wurden, sind durch Hyperlinks bedingte Abhängigkeiten deutlich geworden. Für die Migration sind die hier gewonnenen Erkenntnisse und die Ableitung einer Rangordnung zwischen den Paketen bedeutsam. Bei der Transformation der Website in das Zielsystem müssen auch die Verbindungen zwischen den Webseiten erhalten bleiben. Um dies zu gewährleisten, werden die Pakete gemäß ihrer Rangfolge migriert und die Hyperlinks entsprechend aktualisiert.

Besonders wichtig bei dieser Untersuchung war das Ergebnis, dass der Navigationsbereich *Advance*, der durch das Paket *Advance* repräsentiert wurde, als isoliertes Paket darstellbar ist. Es existieren keine Verweise von den restlichen Webseiten der Website auf diesen Bereich und er kann somit ohne Gefahr inkonsistenter Links aus der Transformation genommen werden.

4 Ziel-Design

Bereits vor Planung der Migration der GXL-Website wurde Plone als zukünftiges Content Management System des Internetauftritts der Universität Koblenz-Landau ausgewählt. Da schließlich auch die GXL-Seiten in die Website der Universität zu integrieren sind, wird die Migration nach Plone notwendig. Eingangs wurden im Abschnitt 1.4 bereits die Begriffe Content und Content Management erläutert. Im Anschluss darauf stellt die Beschreibung der neuen Zielumgebung, des Content Management Systems Plone, ausführlich die gegebenen Technologien und Möglichkeiten dar. Daraus werden sich Handlungsanweisungen für die durchzuführende Migration ableiten lassen.

Im ReMiP erfüllt der Kernbereich des Ziel-Designs die Funktion der „schrittweisen Entwicklung und Verfeinerung des Entwurfs von Zielsystems und Zielumgebung“, unter Berücksichtigung der spezifizierten Anforderungen [Ackermann (2005), S. 189]. Daraus folgt, dass unter intensiver Berücksichtigung der Erkenntnisse der Legacy-Analyse das Software-System auf die gewünschten und existierenden technologischen Möglichkeiten abgebildet wird. Die hierbei zu tätigenden Aktivitäten sind in den folgenden Abschnitten beschrieben und deren Anwendung auf die Migration der Website diskutiert.

4.1 Zielsystem-Kandidaten definieren

Die nach Vorgabe des ReMiP hier notwendige ausführliche Betrachtung verschiedener Zielsystem-Kandidaten auf Ebene verschiedener *Technologien* entfällt. Dies ist darauf zurückzuführen, dass diese Migration durch die bevorstehende Umstellung der gesamten Website der Universität Koblenz-Landau in das Zielsystem Plone notwendig wurde. Alternative Zielsysteme oder ein Verbleiben bei der aktuellen Lösung kommen nicht in Frage. Somit ist die potenzielle Zieltechnologie bereits auf die Nutzung von Plone festgelegt. Auf welchen Grundlagen diese Technologie aufbaut, wurde im Abschnitt 1.5 beschrieben.

4.2 Zielsystem-Architektur verfeinern

In diesem Bereich der Beschreibung des Ziel-Designs werden durch schrittweise Verfeinerung der Systemarchitektur die Grundlagen für das Design der Programme, Datenbanken und Benutzungsschnittstellen geliefert [Ackermann (2005), S. 195 / 196]. Bereits im Abschnitt 3 wurden die Datentypen der zu migrierenden Website als Klassendiagramm im Vergleich zu den einzelnen Webseiten abstrakt dargestellt. Hierbei wurde eine bewusst datenzentrierte Sicht der Website angestrebt.

Im Einführungsabschnitt 1.4.2 wurden Content-Typen als wesentliche Bestandteile eines Content Management Systems erläutert. Plone selbst baut intensiv auf der Verwendung von Content-Typen auf. Bei genauer Betrachtung lassen sich die im Klassendiagramm als eigene Datentypen identifizierten Klassen problemlos auf das Konzept von Content-Typen übertragen. Eine Klasse entspricht dabei einem Content-Typ und die in ihr beschriebenen Attribute lassen sich als Eigenschaften eines Content-Typs abbilden. Auf diese Weise kann unter Einhaltung des Grundprinzips der Migration bezüglich der Unveränderlichkeit der fachlichen Funktionalität [Ackermann (2005), S. 39] durch weitestgehend unveränderte Übertragung des Datenmodells aus Abschnitt 3.3 in die Systematik der Content-Typen die

erste Beschreibung des Zieldesigns durchgeführt werden. Änderungen des Designs sind lediglich in den Fällen notwendig, wo das Datenmodell der zu migrierenden Website unvollständig ist.

4.2.1 Aufzählung der Content-Typen

In diesem Abschnitt wird die Grundarbeit für weitere Design-spezifische Entscheidungen geleistet. Zunächst muss bestimmt werden, inwiefern Datentypen aus der Legacy-Analyse in Content-Typen des Zielsystems überführt werden können und an welcher Stelle Bedarf für die Einbeziehung weiterer Content-Typen besteht.

Bei der Übertragung der Datentypen aus der Legacy-Analyse in das Konzept der Content-Typen im Zielsystem, lassen sich *drei Gruppen* von Content-Typen einteilen:

- Content-Typen, die aus der Legacy-Analyse übernommen werden
- Content-Typen, die nicht aus der Legacy-Analyse übertragen werden
- Content-Typen, die in der Legacy-Analyse noch nicht erkannt wurden und neu hinzukommen

Zunächst werden die Content-Typen behandelt, die aus einzelnen oder mehreren Datentypen gebildet werden können, die bereits in der Legacy-Analyse identifiziert wurden. Diese werden folgend aufgezählt.

- *Organisation*
- *OrganisationalUnit*
- *Project*
- *Conference*
- *Person*
- *FAQ*
- *Example*
- *Tool*
- *Listing*
- *WebArticle*
- *Publications*
- *InProceedings*
- *TechReport*
- *Image*
- *Link*
- *File*
- *Document*

Die letzten vier Einträge der Liste, *Image*, *Link*, *File* und *Document* verweisen auf Content-Typen, die bereits standardmäßig im Zielsystem Plone implementiert sind. Unter den hier genannten Content-Typen befinden sich einige, die in der Legacy-Analyse wesentlich komplexere Entsprechungen hatten. Diese Content-Typen, *FAQ* und *Tool,* waren im Alt-System durch DTDs definiert. Außerdem wurden die Datentypen rund um die Webseite *Publications* erheblich modifiziert, um sie auf das Zielsystem abzubilden. In diesen Bereich gehören *Publications*, *InProceedings* und *TechReport*.

Die zweite Gruppe von Datentypen wurden zwar in der Legacy-Analyse beschrieben, werden aber aus Gründen der mangelnden Aktualität der Inhalte aus der Migration herausgenommen. Hierbei handelt es sich um die Datentypen, die lediglich im Navigationsbereich *Advance* vorkamen, nämlich rund um die Webseite *Change Request*.

In der dritten Gruppe befinden sich Content-Typen, die im Datenmodell des Legacy-Designs nicht als Datentypen erkannt wurden, aber in Hinsicht auf das Zielsystem wichtig sind. Dabei handelt es sich überwiegend um *Container*, wie sie bereits in Abschnitt 1.5.2 kurz beschrieben wurden. Diese Content-Typen beinhalten alle Instanzen einer bestimmten Art von Content-Typen. Von diesen Typen aus können schließlich die einzelnen enthaltenen Instanzen angesteuert werden. Zur Unterscheidung der Container von den normalen Content-Typen wurde den Namen die Erweiterung *Set* angehängt. Die hinzugefügten Content-Typen lauten wie folgt:

- *OrganisationSet*
- *PersonSet*
- *ConferenceSet*
- *FAQSet*
- *ExampleSet*
- *ToolSet*
- *FolderHTML*
- *Topic*
- *GXLFolder*

Nicht als eigene Content-Typen wurden die Datentypen *URL* und *EmailAddress* modelliert. Gründe für diese Entscheidung ist die Berücksichtigung der internen Struktur von Plone. Die *URL* und *EmailAddress* wurden vor allem zur Beschreibung des besonderen Formats des darin enthaltenen Texts definiert. URLs und Email-Adressen weisen eine bestimmte Syntax auf, die über die Verwendung von Validatoren überprüft werden kann. Da Plone die Verwendung von solchen Validatoren zur Überprüfung von Feldeinträgen unterstützt und zudem bereits Validatoren für URLs und Email-Adressen in Plone installiert sind, wird auch hier die Definition eines eigenen Content-Typs überflüssig. Es reicht aus, die Eigenschaftsfelder der Content-Typen, die auf URLs bzw. Email-Adressen verweisen, mit dem geeigneten Validator zu versehen.

Zur Kategorisierung von Content wird das Metadaten-Feld *keyword* verwendet. Der Zweck dieses Felds ist die Ermöglichung der Gruppierung und Sortierung von Elementen nach dem zugewiesenen Schlüsselwort [McKay / da Silva (2006), S. 50]. Damit kann die gewünschte Funktionalität für die Content-Typen *FAQ*, *Example* und *Tool* ohne eigene Konstruktionen, wie im Ist-Datenmodell (siehe Abschnitt 3.3.6), implementiert werden.

4.2.2 Allgemeines zur Beschreibung der einzelnen Content-Typen

Die kurze Auflistung der Content-Typen, die für das Ziel-Design gewonnen werden konnten, gibt nur einen ersten groben Einblick in das Ziel-Design. In den folgenden Abschnitten wird dieses gemäß der Anforderungen des ReMiP schrittweise verfeinert. Als nächste Detaillierungsstufe des Designs werden nun die Content-Typen jeweils nach ihren Eigenschaften beschrieben. Ziel dieses Vorgehens ist es, die innere Struktur der Content-Typen des Zielsystems transparent zu machen und die in ihnen enthaltenen Eigenschaften auf implementierungsnaher Ebene zu beschreiben.

Zusammenhängende Content-Typen werden sowohl textuell als auch in einem Klassendiagramm beschrieben. Hierbei wird die von ArchGenXML vorgegebene Modellierung verwendet. Content-Typen sind als Klassen und Referenzen zwischen Content-Typen als gerichtete Assoziationen dargestellt. Container-Content-Typen, die über die Eigenschaft *addableContent* verfügen, können die dort angegebenen Content-Typen einfügen. In den Klassendiagrammen wird diese Beziehung durch eine Aggregation dargestellt. Eine striktere Einfügeeigenschaft wird durch *addableContentExclusive* gekennzeichnet. Die dort eingetragenen Content-Typen können nur innerhalb dieses Containers eingefügt werden.

Diese Container werden in den Klassendiagrammen durch eine Komposition mit den einfügbaren Content-Typen verbunden.

Um die Content-Typen in einem ausreichend detailliertem Maß zu spezifizieren, werden deren Eigenschaften vollständig aufgezählt und erläutert. Hierzu werden neben deren Namen auch die *Typen* der Felder eingetragen. Bei der Beschreibung der Feldtypen werden Bezeichnungen übernommen, die im Zielsystem Plone existieren. Folgend eine Liste der verwendeten Feldtypen:

- *string*: zur Eingabe kurzer Zeichenketten ohne Zeilenumbruch
- *text*: zur Eingabe von längeren Texten im Plain Text-Format ohne Zeilenumbruch
- *richtext*: zur Eingabe von längeren Texten in HTML-Format
- *date*: zur Eingabe eines Datums
- *computed*: zur Ausweisung eines berechneten Felds

Neben den hier genannten Feldtypen können die Eigenschaften auch auf Instanzen anderer Content-Typen referenzieren, wie z.B. auf Bilder oder Dateien. In solchen Fällen steht als Feldtyp der Name des Content-Typs. Es handelt sich in allen Fällen um Referenzen. Die Werte des jeweiligen Feldes enthalten also nicht Instanzen dieser Content-Typen, sondern verweisen lediglich auf diese. Im Unterschied zu den vorgenannten Feldtypen können Referenzen auf andere Content-Typen leicht an der Großschreibung erkannt werden.

Zudem werden für jedes Feld die *Multiplizitäten* festgelegt. Diese geben die Unter- und Obergrenze der Anzahl der unter einer Eigenschaft zu speichernden Einträge an [Jeckle et al. (2004), S. 47]. Als Notation wird die abgewandelte Darstellung in einer Intervallklammer gewählt. Grundsätzlich kann jede beliebige Zahl in einem sinnvollen Zusammenhang als Unter- und Obergrenze eingetragen werden. Üblicherweise wird aber die Multiplizität einer Eigenschaft lediglich mittels eines der folgenden Beispiele angegeben.

- [0,1]: eine optionale Eigenschaft; sie kann höchsten einmal vorkommen
- [1,1]: eine obligatorische Eigenschaft; sie muss genau einmal vorkommen
- [0,*]: eine optional beliebige Eigenschaft; sie kommt mindestens null mal vor
- [1,*]: eine beliebige Eigenschaft; sie kommt mindestens einmal vor

Neben den Eigenschaften mit ihren Datentypen und Multiplizitäten werden auch zugewiesene Validatoren notiert. Dies ist insbesondere der Fall für die Prüfung auf Einhaltung der Syntax von Email- und Internet-Adressen. Bei den Eigenschaften werden die Validatoren als Prädikat eines Datentyps geschrieben, wie das aus dem *Organisation*-Content-Typ entnommene Beispiel *websiteURL: isURL(string), [0,1]*. Hier legt das Prädikat *isURL()* fest, dass darin eingegebene Zeichenketten für das Feld *websiteURL* auf URL-Syntax geprüft werden. Alle in der Beschreibung der Content-Typen verwendeten Validatoren sind:

- *isURL()*: Syntaxprüfung für URLs
- *isEmail()*: Syntaxprüfung für Email-Adressen
- *isInternationalPhoneNumber()*: Syntaxprüfung für internationale Telefonnummern
- *isTidyHtmlWithCleanup()*: Syntaxprüfung für HTML-Code und automatische Korrektur gefundener Fehler.

Bei der Belegung der Eigenschaften der Content-Typen kommen auch die in Abschnitt 1.5.3 besprochenen Metadaten in Frage. Sie enthalten Daten, die auf jedes Content-Objekt innerhalb von Plone angewendet werden. Eine besondere Rolle bei den Metadaten nehmen die Eigenschaften *title* und *description* ein. Sie werden bei allen für die Ziel-Website entworfenen Content-Typen sogar in der standardmäßigen Bearbeitungssicht, statt in der Eigenschaftensicht, angezeigt. Der Grund für ihre Verwendung liegt in der Abstimmung mit dem Zielsystem. In Plone werden diese Eigenschaften für spezielle Anzeigeoptionen verwendet. So wird *title* zur Füllung des *title*-Elements einer Webseite genutzt, dessen Inhalt

z.B. in vielen Browsern als Text in der Titelleiste erscheint. Zudem wird der Titel eines Dokuments in zusammenfassenden Sichten auf das Dokument, wie z.B. in Suchabfragen in Plone oder Internet-Suchmaschinen sowie in der Ordneransicht von Plone, als Link zum jeweiligen Dokument angezeigt. Ebenso erscheint der Inhalt der Eigenschaft *description* bei zusammenfassenden Sichten auf Dokumente in Plone. Sie dient hierbei als kurze Beschreibung des Dokuments und sollte etwa 20 Wörter enthalten [McKay / daSilva (2005), S. 47]. Außerdem verwendet Plone den *description*-Text zur Füllung des *alt*-Attributs des *a*-Elements. Dieses Attribut wird z.B. angezeigt, wenn sich der Mauszeiger über einem Hyperlink befindet und dient zu seiner kontextuellen Beschreibung. Die *description*-Eigenschaft ist, im Gegensatz zu *title*, nicht obligatorisch, da es nur zusätzliche Informationen enthält. Dennoch ist seine Belegung zweckmäßig. In allen nachfolgenden Content-Typen sind diese Eigenschaften *nicht explizit modelliert und beschrieben*.

Ähnlich verhält es sich mit dem Metadatenfeld *keyword*. Zwar wird es bei Content-Typen, die mit *keyword* kategorisiert werden sollen, explizit beschrieben, allerdings wird es bei der Modellierung als Klassendiagramm als Metadatum ignoriert.

4.2.3 Content-Typ *FolderHTML*

Für die Erstellung der Zielwebsite wird in vielen Fällen auf Ordner zurückgegriffen. Für die Strukturierung der Navigationsbereiche sind sie sehr nützlich, da sie die Ordnerstruktur der Legacy-Webseite nachempfinden. Es existieren aber auch viele Content-Typen, die selbst Spezialisierungen von Ordnern sind. Hierzu gehören die Container-Content-Typen.

Plone stellt standardmäßig den Content-Typ *Folder* zur Verfügung. Dabei handelt es sich um einen recht einfachen Ordner, der lediglich aus den Feldern *title* und *description* besteht. Als nachteilig stellt sich heraus, dass in das Feld *description* zur Eintragung einer Beschreibung nur einfacher Text eingefügt werden kann, nicht aber HTML-Code mit Hyperlinks und eigener Formatierung. Damit wäre die Verwendung von *Folder* in der Regel nicht zufrieden stellend. Zur Erweiterung der Funktionalität von *Folder* wurde der Content-Typ *FolderHTML* entworfen. Seine Felder sind nachfolgend beschrieben:

- *name*: string, [1,1]
- *descriptionHTML*: isTidyHtmlWithCleanup(richtext), [0,1]
- <<ordered>>

Das Feld *name* dient zur Eintragung eines bezeichnenden Namens für den Ordner. Es wurde eingeführt, um eine Differenzierung bei *name* und *title* zu ermöglichen. Zusätzlich wurde das Feld *descriptionHTML* modelliert, das für die Eingabe von beschreibendem Text in HTML-Format vorgesehen ist. Zudem wurde *FolderHTML* der Stereotyp <<*ordered*>> zugewiesen, das bewirkt, dass dieser Content-Typ als ein Ordner mit Elementen definiert ist, die man in eine Reihenfolge bringen kann.

Auf *FolderHTML* wird bei vielen Container-Content-Typen in einer Spezialisierung zurückgegriffen. Die Eigenschaften werden für jeden Container nochmals beschrieben. In Abbildung 43 kann dann die Vererbungshierarchie zwischen den Containern und *FolderHTML* nachvollzogen werden. Die dort ebenfalls modellierten Content-Typen werden weiter unten noch detailliert beschrieben.

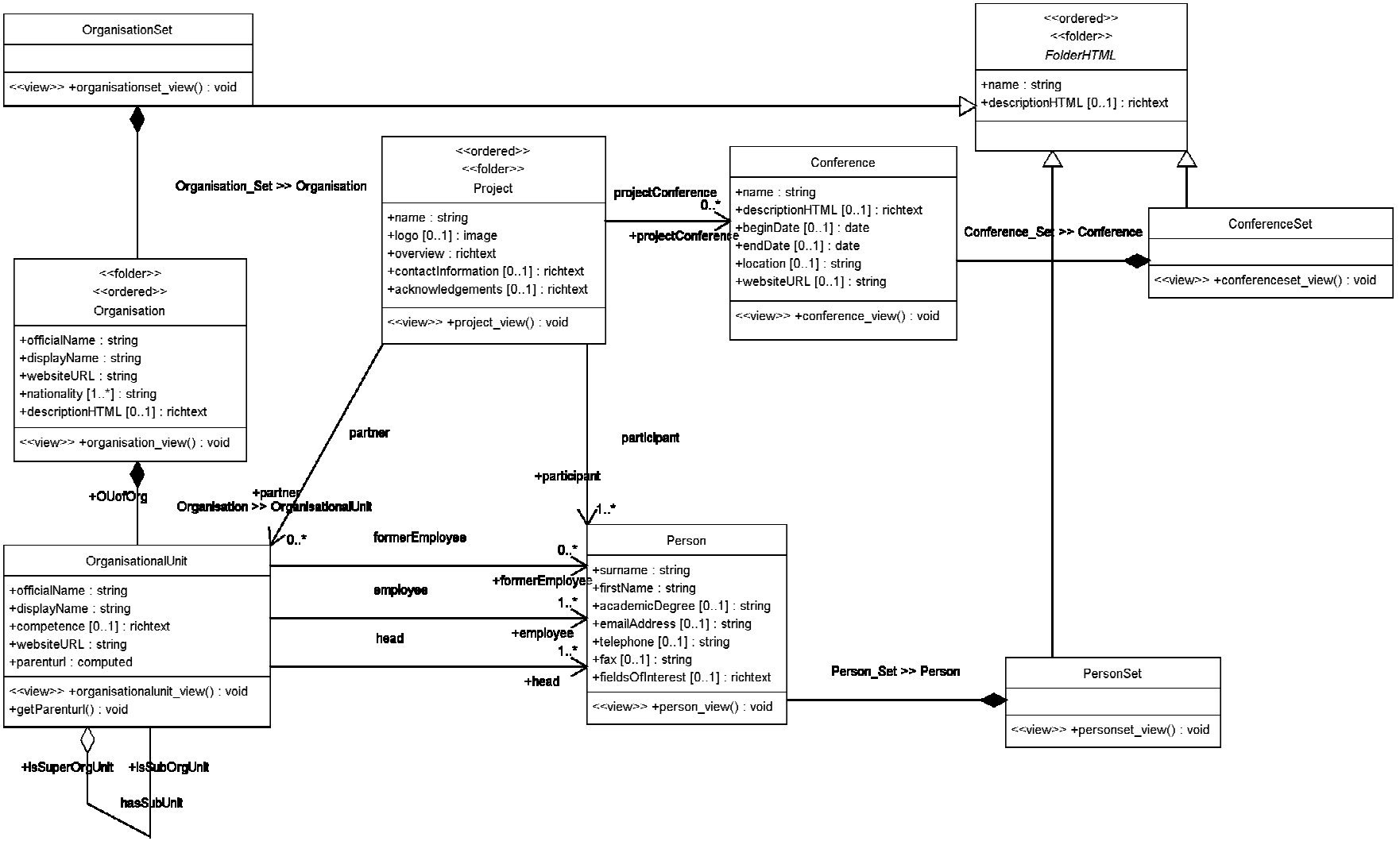

Abbildung 43: Datenmodell zu *Project*, *Organisation*, *OrganisationalUnit*, *Person*, *PersonSet*, *Conference*, *ConferenceSet* und *FolderHTML*

Auf der zu migrierenden Website werden in vielen Fällen Verweise auf Organisationen und Institutionen gemacht, wie z.B. Unternehmen mit ihren Abteilungen wie das Nokia Research Center mit dem Software Technology Laboratory oder Universitäten mit ihren Arbeitsgruppen, wie die Universität Koblenz mit der GUPRO-Gruppe. Es werden stets bestimmte Informationen in verschiedenen Kontexten zu der Organisation oder Untereinheit angezeigt. Beim Entwurf des Zielsystems werden zur Modellierung dieser Zusammenhänge zwei Content-Typen unterschieden, *Organisation* und *OrganisationalUnit*.

4.2.4 Content-Typ *Organisation*

Organisation enthält allgemeine Daten, die eine Organisation, sei es nun ein Unternehmen oder eine Universität, beschreiben. Dabei sind die Daten relativ abstrakt. Genauere Informationen befinden sich im Content-Typ *OrganisationalUnit* (siehe weiter unten). In diesem Datenmodell ist es nicht vorgesehen, dass Organisationen direkte Verknüpfungen zu anderen Objekten haben. Diese werden über Instanzen von *OrganisationalUnit* hergestellt. Die Eigenschaften des *Organisation*-Content-Typs sind nachfolgend aufgelistet:

- *officialName*: string, [1,1]
- *displayName*: string, [1,1]
- *descriptionHTML*: isTidyHtmlWithCleanup(richtext), [0,1]
- *nationality*: string, [1,*]
- *websiteURL*: isURL(string), [0,1]
- (*addableContent:* OrganisationalUnit)

Wie hier ersichtlich, handelt es sich bei den Eigenschaften vor allem um Kontaktdaten. Der offizielle Name der Organisation kann in das Feld *officialName* eingetragen werden. Da es aber sein kann, dass der offizielle Name etwas länglich wird und in der Anzeige in Webseiten zu viel Platz verbraucht, wird auch die Eingabe eines Namens *displayName* verlangt, der die Bezeichnung der Organisation auf Kurzform reduziert. Das Feld *descriptionHTML* ermöglicht die Eingabe von beschreibenden Informationen zu der Organisation in HTML-Format. Auf der zu migrierenden Website wurde häufig auf die Zugehörigkeit der Organisation zu einem Staat verwiesen. Um dies auch in der Ziel-Website zu realisieren, ist es möglich die Nationalität der Organisation als Zeichenkette im Feld *nationality* einzutragen. Dieses Feld wurde als beliebig modelliert, damit multinationale Organisationen mehrere Nationalitäten eintragen können. Ebenfalls sehr wichtig ist die Angabe der Website der jeweiligen Organisation. Dies ist mit dem Eigenschaftsfeld *websiteURL* möglich, das die Eingabe einer Zeichenfolge nur dann akzeptiert, wenn es die Überprüfung auf Validität als URL besteht. Damit wird die Möglichkeit einer falschen Internet-Adresseingabe vermindert. Da eine Organisation in der Regel aus Organisationseinheiten besteht (Beschreibung dazu weiter unten), kann über die Eigenschaft *addableContent* eine Organisationseinheit *OrganisationalUnit* der Organisation hinzugefügt werden.

4.2.5 Content-Typ *OrganisationalUnit*

OrganisationalUnit ist eng verbunden mit dem zuvor beschriebenen Content-Typ *Organisation*. Es bezeichnet eine Organisationseinheit, die in einem hierarchischen Verhältnis zu einer Organisation gehört. Dies umfasst sowohl Abteilungen in Unternehmen als auch Institute und Arbeitsgruppen einer Universität. Auf der zu migrierenden Website werden vor allem Verweise auf die Organisationseinheiten gemacht, da diese und nicht die Organisation im Ganzen am Projekt beteiligt waren. In den seltenen Fällen, bei denen die Organisation als Ganzes beteiligt ist, muss eine Organisationseinheit angelegt werden, die

in der Lage ist, den Bezug herzustellen. Die dort vorhandenen Inhalte werden durch die folgenden Eigenschaften des Content-Typs abgebildet:

- *officialName*: string, [1,1]
- *displayName*: string, [1,1]
- *organisation*: Organisation, [1,1]
- *competence*: isTidyHtmlWithCleanup(richtext), [0,1]
- *head*: Person, [1,*]
- *employee*: Person, [1,*]
- *formerEmployee*: Person, [0,*]
- (*addableContent*: OrganisationalUnit)
- *parent*: computed, [1,1]

Ebenso wie bei Organisation werden bei *OrganisationalUnit* die Eigenschaften *officialName* und *displayName* zur Identifizierung und verkürzten Bezeichnung der Organisationseinheit genutzt. *OfficialName* sollte dabei eine möglichst vollständige Beschreibung der Organisationseinheit beinhalten. Dazu gehört auch die Nennung einer Organisationseinheit, zu der die Organisationseinheit selbst zugeordnet sein kann. Die Eigenschaft *competence* dient dazu, dass die Organisationseinheit je nach Schwerpunkt ihre Zuständigkeiten oder Interessengebiete nennt und sich damit dem Betrachter der Website vorstellt. Die Eingabe hierfür erfolgt in HTML, wodurch Querverweise und Hervorhebungen durch die Setzung der entsprechenden Tags ermöglicht werden. Die Eingabe dieses Feldes ist nicht obligatorisch, da eine ausführliche Beschreibung nur für eigene Organisationseinheiten der Universität erforderlich ist. Dem hingegen wäre die Beschreibung von organisationsfremden Einheiten zwar nützlich für den interessierten Leser, dient aber nicht den Zwecken für die Ziel-Website. Hier wird das Feld *competence* für Organisationseinheiten, die nicht zur Organisation Universität Koblenz-Landau gehören, nicht verwendet. Jeder Organisationseinheit wird ein Leiter zugewiesen, der für die Aktivitäten derselben verantwortlich ist. Der Verweis auf die leitenden Personen der Organisationseinheit ermöglicht das Eigenschaftsfeld *head*, das eine Verknüpfung zu einer oder mehrerer Instanzen des Content-Typs *Person* herstellt. Organisationseinheiten verfügen über mindestens einen Mitarbeiter, auch wenn es sich nur um den eben beschriebenen Leiter der Organisationseinheit handelt. Diese Mitarbeiter werden im Eigenschaftsfeld *employee* eingetragen. Ebenso wie das Feld *formerEmployee*, das auf ehemalige Mitarbeiter verweist, können hier mehrere Verknüpfungen zu Instanzen des Content-Typen Person gemacht werden. Organisationseinheiten können selbst in weitere Organisationseinheiten aufgeteilt werden. Ein Beispiel hierfür ist im Universitätsumfeld eine die Organisationseinheit Arbeitsgruppe, die Bestandteil eines Instituts als Organisationseinheit der Organisation Universität ist. Die Eigenschaft *addableContent* ermöglicht dafür das Hinzufügen von Organisationseinheiten (*OrganisationalUnit*) innerhalb einer Organisationseinheit. Um die Nachvollziehbarkeit der Hierarchie von Organisationseinheiten und Organisationen zu gewährleisten, existiert das berechnete Feld *parent*. Es sucht nach der Organisation oder Organisationseinheit, zu der die aktuelle Organisationseinheit gehört und zeigt diese an.

4.2.6 Content-Typ *OrganisationSet*

Für die Speicherung und Erzeugung der Instanzen des Content-Typs *Organisation* wird der Container *OrganisationSet* definiert. Somit ist eine zentrale Verwaltung dieses Content-Typs gewährleistet. Es handelt sich um eine Spezialisierung von *FolderHTML*. *OrganisationSet* verfügt über folgende Eigenschaftsfelder:

- *name*: string, [1,1]
- *descriptionHTML*: isTidyHtmlWithCleanup(richtext), [0,1]

- (*addableContentExclusive*: Organisation)

Das Feld *name* enthält die Bezeichnung für den Container und ist obligatorisch. Mit *descriptionHTML* können weitere Informationen zu *OrganisationSet* selbst und den darin gehaltenen Containern in HTML-Code hinzugefügt werden. *addableContentExclusive* legt fest, dass lediglich Instanzen des Typs *Organisation* in diesen Container eingefügt werden dürfen und nur *OrganisationSet* solche Instanzen erzeugen darf.

4.2.7 Content-Typ *Project*

Der Content-Typ *Project* dient zur Sammlung der allgemeinen Informationen zu einem Projekt und deren Darstellung. Er stellt hierbei Informationen zusammen, die auf einführenden Beschreibungsseiten zum jeweiligen Projekt zu finden sind. Im Beispiel deckt sich der Inhalt des Content-Typen in großen Teilen mit den Inhalten der Webseite „Background“. *Project* besitzt folgende Eigenschaften:

- *name*: string, [1,1]
- *logo*: Image, [0,1]
- *overview*: isTidyHtmlWithCleanup(richtext), [1,1]
- *participant*: Person, [1,*]
- *conference*: Conference, [0,*]
- *partner*: OrganisationalUnit, [0,*]
- *contactInformation*: isTidyHtmlWithCleanup(richtext), [0,1]
- *acknowledgements*: isTidyHtmlWithCleanup(richtext), [0,1]

In der Eigenschaft *name* wird der Name des Projekts angegeben. Ebenfalls eng verbunden mit der Identifizierung eines Projekts ist die Zuweisung eines Logos *logo*, also einer Art Markenzeichen für das Projekt. Hierbei kann auf den später beschriebenen Content-Typ *Image* zurückgegriffen werden, der für die Verwaltung von Bildern verwendet wird. Da ein Projekt zwar einen Namen besitzen muss, aber nicht zwangsläufig auch ein Logo, ist diese Eigenschaft nur optional. Die Inhalte und Ziele des Projekts können durch die Eigenschaft *overview* dargestellt werden. Dieses Feld erlaubt Eingaben in HTML-Format, damit der Text zur besseren Darstellung formatiert werden kann und Hyperlinks zu anderen Websites oder zur eigenen Website eingefügt werden können. Bei jedem Projekt gibt es beteiligte Mitarbeiter, die durch Einträge im Feld *participant* zugewiesen werden können. Dieses Feld stellt eine Verknüpfung zum Content-Typ *Person* her und ist obligatorisch. In der wissenschaftlichen Diskussion werden Projektthemen in der Regel auch in Tagungen und wissenschaftlichen Konferenzen vorgetragen. Die Möglichkeit der Zuweisung von solchen Konferenzen ist durch das Eigenschaftsfeld *conference* gegeben. Auch hier handelt es sich um einen Verweis auf einen anderen Content-Typen, nämlich *Conference*. Die Eigenschaft *conference* ist optional beliebig, da es für ein Projekt nicht zwingend erforderlich ist, dass Ergebnisse von Projekten durch Beiträge an Konferenzen verbreitet werden. Neben den an einem Projekt beteiligten Personen gibt es Organisationen und Organisationseinheiten, die die Arbeit unterstützen oder selbst aktiv mitarbeiten. Solche Beteiligten können im Eigenschaftsfeld *partner* eingetragen werden. Hierzu werden Verknüpfungen ausschließlich zum Content-Typ *OrganisationalUnit* angelegt. Auch diese Eigenschaft ist optional, da es sich bei Projekten um völlig autonome Entwicklungen handeln kann. Um als Betrachter der Webseiten eines Projekts mit dessen Beteiligten Kontakt aufnehmen zu können, ist die Angabe von Ansprechpartnern unerlässlich. Zur Integrierung von Beschreibungen der Ansprechpartnern oder anderer Informationsquellen wie Mailinglisten oder Foren, wurden die Kontaktinformationen als eigenständiges Eigenschaftsfeld *contactInformation* modelliert. Die dort gemachten Eingaben sind im HTML-Format und sind aufgrund ihres ergänzenden Charakters optional. Gerade bei fortgeschrittenen

Projekten ist es üblich, dass Projektbeteiligten, die sich um die Entwicklung des Projekts verdient gemacht haben, eine Danksagung auszusprechen. Einträge dieser Art sind im *Project*-Content-Typ im Eigenschaftsfeld *acknowledgements* möglich. Hierbei handelt es sich um ein Feld für Eintragungen im HTML-Format, wodurch Hervorhebungen durch Formatierung und das Setzen von Hyperlinks unterstützt wird. Dieses Eigenschaftsfeld wird nicht als obligatorisch angenommen.

4.2.8 Content-Typ *Person*

Innerhalb von Projekten und Organisationseinheiten werden stets Personen als Beteiligte zugewiesen. Sie besitzen dort verschiedene Rollen und je nach Art der Anzeige ist es notwendig, unterschiedliche Informationen zu diesen Personen darzustellen. Zur Verwaltung von Personendaten wurde im Zielsystem der Content-Typ *Person* eingeführt. Er wird von Content-Typen referenziert und stellt personenbezogene Daten zur Verfügung. Die Eigenschaften von Person sind die folgenden:

- *surname*: string, [1,1]
- *firstName*: string, [1,1]
- *academicDegree*: string, [0,1]
- *emailAddress*: isEmail(string), [0,1]
- *websiteURL*: isURL(string), [0,1]
- *telephone*: isInternationalPhoneNumber(string), [0,1]
- *fax*: isInternationalPhoneNumber(string), [0,1]
- *fieldsOfInterest*: isTidyHtmlWithCleanup(richtext), [0,1]

Die Eigenschaftsfelder *surname* und *firstName* sind die wichtigsten Felder zur eindeutigen Bestimmung der beschriebenen Person. Auf Webseiten wird in der Regel über die Angabe dieser Eigenschaften auf Personen verwiesen. Deshalb werden sie auch als obligatorisch modelliert. Ebenfalls zu den Elementen der Anrede einer Person gehört das Eigenschaftsfeld *academicDegree*. Dort kann der akademische Titel eingetragen werden, den die beschriebene Person besitzt. Diese Erfassung ist notwendig, da in bestimmten Veröffentlichungen nicht nur der natürliche Name (sprich Vor- und Nachname) verlangt wird, sondern auch die Nennung des akademischen Titels. Schließlich werden in den folgenden Eigenschaftsfeldern Daten zur Kontaktaufnahme erfasst. Besonders wichtig ist hierbei die Email-Adresse, die im Feld *emailAddress* vertreten ist. Auf der zu migrierenden Website wird bei fast jeder Nennung des Namens einer Person ein Hyperlink zu seiner Email-Adresse zur Verfügung gestellt. Dennoch wird diese Eigenschaft als optional modelliert, da es sein kann, dass die Email-Adresse einer Person unbekannt ist oder diese Person die Veröffentlichung der Adresse ablehnt. Zur Gewährleistung korrekter Einträge für die Email-Adresse wird diese durch einen geeigneten Validator auf Einhaltung der Syntax geprüft. Nützlich zur Beschreibung einer Person ist zudem die Angabe ihrer persönlichen Website. Diese Eigenschaft wird im Feld *websiteURL* berücksichtigt und stellt durch eine URL-Validierung die Einhaltung der URL-Syntax sicher. Die nun folgenden Eigenschaftsfelder beziehen sich verstärkt auf Personen, die der Universität Koblenz-Landau angehören, da sie bereits einen sehr großen Detaillierungsgrad aufweisen. Dazu gehören die Angaben zu Telefonnummer und Faxnummer, repräsentiert in den Eigenschaftsfeldern *telephone* und *fax*. Diese Felder werden auf Validität als internationale Telefonnummer geprüft, wie es von Plone als Validator angeboten wird. Zuletzt können zu jeder Person Interessengebiete und weitere Informationen im Eigenschaftsfeld *fieldsOfInterest* angegeben werden. Durch die Eingabe im HTML-Format ist es möglich, Hervorhebungen und Hyperlinks einzufügen.

4.2.9 Content-Typ *PersonSet*

Ebenso wie bei *OrganisationSet* ist eine zentrale Verwaltung der im Zielsystem erfassten *Personen* interessant. Hierfür wurde der Container *PersonSet* entwickelt, der sich von dem Content-Typ *FolderHTML* ableitet. Er besitzt die folgenden Eigenschaften:

- *name*: string, [1,1]
- *descriptionHTML*: isTidyHtmlWithCleanup(richtext), [0,1]
- (*addableContentExclusive*: Person)

Mit *name* kann dem Container ein Name gegeben werden, der seine Funktion als Sammel-Content-Typ für Personen beschreibt. Näher gehende Informationen können in dem Feld *descriptionHTML* in HTML-Format eingefügt werden. Durch *addableContentExclusive* ist festgelegt, dass innerhalb des Containers, und nur dort, nur Instanzen des Content-Typs *Person* eingefügt werden können.

4.2.10 Content-Typ *Conference*

Zur Verwaltung von Inhalten bezüglich Tagungen und Konferenzen wird der Content-Typ Conference eingesetzt. Er umfasst folgende Eigenschaften:

- *name*: string, [1,1]
- *descriptionHTML*: isTidyHtmlWithCleanup(richtext), [0,1]
- *beginDate*: date, [0,1]
- *endDate*: date, [0,1]
- *location*: string, [0,1]
- *websiteURL*: isURL(string), [0,1]

Die Eigenschaft *name* enthält den Namen der Konferenz. Mittels einer Eintragung, die im HTML-Format erfolgen kann, können im Feld *descriptionHTML* eine erweiterte Beschreibung und Verweise zu anderen Webseiten gemacht werden. Der Zeitraum, an dem eine Tagung stattfindet, kann in den Felder *beginDate* und *endDate* vermerkt werden. Das Eigenschaftsfeld *location* enthält hingegen Angaben zum Ort der Veranstaltung. Normalerweise haben Tagungen eine eigene Website. Ein Link zu dieser kann über die Eigenschaft *websiteURL* gesetzt werden. Hierbei überprüft ein Validator, ob es sich bei Eingabe tatsächlich um eine Zeichenkette handelt, die der Syntax einer Internet-Adresse entspricht.

4.2.11 Content-Typ *ConferenceSet*

Im Zielsystem ist eine Vielzahl von Tagungen und Präsentationen zu verwalten. Der Container *ConferenceSet* ermöglicht die zentrale Speicherung von Instanzen des Content-Typs *Conference*. Er ist eine Spezialisierung von *FolderHTML* und verfügt über folgende Eigenschaften:

- *name*: string, [1,1]
- *descriptionHTML*: isTidyHtmlWithCleanup(richtext), [0,1]
- (*addableContentExclusive*: Conference)

Das Feld *name* dient zur Bezeichnung des Containers, während *descriptionHTML* die Möglichkeit gibt, eine zusätzliche Beschreibung in HTML-Format einzufügen. Dass in *ConferenceSet* nur Instanzen des Typs *Conference* eingefügt werden dürfen, wird durch *addableContentExclusive* festgelegt, was gleichzeitig ausschließt, dass Instanzen von *Conference* außerhalb von *ConferenceSet* erzeugt werden dürfen.

Die bis hierhin behandelten Content-Typen und deren Beziehungen sind in Abbildung 43 dargestellt.

4.2.12 Content-Typ *FAQ*

Auf der zu migrierenden Website befindet sich eine Webseite, die ausschließlich FAQs enthält. Der Aufbau einer FAQ ist verhältnismäßig einfach. Die Eigenschaften, die aus dem Datentyp FAQ entnommen wurden, sind:

- *question*: text, [1,1]
- *answer*: isTidyHtmlWithCleanup(richtext), [1,1]
- (*keyword*: string, [1,*])

Eine FAQ ist eine Gegenüberstellung einer Frage mit ihrer Antwort. Die Frage ist im Eigenschaftsfeld *question* enthalten und ist eine einfache, obligatorische Zeichenkette. Die Antwort befindet sich im Feld *answer*. Da die Antworten umfangreicher ausfallen und dort häufig Verweise auf andere Webseiten notwendig werden, enthält dieses Eigenschaftsfeld Eingaben in HTML-Format. Ebenso wie die Frage ist die Belegung dieses Felds obligatorisch. Die letzte Eigenschaft des Content-Typs *FAQ*, nämlich *keyword*, dient dazu, der FAQ eine thematische Zuordnung zu geben. In der Auflistung wurde sie deshalb in Klammern gesetzt, da sie keine originäre Eigenschaft des Content-Typs *FAQ* darstellt, sondern ein Feld der von Plone für jedes Objekt bereitgestellten Metadaten ist. Die Belegung eines bestimmten Schlüsselworts für eine FAQ ist wichtig, damit die thematische Unterscheidung, wie sie auf der zu migrierenden Webseite gemacht wurde, möglich ist. Hierfür werden in das Zielsystem Schlüsselwörter zur Unterscheidung der folgenden Themen eingeben:

- *General*: allgemeine Fragen
- *DTD*: Fragen zur DTD
- *Metaschema*: Fragen zum Metaschema
- *Participation*: Fragen zur Möglichkeit der Beteiligung an diesem Projekt
- *Other*: sonstige Fragen

Die Definition weiterer Schlüsselwörter für FAQs ist möglich, aber angesichts der bisherigen Einträge nicht notwendig. Zur Gruppierung der FAQs reicht die Auswahl eines Schlüsselworts aus. Grundsätzlich können einem Objekt allerdings mehrere Schlüsselwörter zugeordnet werden. Da die Zuweisung weiterer Schlüsselwörter unabhängig von der Zugehörigkeit des Objekts zum Content-Typ *FAQ* sinnvoll sein kann, wird in *keyword* mehr als ein Eintrag zugelassen.

4.2.13 Content-Typ *FAQSet*

Neben der einzelnen FAQ soll in der Ziel-Website auch eine thematisch geordnete Übersichtsliste enthalten sein. Hierbei werden die einzelnen FAQs gemäß der Eintragungen im *keyword*-Feld der Metadaten in optisch getrennte Gruppen eingeteilt. Von dort aus ist es möglich, auf die Anzeige der einzelnen FAQ zu navigieren. Hierfür wird ein Container aller FAQs als Spezialisierung von *FolderHTML* erstellt, der folgende Eigenschaften enthält:

- *name*: string, [1,1]
- *descriptionHTML*: isTidyHtmlWithCleanup(richtext), [0,1]
- (*addableContentExclusive*: FAQ)

In der Eigenschaft *name* kann ein aussagekräftiger Name für die Webseite der gesammelten FAQs eingegeben werden. Vorgesehen ist hierfür „Frequently Asked Questions to GXL“. Zur Beschreibung des Inhalts dieser Übersichtsseite wird die Eigenschaft *descriptionHTML* bereitgestellt. Die Modellierung eines Feldes, das Eingaben in HTML-Format vorsieht, ist zweckmäßig, da auf der zu migrierenden Website auf der

entsprechenden Webseite auch Hyperlinks benutzt wurden. Da der Content-Typ *FAQSet* ein Container ist, werden die ihm zugewiesenen FAQs nicht als Eigenschaft definiert. Stattdessen handelt es sich um Content des Content-Typen *FAQ* und wird deshalb auch nicht in den Eigenschaften des Content-Typen aufgeführt. Tatsächlich soll in diesen Container außer *FAQ* keine Instanzen anderer Content-Typen hinzugefügt werden können. Diese Restriktion wird in dem geklammerten Eintrag *addableContentExclusive*, einfügbarer Content, aufgeführt. Es beschreibt, dass im Content-Typ *FAQSet* lediglich das Hinzufügen von FAQs möglich ist. Ebenso darf keine Instanz von *FAQ* außerhalb des *FAQSet*-Containers erzeugt werden.

Die Content-Typen *FAQ* und *FAQSet* sind in Abbildung 44 nachzuvollziehen.

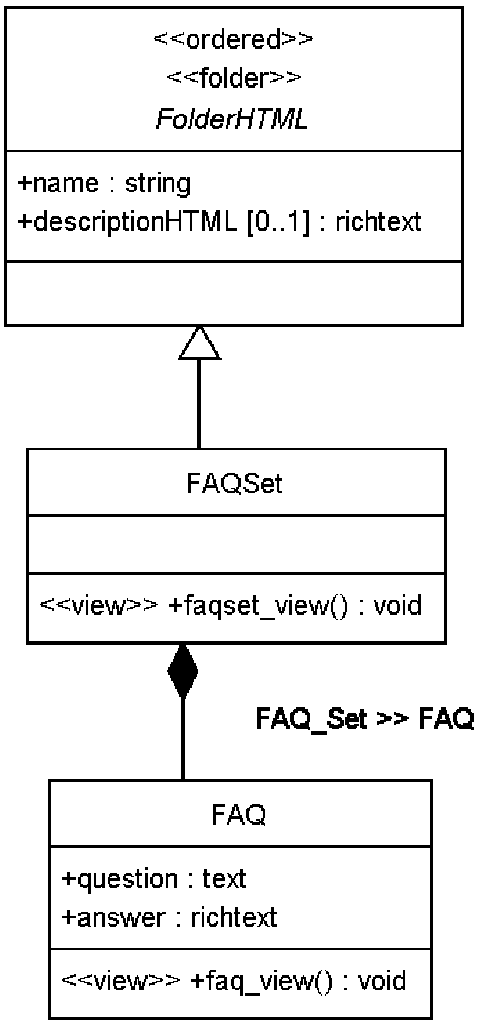

Abbildung 44: Datenmodell zu *FAQ* und *FAQSet*

Einen wesentlichen Teil der zu migrierenden Website nimmt die Beschreibung der Beispiele ein. Die Beispiele sind in drei Abstraktionsebenen unterteilt. Einerseits werden Beispiele auf schematischer Ebene angezeigt (Ebene M1), die schließlich durch Beispiele auf Instanzebene verdeutlicht werden (Ebene M0). Die Metaschema wiederum definiert alle Schemabeispiele und sich selbst und befindet sich damit auf der höchsten Abstraktionsebene (Ebene M2). Beide Beispielarten werden im Content-Typ *Example* erfasst. Da die Instanzen der genannten Content-Typen in einem Zusammenhang stehen, werden sie in einem Container *ExampleSet* zusammengefasst. Diese angesprochenen Content-Typen werden wie folgt beschrieben.

4.2.14 Content-Typ *Example*

Der Content-Typ *Example* beschreibt GXL-Beispiele. Er deckt dabei alle Beispiele ab, die auf den entsprechenden Webseiten vorkommen. Dazu gehören nicht nur die Schemata und Instanzen, sondern auch das Metaschema. *Example* verfügt hierbei über folgende Eigenschaften:

- *name*: string, [1,1]
- *descriptionHTML*: isTidyHtmlWithCleanup(richtext), [0,1]
- *classDiagram*: Image, [0,1]
- *graph*: Image, [1,1]
- *gxl*: isTidyHtmlWithCleanup(richtext), [1,1]
- *instance*: Example, [1,1]
- (*keyword*: string, [1,*])

Der Name des Beispiels, *name*, beschreibt in Kurzform seinen Inhalt und ist obligatorisch. Um das vorgestellte Beispiel einführend zu kommentieren, wird optional die Eigenschaft *descriptionHTML* zur Verfügung gestellt. In der zu migrierenden Website reichte es aus, dass nur einfacher Text ohne HTML-Anweisung eingetragen werden kann. Um allerdings an dieser Stelle keine unnötigen Restriktionen zu setzen, wurde *descriptionHTML* zur Eingabe von HTML-Text konzipiert. Wichtiger für das Beispiel sind die nun folgenden Eigenschaften, die alle obligatorisch sind. In *classDiagram* wird für das Beispiel ein Bild gespeichert, das das Beispiel in einem UML-Klassendiagramm darstellt. Auch wenn hierbei für Instanzbeispiele eine Darstellung als Objektdiagramm denkbar wäre, gibt es im Legacy-System keine solche Repräsentation. Deshalb wurde diese Eigenschaft als optional modelliert. Bei *graph* handelt es sich ebenfalls um eine bildliche Darstellung, in diesem Fall des Beispielgraphen. Die Umsetzung der Diagramme in die Graphenbeschreibungssprache GXL wird in dem Feld *gxl* eingetragen. Dieses besitzt Verknüpfungen zu anderen Dokumenten, deshalb muss der Quelltext in HTML gehalten werden. Bei Schemabeispielen herrscht eine hierarchische Ordnung vor. Jedes Beispiel ist eine Instanz eines anderen Schemas. Selbst das Metaschema ist eine Instanz von sich selbst. Diese Verbindung zwischen den Beispielen wird durch das Feld *instance* hergestellt. Es ist obligatorisch und verweist genau auf ein anderes Schema. Um die Beispiele Gruppen zuzuordnen, wird das Metadatenfeld *keyword* für Schlüsselwörter verwendet. Die für die Instanzen der Content-Typen *Example* notwendigen Kategorien sind Folgende aufgelistet:

- Beispiele einfacher Graphen
- Beispiele zu Hypergraphen
- Beispiele zu hierarchischen Graphen
- Beispiel zum Metaschema
- Ebene M0
- Ebene M1
- Ebene M2

Wie bereits bei FAQ wird das Feld *keyword* als obligatorisch modelliert. Es können mehrere Schlüsselwörter zugewiesen werden, was vor allem bei den Schema- und den Instanzbeispielen notwendig ist. Dort wird neben dem Schlüsselwort für die Beispielgruppe auch die Ebene mit einem entsprechenden Schlüsselwort angegeben. Die Ebene M0 steht für Instanzbeispiele, M1 für Schemabeispiele und M2 für das Metaschema.

4.2.15 Content-Typ *ExampleSet*

Um alle Instanzen von *Example* in einem Content-Typen zu sammeln, wird der Container *ExampleSet* mitmodelliert. Die hier zusammengefassten Instanzen sind von dort zentral zu verwalten. Hierfür verfügt der Content-Typ, der sich von *FolderHTML* ableitet, über folgende Eigenschaften:

- *name*: string, [1,1]
- *descriptionHTML*: isTidyHtmlWithCleanup(richtext), [0,1]
- *download*: file, [0,*]
- (*addableContentExclusive*: Example)

Über das Eigenschaftsfeld *name* wird *ExampleSet* kurz beschrieben. Wie auch bei anderen Container-Content-Typen wird hier eine Eigenschaft *descriptionHTML* definiert, die Eingaben in HTML-Format vorsieht. Das Feld *download* sieht die Bereitstellung von Dateien vor, die in Zusammenhang mit den Beispielen verfügbar gemacht werden sollen. Schließlich legt die Eigenschaft *addableContentExclusive: Example* fest, dass in diesen Container lediglich Instanzen des Content-Typs *Example* eingefügt werden können. Gleichzeitig wird, wie „exclusive“ andeutet, modelliert, dass nur innerhalb von *ExampleSet* Instanzen von *Example* erzeugt werden dürfen.

Abbildung 45 stellt die Zusammenhänge und Belegung der Felder von *Example* und *ExampleSet* grafisch dar.

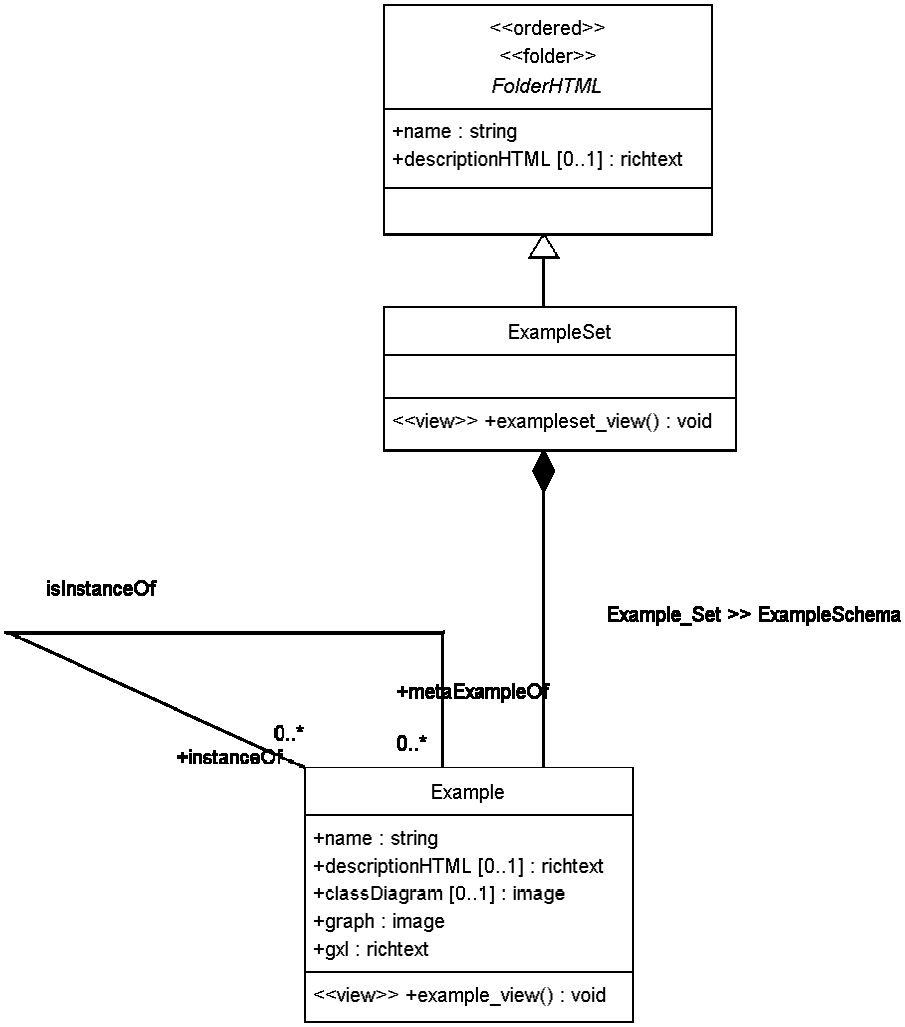

Abbildung 45: Datenmodell zu *Example* und *ExampleSet*

4.2.16 Content-Typ *Tool*

Der Content-Typ *Tool* beschreibt Software, die in engem Zusammenhang mit der Arbeit des GXL-Projekts steht. Dabei ist es weniger wichtig, eine ausführliche Beschreibung der jeweiligen Software zu leisten, als vielmehr Kontaktdaten und einführende Informationen zum Verwendungszweck des jeweiligen Werkzeugs zur Verfügung zu stellen. Tool umfasst für diese Aufgabe folgende Eigenschaften:

- *name*: string, [1,1]
- *descriptionHTML*: isTidyHtmlWithCleanup(richtext), [0,1]
- *currentVersion*: string, [0,1]
- dateCurrentVersion: string, [0,1]
- *platform*: string, [0,*]
- *websiteURL*: isURL(string), [0,1]
- *downloadURL*: isURL(string), [0,1]
- *sourceCodeURL*: isURL(string), [0,1]
- *supplyingOrganisation*: Organisation, [0,*]
- *supplyingOU*: OrganisationalUnit, [0,*]
- *supplyingPerson*: Person, [0,*]
- (*keyword*: string, [1,*])

Das Eigenschaftsfeld *name* beschreibt den Namen des Werkzeugs. Für die Eingabe einer umfangreicheren Beschreibung des Werkzeugs im HTML-Format, enthält Tool die Eigenschaft *descriptionHTML*. Um auf den aktuellen Stand der Version und deren Aktualität hinzuweisen, wurden die Felder *currentVersion* und *dateCurrentVersion* integriert. *CurrentVersion* beschreibt die aktuelle Versionsnummer, während *dateCurrentVersion* das Veröffentlichungsdatum der aktuellen Version enthält. Die Angabe dieser Informationen ist allerdings nicht elementar wichtig, weswegen diese Felder als optional modelliert wurden. Um anzuzeigen, für welche Plattformen das Werkzeug geeignet ist, kann darauf im Feld *platform* eingegangen werden. Internet-Adressen zur Website des jeweiligen Tools, zum Download der Software oder ihres Quellcodes werden in den Feldern *websiteURL*, *downloadURL* und *sourceCodeURL* erfasst. Um die Einhaltung der Syntax für URLs zu gewährleisten, werden diese Felder validiert. Für Werkzeuge, die vom Institut für Softwaretechnik verfügbar gemacht werden, wurde die Eigenschaft *download* entworfen. Hiermit wird eine Verbindung zwischen der allgemeinen Werkzeugbeschreibung und der detaillierten Downloadansicht hergestellt. Um Kontaktinformationen zu den für die Software verantwortlichen Personen, Organisationen oder Organisationseinheiten zu speichern, existieren die Eigenschaftsfelder *supplyingPerson, supplyingOrganisation* und *supplyingOU*. Diese Felder stellen eine Verknüpfung zu Instanzen der Content-Typen *Person*, *Organisation* bzw. *OrganisationalUnit* her. Hier können Einträge für mehrere Organisationen und Personen gemacht werden. Über die Eingabe in das Metadatenfeld *keyword* wird dem beschriebenen *Tool* zugewiesen, zu welcher Funktionsgruppe es gehört. Um die auf der zu migrierenden Website dargestellten Werkzeuge zu beschreiben, werden durch Schlüsselwörter die folgenden Kategorien abgedeckt:

- Konvertierungswerkzeuge
- Visualisierungswerkzeuge
- Analyse- und Transformationswerkzeuge
- Werkzeuge für das Software-Engineering
- Grapheditoren
- Werkzeuge für verschiedene Zwecke
- Werkzeuge, die an der Universität Koblenz-Landau entwickelt wurden

4.2.17 Content-Typ *ToolSet*

Zur Verwaltung der Instanzen des Content-Typs *Tools* wird auch hier eine Spezialisierung von *FolderHTML* als Container verwendet. Er besitzt folgende Eigenschaften:

- *name*: string, [1,1]
- *descriptionHTML*: isTidyHtmlWithCleanup(richtext), [0,1]
- (*addableContentExclusive*: Tool)

Bei *ToolSet* beschreibt *name* kurz den Container; in diesem Fall wird in Anlehnung an die zu migrierende Website der Titel „Tool Catalogue“ verwendet. Das Feld *descriptionHTML* ermöglicht die Beschreibung des Containers unter Verwendung von HTML. Abschließend wird durch *addableContentExclusive* für *ToolSet* festgelegt, dass lediglich Instanzen des Content-Typs *Tool* in den Container eingefügt werden dürfen. Es ist nicht zulässig, Instanzen von *Tool* außerhalb einer Instanz von *ToolSet* zu erzeugen.

Abbildung 46 stellt die Content-Typen *Tool* und *ToolSet* in ihrem Kontext mit von ihnen referenzierten Content-Typen dar.

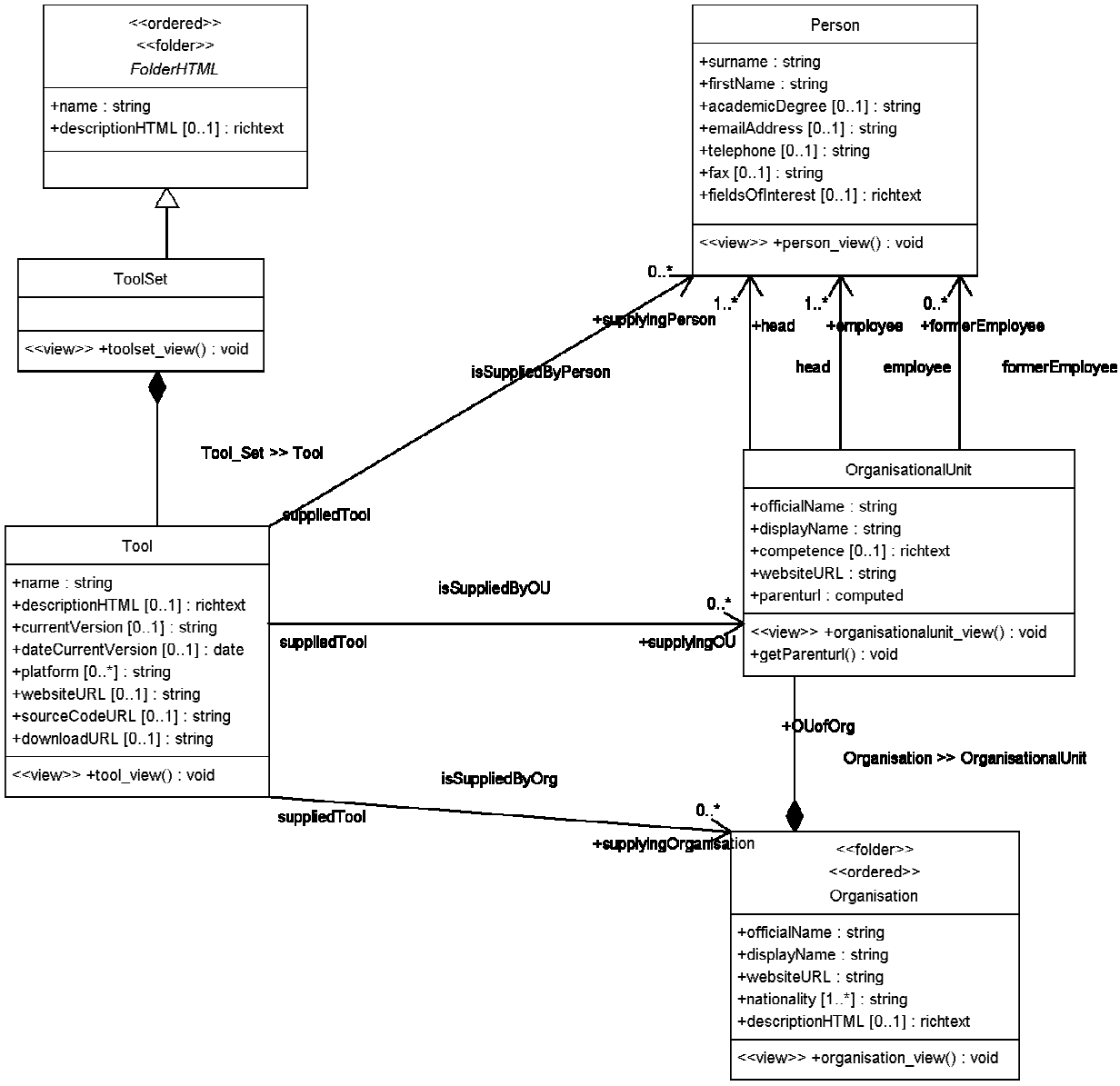

Abbildung 46: Datenmodell zu *Tool* und *ToolSet*

4.2.18 Content-Typ *Listing*

Der Content-Typ *Listing* ist zur Verwaltung von Quelltexten in verschiedenen Programmier- und Dokumentenbeschreibungssprachen gedacht. In der Legacy-Analyse wurde aufgrund der häufigen Verwendung von Quelltexten dieser Datentyp mitmodelliert. In der zu migrierenden Website sind zwei DTDs und ein XML-Schema integriert. Zudem werden unterschiedliche Quelltexte bezüglich der Kommentierung und der Syntaxhervorhebung gespeichert. Der Content-Typ *Listing* weist die nachfolgend aufgelisteten Eigenschaften auf:

- *name*: string, [1,1]
- *descriptionHTML*: text, [0,1]
- *version*: string, [0,1]
- *language*: string, [0,1]
- *copyright*: isTidyHtmlWithCleanup(richtext), [0,1]
- sourceCodeCommented: text, [0,1]
- *sourceCodeSyntaxHighlighted*: isTidyHtmlWithCleanup(richtext), [0,1]
- sourceCodeUncommented: file, [0,1]

Die Eigenschaft *name* gibt in verständlicher Kurzform wieder, welchen Inhalt das Listing besitzt und ist obligatorisch. Eine sinnvolle Verwendung dieses Felds für die Ziel-Website wäre bspw. die Benennung des Listings in „GXL-DTD, Version 1.0“. Damit kann der Leser des *name*-Eintrags direkt erkennen, welches Thema das Listing behandelt (also die GXL DTD) sowie auf welche Version sich der Eintrag bezieht. Im Feld *descriptionHTML* kann der Inhalt ausführlicher in HTML-Format dargestellt werden. Da bereits durch *title* und *descriptionHTML* die wesentlichen Informationen zum Listing enthalten sein können, ist das Eigenschaftsfeld zur Angabe der dort behandelten Version *version* nur optional. Ebenso verhält es sich mit dem Feld *language*. Die Angabe der verwendeten Programmier- bzw. Dokumentenbeschreibungssprache des Listings in *language* hilft dem Leser bei der Orientierung. Zudem können Unterschiede von Listings, die sich nicht durch den Inhalt, sondern durch die verwendete Sprache unterscheiden, verdeutlicht werden. Die Eigenschaft *copyright* ermöglicht die Angabe eventuell existierender Copyright-Regelungen. An dieser Stelle kann es sinnvoll sein, Kontaktinformationen anzugeben, damit Interessierte sich die Genehmigung zur Nutzung des Listings einholen können. Deshalb wird in diesem Feld eine HTML-Eingabe vorgesehen. Der eigentliche Quelltext ist schließlich in drei optionalen Versionen gespeichert. Der kommentierte Quelltext *sourceCodeCommented* ermöglicht die Eingabe von HTML-Code zur besseren Hervorhebung der Kommentare. Der Quelltext mit Syntaxhervorhebung *sourceCodeSyntaxHighlighted*, greift ebenso auf HTML-Quelltext zur farblichen Code-Hervorhebung zurück. Zuletzt kann in dem Feld *sourceCodeUncommented* eine Datei gespeichert werden, die über unkommentierten Quelltext verfügt. Die Speicherung als Datei ist deshalb notwendig, damit z.B. bei DTDs explizit darauf verwiesen werden kann.

Abbildung 47 zeigt das Datenmodell von *Listing*.

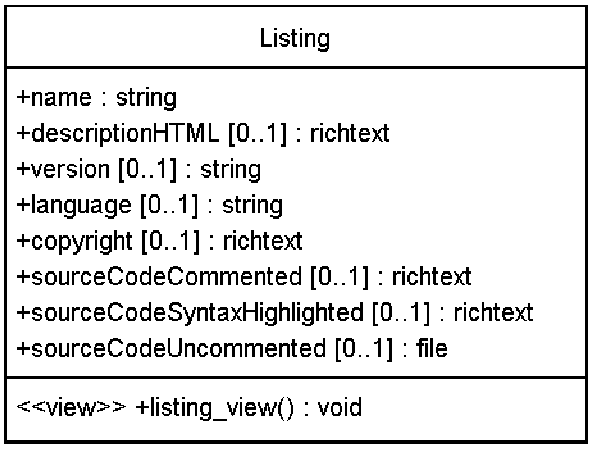

Abbildung 47: Datenmodell zu Listing

4.2.19 Content-Typ *WebArticle*

Eine Instanz des Content-Typs *WebArticle* dient zur Umsetzung eines zusammenhängenden Dokuments, das in HTML-Code geschrieben ist und über lokale und globale Navigationslinks verfügt. Ein Web-Artikel besteht aus einzelnen Abschnitten, dessen Datenstruktur der einer herkömmlichen Webseite sehr ähnlich ist. Somit ist ein Instanz des Content-Typs *WebArticle* ein Container, der einzelne Abschnitte enthält. Diese Abschnitte entsprechen dem Content-Typ *Document*. Gleichzeitig muss dieser Content-Typ über die Funktionalität verfügen, dass automatisch die Navigation zwischen den einzelnen Abschnitten erstellt wird. Der *WebArticle* besitzt folgende Eigenschaften:

- *title*: string [1, 1]
- *description*: text [0, 1]
- (*addableContent*: Document)

Die Eigenschaft *title* ordnet dem Dokument einen passenden Titel zu, während das Feld *description* eine kurze prägnante Beschreibung des Inhalt gibt. Da es sich bei einem Web-Artikel um einen Container von mehreren Abschnitten handelt, legt die Eigenschaft *addableContent: Document* fest, dass dort Instanzen des Content-Typs *Document* hinzugefügt werden können (zu *Document* siehe weiter unten).

Alleine durch die Beschreibung der Datenstruktur von *WebArticle* ist die Spezifikation allerdings noch nicht vollständig. Tatsächlich muss der Content-Typ noch Funktionalität implementieren, die die dynamische Erzeugung lokaler und globaler Navigation ermöglicht. Unter lokaler Navigation wird verstanden, dass es möglich sein soll, von einem Abschnitt des Web-Artikels zum nächsten, bzw. vorigen Abschnitt zu wechseln. Diese Möglichkeit wird durch entsprechende Hyperlinks hergestellt. Die globale Navigation unterstützt hingegen das Wechseln zwischen Abschnitten des Dokuments, die nicht direkt sequenziell aufeinander folgen. Diese Funktionalität kann durch ein Inhaltsverzeichnis hergestellt werden, auf das in jedem Abschnitt des Web-Artikels zugegriffen werden kann. Aus diesen Anforderungen lassen sich folgende Funktionen ableiten:

- *goToNext*: Document
- *goToPrevious*: Document
- *tableOfContents*: Document

Die Funktion *goToNext* ermöglicht es dem Nutzer, vom aktuellen Abschnitt zum Nächsten innerhalb des Web-Artikels zu wechseln. Ist der aktuelle Abschnitt der letzte, so ist diese Funktion deaktiviert. Analog dazu kann durch *goToPrevious* relativ zum aktuellen Dokument zum vorigen Abschnitt gewechselt werden. Auch hier ist die Funktionalität

deaktiviert, wenn der Abschnitt der erste des Web-Artikels ist. Durch Aufruf der Funktion *tableOfContents* kann auf ein beliebiges Element innerhalb des Web-Artikels gewechselt werden. Die genannten Funktionen sind in allen Abschnitten des Content-Typs *WebArticle* abrufbar.

Die Funktionalität des Content-Typs *WebArticle* wird durch das über die Plone-Website verfügbare Produkt *PloneArticle* von IngeniWeb implementiert.[25] Dieses Produkt ist komplexer als das hier beschriebene *WebArticle*. Für die Verwendung im Zielsystem ist lediglich die Erfüllung der hier genannten Eigenschaften von Bedeutung. PloneArticle besteht aus zwei Content-Typen, *PloneArticleMultiPage* und *PloneArticle*. *PloneArticleMultiPage* fungiert wie *WebArticle* als Container. Der Content-Typ *PloneArticle* selbst ist dem Content-Typ *Document* ähnlich. Auf die zusätzliche Funktionalität des Produkts *PloneArticle* wird wegen fehlender Relevanz für die Migration nicht weiter eingegangen.

Eine Sonderrolle im Ziel-Design nimmt die Modellierung der Publikationsliste ein. Sie wurde im Legacy-Design aus der Webseite *Publications* abgeleitet. In der Anforderungsanalyse wurde das Bedürfnis von Seiten des Website-Betreibers festgestellt, dass die Publikationsliste in das bestehende Literaturmanagementsystem des Fachbereichs 4 der Universität Koblenz-Landau *LitDB* eingebunden wird. Durch die Nutzung einer bereits vorhandenen Technologie entfällt die eigene Entwicklungsarbeit für ein solches Verwaltungssystem. Der einzige Aufwand, der sich daraus ergibt, ist die Überführung der Daten von der zu migrierenden Website und die Herstellung einer Schnittstelle zur Anzeige der Daten der Publikationsliste auf der Ziel-Website.

Ebenso interessiert an der Entwicklung einer Schnittstelle zur Überführung der Daten vom Literaturmanagementsystem nach Plone, ist der Website-Verantwortliche der Universitäts-Website. Im Rahmen der Umstellung der Universitäts-Website nach Plone wird er zukünftig eine solche Schnittstelle zur Erstellung von Publikationslisten bereitstellen müssen. Deshalb wurde die Absprache getroffen, dass die Entwicklung dieser Schnittstelle an den Website-Verwantwortlichen outgesourced wird, während die Überführung der Daten in das Literaturmanagementsystem eine interne Aufgabe bleibt.

Innerhalb des Ziel-Designs erscheint die Publikationsliste in Form von drei Content-Typen. Zum einen die Content-Typen für einzelne Publikationseinträge, *Inproceedings* und *Techreport*, und zum anderen die komplette Liste *Publications*.

Für die einzelnen Publikationseinträge ist das in der Legacy-Analyse beschriebene Datenmodell für die Webseite *Publications* beim Ziel-Design nicht von Bedeutung. Einzig relevant sind die Vorgaben des Literaturmanagementsystems des Fachbereichs 4 der Universität Koblenz-Landau. Dessen Datenmodell ist wiederum sehr komplex. Zur Gewährleistung der Übersichtlichkeit wird es nur insoweit beschrieben, als dass es für die Migration relevant ist.

Die Literaturdatenbank des Fachbereichs 4 der Universität Koblenz-Landau *LitDB* baut auf das Datenmodell auf, das durch BibTex vorgegeben ist. Welche Felder gespeichert werden, wird nach Literaturtyp unterschieden. In der Literaturdatenbank sind folgende Literaturtypen verfügbar:[26]

- *article*: ein Artikel aus einer Zeitschrift
- *book*: ein Buch mit Verlagsinformationen
- *booklet*: ein Buch ohne Verlagsinformationen

[25] Siehe http://ingeniweb.sourceforge.net/Products/PloneArticle
[26] Die Beschreibung der einzelnen Literaturtypen wurde von http://www.ecst.csuchico.edu/~jacobsd/bib/formats/bibtex.html entnommen.

- *inbook*: ein Buchausschnitt
- *incollection*: ein Buchausschnitt mit eigenem Titel
- *conference*, *inproceedings*: ein Artikel aus einem Konferenzvortrag
- *manual*: technische Dokumentation
- *mastersthesis*: eine Master-Arbeit
- *misc*: nicht zuzuordnende Literatur
- *phdthesis*: eine Doktorarbeit
- *proceedings*: Gesamtheit der Vorträge einer Konferenz
- *techreport*: ein nicht von einer Zeitschrift veröffentlichter Artikel
- *unpublished*: ein noch nicht veröffentlichtes Dokument

Im Fall der zu migrierenden Webseite sind die Literaturtypen *conference* / *inproceedings* und *techreport* relevant. Nachfolgend werden deshalb nur diese Typen beschrieben und von diesen auch nur die Felder, die durch die Publikationseinträge befüllt werden. Es wird darauf hingewiesen, dass diese Datenbeschreibungen nicht selbst für die Ziel-Website implementiert wurden, sondern sich lediglich auf die bereits vorhandenen Strukturen von *LitDB* beziehen. So existieren im Zielsystem Plone auch keine Entsprechungen von *InProceedings* oder *TechReport*.

4.2.20 InProceedings

Der Literaturtyp *InProceedings* (es wird nur auf *inproceedings* und nicht auf *conference* eingegangen) kommt in der Publikationsliste der zu migrierenden Website am häufigsten vor. Er umfasst Daten, die zu der Beschreibung eines veröffentlichten Vortrags einer Konferenz notwendig sind. Er umfasst folgende Felder:

- *title*: string, [1,1]
- *booktitle*: string, [1,1]
- *year*: int, [1,1]
- *url*: string, [0,*]
- *category*: string, [0,*]
- *author*: Author, [1,*]
- *organisationLitDB*: OrganisationLitDB, [0,*]

Das Feld *title* enthält den Titel des Publikationseintrags. Da Konferenzvorträge in Büchern veröffentlicht werden, steht *booktitle* für den Eintrag des Buchtitels zur Verfügung. Das Pflichtfeld *year* gibt das Veröffentlichungsjahr des Konferenzvortrags an. Über das Feld *url* können beliebig viele Internet-Adressen für den Download des Konferenzvortrags eingegeben werden. Um verschiedene Publikationseinträge logisch miteinander in Verbindung zu bringen, wird das Feld *category* verwendet. Dort können beliebig viele thematische Kategorien eingetragen werden. Im vorliegenden Fall wird das Feld *category* zur Markierung der Publikationseinträge der zu migrierenden Website durch die Kategorie *„GXL_Pub“* verwendet. Die Autoren eines Konferenzbeitrags werden durch das Feld *author* eingetragen, das eine Verbindung zum komplexeren Datentyp *Author* herstellt. Ebenso wird beim Feld *organisationLitDB* verfahren, das Daten einer Organisation sammelt, die die Durchführung einer Konferenz unterstützt.

Der von *InProceedings* verwiesene Datentyp *Author* umfasst folgende Felder:

- *surname*: string, [1,1]
- *firstName*: string, [0,1]
- *url*: string, [0,1]

Das Feld *surname* enthält den Nachnamen des Autors, während *firstName* den Vornamen oder mehrere Vornamen speichert. Falls der Autor über eine eigene Webseite verfügt, kann diese in dem Feld *url* eingegeben werden.

Der Datentyp *OrganisationLitDB* enthält folgende Felder:

- *fullName*: string, [1,1]
- *shortName*: string, [0,1]
- *url*: string, [0,1]

Der volle Name der Organisation muss in das Feld *fullName* eingetragen werden. Ein Kürzel zur Beschreibung der Organisation kann im Feld *shortName* gespeichert werden. Existiert eine Webseite zur Organisation, kann diese optional unter *url* eingetragen werden.

Abbildung 48 zeigt den Aufbau von *InProceedings* und dessen Beziehungen mit den komplexeren Datentypen *Author* und *OrganisationLitDB*.

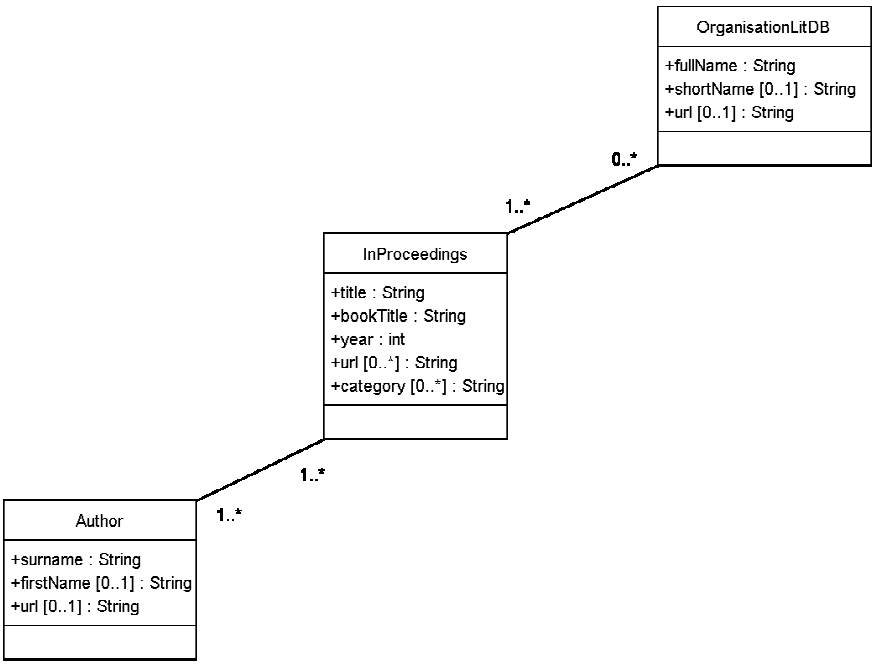

Abbildung 48: Datenmodell zu *InProceedings*

4.2.21 TechReport

Wesentlich seltener kommen Instanzen des Literaturtyps *TechReport* auf der Publikations-Website vor. Dennoch müssen sie zur korrekten Abbildung der Publikationseinträge beschrieben werden. Durch *TechReport* werden Artikel beschrieben, die von einer Institution wie einer Universität und nicht von einer Zeitschrift veröffentlicht wurden. Hierfür werden folgende Felder zur Speicherung der Daten verwendet:

- *title*: string, [1,1]
- *year*: int, [1,1]
- *url*: string, [0,*]
- *category*: string, [0,*]
- *author*: Author, [1,*]
- *institution*: Institution, [1,*]

Das Feld *title* enthält den Titel des Artikels. Das Jahr der Veröffentlichung wird in dem Feld *year* festgehalten. Wenn über das Internet verfügbare Versionen des Artikels existieren, können diese im Feld *url* referenziert werden. Wie schon bei *InProceedings*,

gibt das Feld *category* die Zugehörigkeit von *TechReport* zu einer Gruppe wieder. Auch hier wird durch die Verwendung des Kategorienamens *„GXL_Pub“* die Zugehörigkeit zu der Publikationsliste der zu migrierenden Website ausgedrückt. Das Feld *author* ermöglicht die Zuweisung eines oder mehrere Autoren des Datentyps *Author*. Das Datenmodell ist zu der bei *InProceedings* beschriebenen Struktur von *Author* identisch. Im Feld *institution* werden die veröffentlichenden Institutionen vermerkt.

Institution verfügt über folgende Felder:

- *fullName*: string, [1,1]
- *shortName*: string, [0,1]
- *url*: string, [0,1]

Das Feld *fullName* ermöglicht die Eintragung des vollständigen Namens der veröffentlichenden Institution, während das Feld *shortName* für Kürzel vorgesehen ist. Wenn die Institution über eine Internet-Adresse verfügt, kann diese im Feld *url* eingetragen werden.

Bis jetzt wurden die Datentypen beschrieben, wie sie für die Erfassung der einzelnen Publikationseinträge wichtig sind. Nachfolgend wird erläutert, wie die in der Datenbank erfassten Einträge über die Schnittstelle *portal_litdbtool* als Publikationsliste ausgegeben und formatiert werden.

Portal_litdbtool

Das Werkzeug *portal_litdbtool* ist ein vom Website-Verantwortlichen erstelltes Produkt für Plone zur Abfrage von *LitDB*. Die Funktionsweise von *portal_litdbtool* ist nur insoweit von Belang, als dass es zur Ausgabe er Literatureinträge der Ziel-Website notwendig ist.

Über die Funktion *portal_litdbtool.queryByCategory(category)* werden durch Übergabe des Arguments *category* alle Literaturschlüssel, also eindeutige Bezeichner für jeden Literatureintrag, der zugehörigen Kategorie zurückgegeben. Durch *portal_litdbtool.getEntries(litkeys)* werden zu den Literaturschlüsseln die vollständigen Einträge aus der Datenbank gelesen und in Form einer Liste von Dictionaries zurückgegeben. Auf diesen Listeneinträgen können nun mehrere Funktionen zum Formatieren der Publikationseinträge ausgeführt werden. Ein Listeneintrag wird als *row* bezeichnet.

- *row[,litkey']*: es wird der Wert des Schlüssels *,litkey'* innerhalb von *row* ausgelesen. Bei dem Schlüssel *,litkey'* handelt es sich um den Literaturdatenbank-internen eindeutigen Bezeichner für einen Literatureintrag. Diese ist keine schnittstellenspezifische Funktion, sondern eine Python-Methode.
- *portal_litdbtool.formatAuthors(row)*: die Funktion *formatAuthors* von *portal_litdbtool* liest die Autoren eines Publikationseintrags aus und formatiert deren Namen. Da HTML-Code übergeben wird, erzeugt *formatAuthors* bei Vorhandensein einer URL für den Autor (siehe Beschreibung zu *InProceedings*) gleichzeitig einen Hyperlink.
- *portal_litdbtool.formatTitle(row)*: durch die Funktion *formatTitle* von *portal_litdbtool* wird aus dem Listeneintrag *row* der formatierte Titel ausgelesen.
- *portal_litdbtool.formatOthers(row)*: durch *formatOthers* gibt *portal_litdbtool* alle anderen Daten innerhalb des Publikationseintrags *row* aus. Dazu gehören je nach Literaturtyp die Daten zu *booktitle*, *year*, *organisationLitDB*, *institution* oder *url*.

Die Ausgabe der Publikationsdaten der oben genannten Funktionen ist vorformatiert. So wird bei *formatAuthors* von den Vornamen der Autoren nur der Anfangsbuchstabe angezeigt und bei *formatOther* das Erscheinungsjahr geklammert. Dennoch kann durch Integration der Funktionen in einem Template ein beschränkter Einfluss auf die Endformatierung, z.B. durch Einfügen von HTML-Tags oder Text, genommen werden. Will man die Formatierung der

Rückgabewerte der *portal_litdbtool*-Funktionen ändern, so muss dies mit dem Website-Verantwortlichen abgesprochen werden.

4.2.22 Content-Typ *Publications*

Während die vorgenannten Datentypen *InProceedings* und *TechReport* beschreiben, wie die Publikationseinträge in der Literaturdatenbank gespeichert sind, bedient sich der Content-Typ *Publications* nur der vom Website-Verantwortlichen erstellten Schnittstelle *portal_litdbtool* zu *LitDB* und gibt eine formatierte Liste aus. Somit wird dieser Content-Typ ausschließlich zur Anzeige verwendet.

Für *Publications* wird ein Template *publications_view* entwickelt, das mit *portal_litdbtool* interagiert. Hierfür ruft es die Funktion *portal_litdbtool.queryByCategory(category)* auf und übergibt den Parameter *category="GXL_Pub"*. Die in *category* beinhaltete Zeichenkette entspricht der Kategorie der Publikationseinträge der Ziel-Website in *LitDB*. Schließlich durchsucht die Schnittstelle die Literaturdatenbank nach Einträgen, die der Kategorie *„GXL_Pub"* zugeordnet wurden und gibt diese als Liste zurück.

Nun können die oben genannten Funktionen von *portal_litdbtool* durch den Content-Typ *Publications* verarbeitet werden. Innerhalb des Templates publications_view werden die Rückgabewerte der Funktionen in HTML formatiert. Abbildung 49 zeigt den Aufbau des Content-Typs *Publications*.

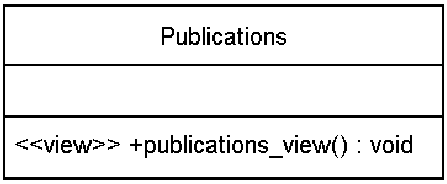

Abbildung 49: Datenmodell zu *Publications*

Die fünf nachfolgend beschriebenen Content-Typen unterscheiden sich von den bisherigen. Während die anderen Content-Typen in Hinsicht auf das Zielsystem Plone sämtlich neu entworfen werden müssen, stellt Plone diese Content-Typen bereits in der Standardinstallation zur Verfügung. Deshalb werden sie hier auch so vorgestellt, wie sie mit ihren Eigenschaften in Plone gespeichert sind.

4.2.23 Content-Typ *Image*

Der Content-Typ *Image* wird für die Verwaltung von Bildern innerhalb der Ziel-Website. Er stellt hierbei weniger einen Content-Typ dar, der für sich alleine steht, sondern wird vielmehr als Eigenschaft in andere Content-Typen integriert.[27] Seine Eigenschaften wie folgt aufgelistet. Zur Unterscheidung zwischen der gespeicherten Datei des Content-Typs *Image* und dem Content-Typ *File*, wurde hier stattdessen für die Datei die Bezeichnung „Datei" verwendet.

- *imageFile*: Datei, [1,1]

Die wesentliche Eigenschaft des Content-Typs Image ist allerdings *imageFile*. Da dort die Bilddatei für diesen Content-Typ abgelegt wird, ist die Belegung von *imageFile* obligatorisch. Der Inhalt dieses Eigenschaftsfelds wird üblicherweise auf allen Webseiten, die den Content-Typ *Image* einbinden, angezeigt. Hierbei ist es wichtig, dass die gespeicherten Dateien tatsächlich Bilddateien sind. Somit soll eine Validierung

[27] Dies kann z.B. im Klassendiagramm zur Darstellung der Content-Typen um die Webseiten *Examples* des Zielsystems in Abbildung 45 nachvollzogen werden. Dort erscheint „Image" als Eigenschaftsdatentyp.

durchgeführt werden, die sicherstellt, dass die im dem *Image*-Content-Typ gespeicherten Dateien die für Bilddateien üblichen Dateiendungen besitzen.

4.2.24 Content-Typ *File*

Der Content-Typ *File* ist rein strukturell dem bereits beschriebenen Content-Typen Image sehr ähnlich. Seine Funktion ist die Verwaltung von Dateien und deren Verfügbarmachung zum Download. Die Eigenschaften von *File* sind:

- *file*: Datei, [1,1]

Hinzu kommt schließlich das Eigenschaftsfeld *file*, dem die zu speichernde Datei zugewiesen wird. Hierbei existieren keine Beschränkungen bezüglich des Dateiformats, wie bei Image. Auf der Ansichtsseite einer Instanz des Content-Typs werden zusätzlich für den Download relevante Informationen wie die Größe der Datei und der Dateityp angegeben.

4.2.25 Content-Typ *Link*

Ebenso wie Image ist auch der Content-Typ *Link* in der Installation von Plone bereits mit eingeschlossen. Durch ihn ist es leicht möglich eine Sammlung von Verweisen zu erstellen und diese zu kommentieren. Unter diesem Content-Typ sollen nicht Hyperlinks verstanden werden, die sich z.B. in HTML-Dokumenten als *a*-Elementen befinden und die auf Website-interne oder –externe Ressourcen verweisen. Dieser Content-Typ dient lediglich zur separaten Sammlung von Hyperlinks, die schließlich, z.B. in Ordnern, thematisch sortiert werden. Hierfür werden die folgenden Eigenschaften für den Content-Typ Link verwendet:

- *URL*: isURL(string), [1,1]

Das Eigenschaftsfeld *URL* enthält die eigentliche Internet-Adresse. Diese wird in der Ansichtsseite als Hyperlink angezeigt und verlinkt direkt auf die in diesem Feld angegebene Adresse. Wichtig ist hier, dass die eingetragene Internet-Adresse auch tatsächlich der Syntax einer URL entspricht. Dieser Anforderung ist durch die Implementierung eines Validators auf URL-Syntax zu entsprechen.

4.2.26 Content-Typ *Document*

Ein besonders nützlicher, weil flexibler Content-Typ, ist *Document*. Dieser wird allgemein für die Verwaltung einfacher Webseiten verwendet. In ihm kann Content gehalten werden, der schwer zu klassifizieren ist oder dessen Klassifizierung als eigener Content-Typ aus bestimmten Gründen, wie zu großer Aufwand, nicht ratsam ist. Er besteht aus folgenden Eigenschaften:

- *bodyText*: isTidyHtmlWithCleanup(richtext), [1,1]

Neben den Eigenschaftsfeldern zur Ausweisung und Beschreibung der Instanz des Content-Typs verfügt *Document* auch über die Eigenschaft *BodyText.* In diesem Feld wird HTML-Code eingetragen, der innerhalb des *body*-Tags der Webseite erscheinen soll. Auf diese Weise können Webseiten schnell und auf einheitliche Weise erstellt werden.

4.2.27 Content-Typ *Topic*

Eine besondere Rolle kommt dem Content-Typ *Topic* zu. Bei ihm handelt es sich nicht um einen Content-Typ im klassischen Sinne. So trägt er kaum eigene Information. Stattdessen sammelt er wie in einer gespeicherten Abfrage den Content der Website und bereitet ihn für den Betrachter auf. *Topic* ist deshalb so wichtig, da er die Aufgabe übernimmt,

thematisch differenzierte Content-Typen wie Tool und FAQ nach Kategorien geordnet darzustellen.

Der Content-Typ *Topic* sucht in den indizierten Metadatenfeldern der Content-Typen nach Informationen. Für seine Suche werden *Topic* Kriterien übergeben. Besonders relevant für das Zielsystem ist das Metadatum *keyword*, das es zur thematischen Strukturierung verwendet wurde. Für die Ergebnisse stellt *Topic* einige Einstellungen für die Anzeige zur Verfügung.

4.2.28 Content-Typ *GXLFolder*

Um einen zentralen Zugriff auf alle relevanten Content-Typen der Zielwebsite zu erhalten, wurde der Content-Typ *GXLFolder* entworfen. Bei ihm handelt es sich um einen in seiner Funktionalität erweiterten Ordner, der als einziger Content-Typ das Hinzufügen der Website-spezifischen Content-Typen ermöglicht. Seine Eigenschaften sind wie folgt:

- *name*: string, [1,1]
- *descriptionHTML*: isTidyHtmlWithCleanup(richtext), [0,1]
- <<ordered>>
- (*addableContent*: FolderHTML)
- (*addableContentExclusive*: Project)
- (*addableContentExclusive*: Listing)
- (*addableContentExclusive*: GXLFolder)

Die ersten drei Eigenschaften sind die gleichen wie bei *FolderHTML*. Im Feld *name* kann *GXLFolder* ein Name zugewiesen werden, während *descriptionHTML* eine ausführlichere Beschreibung des Ordners in HTML-Format bereitstellt. Durch <<*ordered*>> wird festgelegt, dass es sich um einen Ordner handelt, dessen Elemente in einer Reihenfolge geordnet werden können. Die *addableContentExclusive* -Anweisungen für *Project*, *Listing* und *Publications* geben an, dass in *GXLFolder*, und nur dort, Instanzen der genannten Content-Typen eingefügt werden können. Zudem können Instanzen von Content-Typen eingefügt werden, die sich von *FolderHTML* ableiten, wie *ToolSet* und *OrganisationSet*. Diese können aber auch in anderen Containern eingefügt werden als *GXLFolder*, wie die Anweisung *addableContent* aussagt.

Neben den aufgeführten Content-Typen können allerdings auch alle anderen auf der Website installierten Typen eingefügt werden. Dazu gehören neben den Standardtypen wie *Document* und *Image* auch *PloneArticle* und *PloneArticleMultiPage*.

Abbildung 50 stellt den Aufbau von *GXLFolder* und seinen Beziehungen zu anderen Content-Typen dar. Die Verwendung des Stereotyps <<*stub*>> ist eine Konvention für die Modellierung für ArchGenXML, um ein externes Produkt einzubinden.

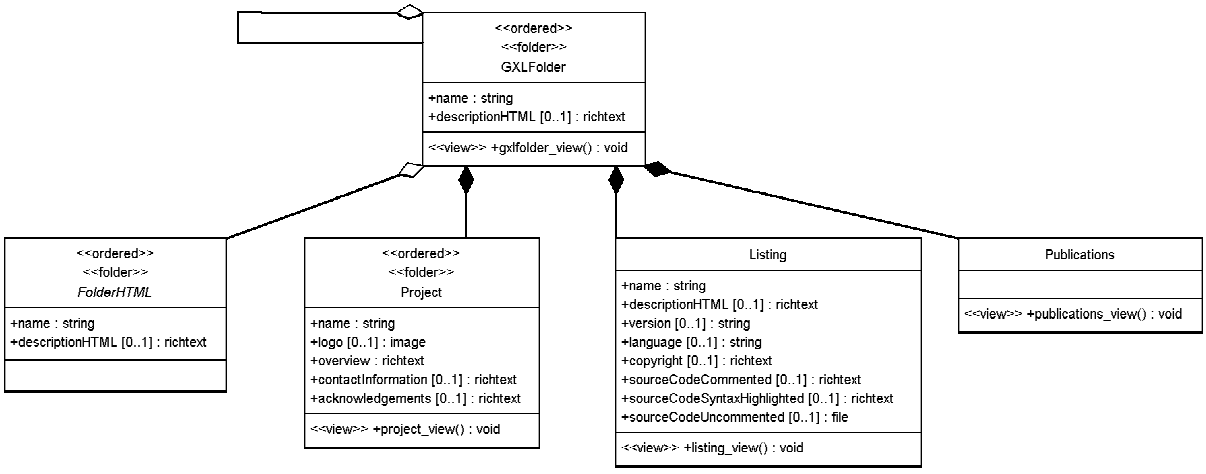

Abbildung 50: Datenmodell zu *GXLFolder*

4.2.29 Abhängigkeiten des Zielsystems zu Fremdprodukten

Die gesamten hier beschriebenen Content-Typen und ihre Beziehungen zueinander werden in dem für die Arbeit erstellten Plone-Produkt *GXL_Website* zusammengefasst. Für die Erzeugung der für die Migration benötigten Strukturen muss somit lediglich das Produkt installiert werden. Eine ganz wesentliche Bedingung ist allerdings das Vorhandensein bestimmter Produkte, auf die *GXL_Website* aufbaut.

GXL_Website greift auf diese Produkte in verschiedenster Weise zu. Hauptsächlich handelt es sich bei diesen Produkten um definierte Content-Typen. Am offensichtlichsten ist dies der Fall bei Content-Typen, die Plone mit seiner derzeitigen Standardinstallation bereits mitliefert. Dazu gehören folgende Content-Typen:

- *Document*
- *Image*
- *File*
- *Topic*

Diese vier Content-Typen wurden in diesem Abschnitt bereits beschrieben. Nach erfolgter Installation von Plone in der aktuellen Version, stehen diese Content-Typen direkt zur Verfügung und müssen nicht durch die Installation eines separaten Produkts integriert werden.

In Zusammenhang mit der Beschreibung des Content-Typs *WebArticle* wurde darauf eingegangen, dass die Funktionalität nicht selbst programmiert wird, sondern durch die Nutzung eines externen Produkts erreicht wird. Hierbei handelt es sich um das Produkt *PloneArticle*, dessen Content-Typen *PloneArticle* und *PloneArticleMultiPage* genutzt werden. Zur Implementierung von des logischen Content-Typs *WebArticle* ist die Installation des Produkts *PloneArticle* erforderlich.

Das letzte Produkt, von dem *GXL_Website* abhängt, ist bisher noch nicht in Erscheinung getreten: *CompoundField*.[28] Der Grund liegt darin, dass das Zielsystem nicht auf einen Content-Typ, sondern eine viel elementarere Funktionalität zurückgreift. *CompoundField* stellt sicher, dass bestimmte Felder der Content-Typen, die eine Multiplizität von [0,*] oder [1,*] aufweisen, überhaupt erst in Plone implementiert werden können. Diese Felder umfassen alle Datentypen, die nicht auf Instanzen anderer Content-Typen verweisen, wie z.B.

[28] Siehe http://plone.org/products/compoundfield

String-, *Text*- oder *Richtext*-Felder. Standardmäßig lässt Plone nur die Zuweisung von Feldern mit einer Multiplizität von [0,1] oder [1,1] zu.

Zur Einbindung der LitDB des Fachbereichs 4 der Universität Koblenz in Plone wurde vom Website-Verantwortlichen das Produkt *Portal_litdbtool* entwickelt. Es verfügt über Funktionen, die die Inhalte der LitDB vorformatiert durch die Nutzung eines Templates ausgeben. Damit muss zur Einbindung dieser Funktionalität auch das Produkt *Portal_litdbtool* installiert werden.

Abbildung 51 verdeutlicht die Abhängigkeiten des Zielsystems von Produkten und Content-Typen außerhalb des Produkts *GXL_Website*. Hierfür wurde die Darstellung eines Paketdiagramms gewählt, wobei die Abhängigkeit durch den gerichteten Pfeil mit dem Stereotyp <<*uses*>> ausdrückt.

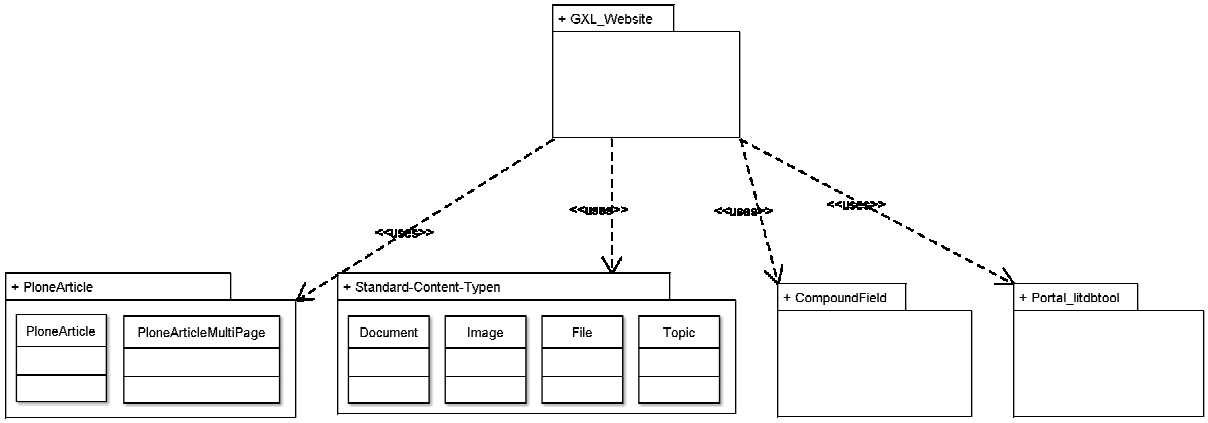

Abbildung 51: Abhängigkeit des Zielsystems von externen Produkten

4.3 Benutzungsschnittstellen entwerfen

Benutzungsschnittstellen beschreiben die Teile eines Systems, über die ein Benutzer mit dem System interagiert. Im Kontext von Websites werden unter Benutzungsschnittstellen die dem Betrachter zur Verfügung gestellten Webseiten verstanden. Beim Entwurf der Benutzungsschnittstellen kommt es insbesondere auf Benutzerfreundlichkeit und die Nachvollziehbarkeit der Navigationspfade zwischen den einzelnen Oberflächen an [Ackermann (2005), S. 198].

Für die Bearbeitung der Inhalte der Website kommen andere Benutzeroberflächen zum Einsatz. Im Legacy-System konnten die Webseiten nur durch direkte Bearbeitung der HTML-, bzw. XML- und XSL-Dokumente, sowie der für die Generierung der Webseiten zuständigen *Makefiles* verändert werden. Bei der Zielwebsite entfallen solche Bearbeitungsschritte, da Plone über ein eigenes System zur Veränderung der Inhalte verfügt. Dieses wird später in diesem Abschnitt erläutert.

4.3.1 Navigationspfade der Ziel-Website

Bereits im Abschnitt 3 zur Legacy-Analyse wurden die Website und die Verknüpfungen der Webseiten untereinander beschrieben. Ähnlich wie dort werden nun an dieser Stelle die Design-Entscheidungen der Websitestruktur im Zielsystem dargestellt. Zur Beschreibung der Navigationsstruktur von Websites gibt es verschiedene Modellierungsalternativen. Manche orientieren sich an semantisch veränderten Klassendiagrammen (siehe z.B. [Kappel et al. (2004), S. 49 ff.] oder [AGAS (2003), S. 18 ff.]). Diese Art der Modellierung wurde in Abschnitt 3 zur Legacy-Analyse verfolgt. Als Grundlage der Modellierung der Navigationsstruktur des Ziel-Design wird eine andere Notation gewählt, wie sie in [Gipp

(2006), S. 115 ff.] beschrieben wird. Es wird hierbei nicht nur die hierarchische und beliebige Verlinkung von Webseiten berücksichtigt, sondern ermöglicht auch die Beschreibung der Dynamik der Webseiten, wie sie im Zielsystem vorliegt.

Die Notation baut auf die grundlegenden Elemente Webseiten und Verknüpfungen unter diesen auf. Bei den Verknüpfungen wird in [Gipp (2006), S. 115] primäre und sekundäre Navigationsstruktur unterschieden. Die primäre Navigationsstruktur lässt sich aus den Gegebenheiten der hierarchischen Ordnerstruktur der Website herleiten. Dem gegenüber umfasst die sekundäre Navigationsstruktur sämtliche beliebige Verlinkungen zwischen den Webseiten einer Website. Die Modellierung dieser Elemente ist in Abbildung 52 dargestellt.

Abbildung 52: Darstellung der Navigationsstruktur

Im Kontext des Zielsystems Plone ist eine genauere Betrachtung der sekundären Navigationsstruktur sinnvoll. Durch die sekundären Links werden beliebige Verlinkungen und Zusammenhänge zwischen Webseiten dargestellt, die in keiner hierarchischen Beziehung zueinander stehen. Darunter fallen im klassischen Sinn *Hyperlinks*, die von einer auf eine andere Webseite verweisen. Durch Hyperlinks werden also allgemeine Querbezüge hergestellt.

Zwischen den Content-Typen existiert aber auch eine Vielzahl von direkten Referenzen wie zwischen *Project* und *Conference* oder *Tool* und *Person*. Der Unterschied zu einfachen Hyperlinks ist allerdings, dass die Referenz intern im Zielsystem Plone existiert. Damit kann nicht nur auf das Objekt verwiesen, sondern auch sehr einfach auf die Daten zurückgegriffen werden. Ein solcher Navigationslink ist semantisch mächtiger als ein einfacher Hyperlink zwischen Webseiten. Deshalb wird das Modell für die sekundäre Navigationsstruktur erweitert. Abbildung 53 stellt die Repräsentation von *Plone-internen* Links und einfachen *Hyperlinks* dar.

Abbildung 53: Darstellung der Unterteilung der sekundären Navigationsstruktur

Die Webseiten wiederum werden nach ihrer Art der Dynamik differenziert. In [Gipp (2006), S. 116 / 117] werden neben statischen Seiten auch Seiten mit dynamisch generiertem Content, Web-Formulare mit anschließend ausgeführten Skripten und so genannte *virtuelle Seiten* aufgeführt. Virtuelle Seiten sind dynamisch generierten Seiten sehr ähnlich, unterscheiden sich allerdings dadurch, dass sie vollständig durch ein Skript und an Hand bestimmter Parameter berechnet werden. Ebenso verfügen sie nicht über eine vordefinierte Adressierung [Gipp (2006), S. 117]. Ein Beispiel für solche Seiten sind Suchmaschinenanfragen, die erst durch die Übergabe der Suchparameter die Webseite erzeugt. Die für die Modellierung von Webseiten verwendeten Elemente sind in Abbildung 54 angezeigt.

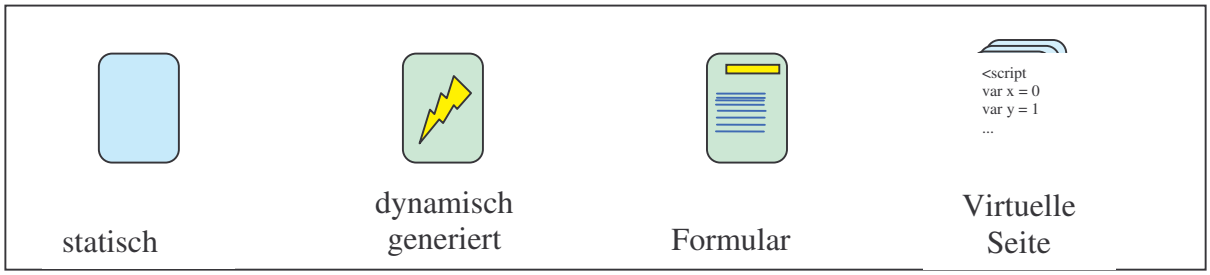

Abbildung 54: Darstellung der Webseitenarten

Ausgehend von diesen Modellierungselementen wurde die Navigationsstruktur für die zu migrierende Website erstellt, wie sie in Abbildung 55 zu sehen ist. Das Diagramm beschreibt die Struktur der Webseiten, auf denen schließlich die Inhalte der verschiedenen Content-Typen verteilt werden. Es orientiert sich an der bei der zu migrierenden Website vorgefundenen Navigationsstruktur (siehe Abbildung 25). Im Vergleich zum Navigationsnetz der Legacy-Website sind noch einige Navigationslinks hinzugekommen. Das liegt vor allem darin begründet, dass die zuvor webseiteninternen Content-Strukturen in den Webseiten *Background* und *Tool Catalogue* in eigene Content-Typen expliziert wurden.

Bei der Wahl der Webseitenarten wurden zwischen statischen, dynamischen und virtuellen Seiten unterschieden. Im Zusammenhang mit Plone gibt es streng genommen keine statischen Webseiten. Sie werden vom Content Management System bei jedem Aufruf neu generiert. In der Navigationsstruktur steht diese Art von Webseiten für Seiten, deren Inhalte bei Aufruf in den Daten des Content-Typs bereits feststehen. Dynamische Webseiten hingegen stehen für Webseiten, die entweder Ordner sind, deren genaues Erscheinungsbild durch Referenzen zu anderen Instanzen von Content-Typen gebildet wird, oder bei denen Skripte zur Anzeige ausgeführt werden. Virtuelle Seiten stehen für gespeicherte Abfragen, wie sie beim Content-Typ *Topic* vorkommen.

Zudem wurden Instanzen von Content-Typen, die sich in Containern befinden, separat dargestellt (erkennbar am Zusatz *(id)* in der Bezeichnung). Bei der Zusammenfassung von Webseiten werden zwei Symbole hintereinander angeordnet, um auf die Existenz mehrerer Seiten zu verweisen. In der Navigationsstruktur des Ziel-Designs können die Navigationsbereiche aus der Legacy-Website gut nachvollzogen werden. Sie folgen in der Hierarchie der Website direkt auf die Homepage des GXL-Bereichs.

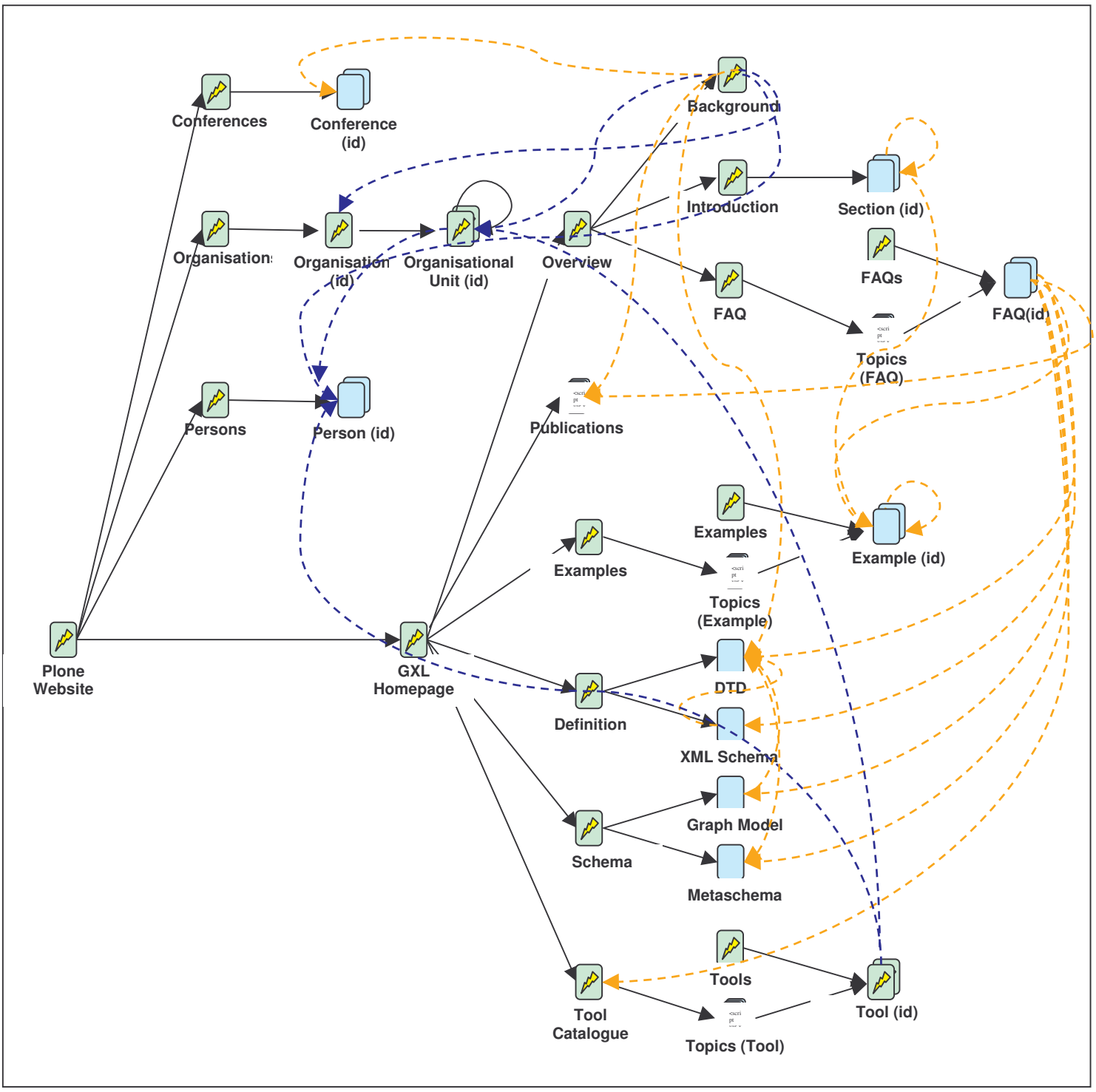

Abbildung 55: Navigationsstruktur der zu migrierenden Website

Neben der Angabe der Navigationsstruktur kann mit Abbildung 55 zusätzlich eine Aussage über die Verteilung der Content-Typen in den Navigationsbereichen gemacht werden. Diese Information ist zur Vorbereitung von Layout-Entscheidungen und der Verdeutlichung der Verknüpfungen, die zwischen den Content-Typen und den Webseiten bestehen, notwendig. Neben den offensichtlichen hierarchischen Links der primären Navigationsstruktur, dienen auch die sekundären, Plone-internen Links als Informationsquelle. Die einfachen Hyperlinks werden ignoriert. Die Container *Organisations*, *Persons* und *Conferences* werden nicht als eigene Navigationsbereiche definiert, da sie lediglich von Content-Typen der Ziel-Website aufgerufen werden. Zu bemerken ist auch die Isolation der Container *FAQs*, *Examples* und *Tools*. Ihre einzige Funktion besteht darin, die jeweiligen Objekte zu speichern. Die Navigation zu diesen Objekten wird durch die gespeicherten Abfragen von *Topics(FAQ)*, *Topics(Example)* und *Topics(Tool)* übernommen, die entsprechende Kriterien enthalten. Tabelle 4 ordnet den Navigationsbereichen Content-Typen zu, die innerhalb des Bereiches oder durch Plone-interne Verweise vorkommen.

Navigationsbereich	**Enthaltene Content-Typen**
Overview	*Project* *Organisation* *OrganisationalUnit* *Person* *Conference* *PloneArticleMuliPage* *PloneArticle* *FAQSet* *FAQ* *Topic* *Publications*
Examples	*ExampleSet* *Example* *Topic*
Definition	*Listing*
Schema	*Document*
Tool Catalogue	*ToolSet* *Tool* *Topic*

Tabelle 4: Verteilung der Content-Typen über die Webseiten

4.3.2 Webseiten und deren Aufbau

Die Navigationsstruktur beschreibt das große Bild, das sich von der zu migrierenden Website machen lässt. Hierin werden durch die Einbeziehung der Verknüpfungsabhängigkeiten zwischen den einzelnen Webseiten sowohl die zu berücksichtigenden Verlinkungen modelliert als auch, eher indirekt, die hierarchische Anordnung der Webseiten. Im nächsten Schritt werden die bisher erarbeiteten Designentscheidungen zu den Content-Typen und der Navigationsstruktur durch die Beschreibung der Webseiten über die in ihnen enthaltenen Content-Typen zusammengebracht.

Das Rahmenwerk *Archetypes* (siehe Abschnitt 1.5.5) erzeugt zur Anzeige von Content-Typen eine standardmäßige Ansicht, bei der alle Felder des Content-Typen in der Reihenfolge, wie sie definiert wurden, aufgelistet werden. Die dabei generierte Sicht auf die Content-Typen ist sehr einfach. Abbildung 56 zeigt einen exemplarischen Screenshot der standardmäßigen Archetypes-Darstellung einer Instanz des Content-Typs *Tool*.

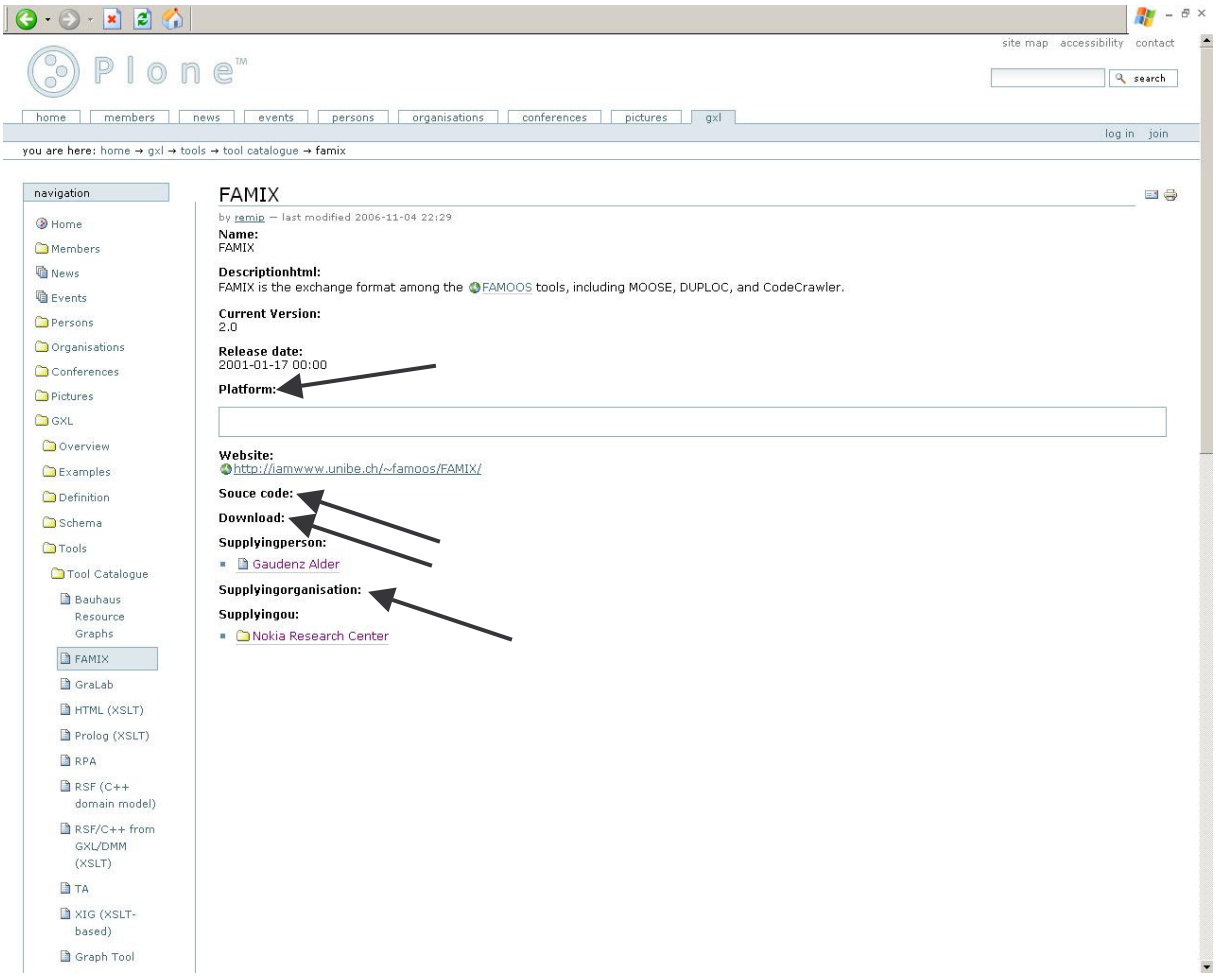

Abbildung 56: Standardansicht von *Tool* mit Archetypes

Es ist zu erkennen, dass die Standardansicht von Archetypes höheren Design-Ansprüchen nicht genügt. In dieser Ansicht werden der Reihe nach alle Felder ausgegeben und mit der ihnen zugewiesenen Beschriftung versehen, unabhängig davon, ob dieses Feld überhaupt einen Eintrag aufweist. Ersichtlich ist dieses Verhalten in Abbildung 56 u.a. bei den Feldern mit der Beschriftung *Platform* (*Tool.platform*) und *Source Code* (*Tool.sourceCodeURL*), die neben anderen mit Pfeilen gekennzeichnet wurden.

Dennoch ist sie auf Grund ihrer Einfachheit auch übersichtlich und zweckmäßig zur Anzeige von Content-Typen. Deshalb wurde beschlossen, im Rahmen der Migration keine umfassenden Anpassungen der Darstellung auf Ebene der Content-Typen durchzuführen. Stattdessen sollte die Standardansicht, die Archetypes zur Verfügung stellt, geringfügig verändert werden. Als wesentliches Problem dieser Ansicht wurde zusammen mit dem Website-Betreiber die Anzeige nicht befüllter Felder ausgemacht.

Zur Umsetzung wurde auf den im Abschnitt 1.5.1 beschriebenen METAL-Mechanismus zurückgegriffen. Jeder auf Archetypes basierender Content-Typ wird, wenn kein eigenes Template definiert wurde, durch das Archetypes-Ansichtstemplate angezeigt. Wird aber für einen Content-Typen ein eigenes *view*-Template erstellt, so kann durch die Anweisung *metal:define-macro* das Makro aus dem Archetypes-Template für diesen Content-Typen überschrieben werden. Es gelten dann nur noch die Festlegungen, die in dem *view*-Template angegeben werden. Abbildung 57 zeigt das selbst definierte Template *gxl_view*, das die erforderlichen Erweiterungen implementiert.

```
<!-- Namespace-Zuweisungen, darunter für tal und metal -->
<html xmlns="http://www.w3.org/1999/xhtml"
   xmlns:tal="http://xml.zope.org/namespaces/tal"
   xmlns:metal="http://xml.zope.org/namespaces/metal"
   xmlns:i18n="http://xml.zope.org/namespaces/i18n"
   i18n:domain="plone">
 <head><title></title></head>
 <body>

<!-- Makro zur Einbindung der CSS-Datei zur Formatierung von Content-Typen der GXL-Website
-->
  <metal:css_macro metal:define-macro="css">
   <link rel="stylesheet" type="text/css" href="#"
      tal:attributes="href string:gxlstyle.css" />
  </metal:css_macro>

<!-- Makro für die modifizierte Ansicht des Contents -->
  <div metal:define-macro="body"
     tal:define="field_macro field_macro | here/widgets/field/macros/view;">

<!-- es werden nur Feldinhalte ausgegeben, die tatsächlich belegt sind -->
   <tal:field_repeat tal:repeat="field python:here.Schema().filterFields(isMetadata=0)">
     <tal:if_visible define="mode string:view; accessor python: field.getAccessor(here);
          visState python:field.widget.isVisible(here, mode);
          visCondition python:field.widget.testCondition(here, portal, template);

<!-- Bedingung, dass nur Felder mit echtem Inhalt angezeigt werden
  accessor() ist erfüllt, wenn das Feld über Content verfügt
  accessor() != ['', '', '', '', ''] ist erfüllt, wenn ein Feld mit Text-Widget und mit
  [0,*] oder [1,*]-Multiplizität mindestens einen Eintrag besitzt
  accessor() != [None, None, None, None, None]) ist erfüllt, wenn ein Objekt-Feld (z.B.
  Image) mit [0,*] oder [1,*]-Multiplizität mindestens einen Eintrag besitzt
  accessor() != '<p></p> ist erfüllt, wenn ein Feld mit RichText-Widget befüllt ist
-->
          visWanted python: (accessor()) and (accessor() != ['', '', '', '', '']) and
(accessor() != [None, None, None, None, None]) and (accessor() != '<p></p>');"
       condition="python:visState == 'visible' and
            visCondition and visWanted">
    <metal:use_field use-macro="field_macro" />
  </tal:if_visible>
   </tal:field_repeat>
  </div>

<!-- Makro zur Anzeige des Labels eines Felds -->
  <metal:label_macro metal:define-macro="label">
    <metal:default_label metal:use-macro="here/widgets/field/macros/label" />
  </metal:label_macro>

 </body>
</html>
```

Abbildung 57: Template *gxl_view* zur allgemeinen Definition der modifizierten Anzeige mit Archetypes

Die im Template *gxl_view* gemachten Makro-Definitionen überschreiben das Standardansichts-Template von Archetypes und bewirken, dass nur noch Felder angezeigt werden, die tatsächlich befüllt werden. Um diesen Effekt bei allen anderen Content-Typen der Website zu aktivieren, muss ein *view*-Template jeden Content-Typen erstellt und die oben definierten Makros durch *metal:use-macro* eingebunden werden. Abbildung 58 zeigt dies exemplarisch für das Template des Content-Typs *Tool*.

```
<html xmlns="http://www.w3.org/1999/xhtml"
   xmlns:tal="http://xml.zope.org/namespaces/tal"
   xmlns:metal="http://xml.zope.org/namespaces/metal"
   xmlns:i18n="http://xml.zope.org/namespaces/i18n"
   i18n:domain="plone">
 <head><title></title></head>
 <body>

  <metal:css_macro metal:define-macro="css">
   <link rel="stylesheet" type="text/css" href="#"
      tal:attributes="href string:gxlstyle.css" />
  </metal:css_macro>

<!-- Einbindung des body-Makros von gxl_view -->
  <div metal:define-macro="body">
    <div metal:use-macro="here/gxl_view/macros/body" />
  </div>

<!-- Einbindung des label-Makros von gxl_view -->
  <metal:label_macro metal:define-macro="label">
    <div metal:use-macro="here/gxl_view/macros/label" />
  </metal:label_macro>

 </body>
</html>
```

Abbildung 58: Template zur modifizierten Anzeige einer Instanz von *Tool*

Die Umsetzung dieser Änderungen kann auf der migrierten GXL-Website auf der mitgelieferten CD nachvollzogen werden. Weitergehende Änderungen des Designs zur Anzeige der Content-Typen der neuen GXL-Website sind damit nicht ausgeschlossen. Die Entwicklung eines elaborierten Designs mit einer individuellen Darstellung für jeden einzelnen Content-Typen wird in ein Projekt im Anschluss an die Migration ausgelagert (siehe Abschnitt 2.2.2).

4.4 Datenbank(en) entwerfen

Der Entwurf des Datenmodells zur Abbildung auf eine Datenbank nimmt eine wichtige Position bei der Durchführung einer Migration ein. Die Verwaltung der Daten in Form einer objektorientierten Datenbank im Zielsystem wird bereits durch Plone bzw. Zope übernommen (siehe Abschnitt 1.5). Somit entfällt der Entwurf einer eigenen Datenbank, da das Datenmodell durch die Definition der Content-Typen im Zielsystem bereits verwaltet wird.

4.5 Programme entwerfen

Die im Legacy-System enthaltene Funktionalität wurde durch die Verwendung des Programms *make* und dessen Konfiguration über *Makefiles* erreicht. Das Programm wurde zur konsistenten Verwaltung der Webseiten verwendet, da es über die Definition von Abhängigkeiten eine komfortable Aktualisierung der Inhalte ermöglichte (vgl. Abschnitt 3.3).

Die für die Migration gewählte Strategie einer Datenkonversion definiert. Daraus folgt, dass die alten Programme des Legacy-Systems, wenn überhaupt, nur in sehr geringem Umfang in das Zielsystem übertragen werden sollen. Lediglich die Wirkungsweise der Programme wird im Zielsystem abgebildet, die Programmierung muss völlig neu durchgeführt werden.

Dennoch ist der Aufwand für den Entwurf der Programme im Zielsystem Plone denkbar gering. Plone umfasst eine komplexe Content Management-Komponente und übernimmt somit bereits die gesamte Funktionalität, die zuvor durch *make* und die *Makefiles* geleistet wurde. Die Aktivitätsgruppe „Programme entwerfen“ wird deshalb nicht weiter berücksichtigt.

4.6 Transformationen entwerfen

Nachdem das Ziel-Design in den vorigen Schritten ausführlich beschrieben wurde, dient der Prozess „Transformationen entwerfen“ der Formulierung von Regeln, die Querverweise zwischen Legacy- und Zielsystem herstellen [Ackermann (2005), S. 199]. Diese Regeln werden auch als *Transformationsregeln* bezeichnet. Unter Berücksichtigung der Wiederverwendbarkeit der Elemente aus dem Alt-System kann es vorkommen, dass das bereits spezifizierte Design zur Vereinfachung der Transformation angepasst wird. Die formale Spezifikation der Transformationsregeln kann später als Grundlage zur Erstellung von Transformationswerkzeugen zur automatisierten Durchführung der Migration verwendet werden [Ackermann (2005), S. 200].

4.6.1 Vorgehen und Berücksichtigung des Zielsystems „Website“

Wie bereits aus der Analyse des Legacy-Systems und dem Ziel-Design hervorgegangen ist, handelt es sich bei dieser Migration um ein datenzentriertes Verfahren. Zwar existiert mit dem Teilbereich „Downloads“ auch eine Komponente mit Funktionalität, diese wird allerdings durch Verlinkung gekapselt und wirkt sich nicht wesentlich auf die Migration aus. Viel wichtiger ist hingegen die Übertragung der Inhalte der Website in eine leicht wartbare Struktur im Zielsystem. Die Technologien, die hierfür zum Einsatz kommen, dienen deshalb der Verarbeitung der Inhalte und nicht der Funktionalität.

Zum Auslesen der Inhalte der zu migrierenden Website bieten sich die Nutzung von Parsern an. Da die XML-Dokumente selbst sämtliche Daten ihrer zugehörigen Webseiten enthalten und ihre Struktur durch ihre DTDs transparent ist, können die Dokumente geparst und leicht verarbeitet werden.

Etwas komplizierter ist die Verarbeitung der Inhalte, die lediglich auf HTML-Webseiten gespeichert ist. Hier ist es anfangs nicht klar, an welcher Stelle sich die Inhalte befinden. Zur Auswertung dieser Webseiten ist zunächst die Definition der Position innerhalb des Dokuments notwendig. Um die definierten Stellen zu finden, kann auf Parser oder reguläre Ausdrücke zurückgegriffen werden. Bei der Programmierung der Transformation wurden reguläre Ausdrücke zum Auffinden der Bereiche verwendet.

4.6.2 Verarbeitung von Verweisen

Ein besonderes Problem bei der Transformation von Webseiten ergibt sich durch im Quelltext enthaltene Referenzen. Hierzu gehören vor allem durch das Attribut *href* gekennzeichnete Links innerhalb eines *a*-Elements, aber auch Quellreferenzen wie z.B. das Attribut *src* innerhalb eines *img*-Elements. Dort werden Adressen von anderen Objekten im Internet, wie z.B. anderen Webseiten oder Bilddateien, gespeichert. Externe Links, also Links, die nicht auf Objekte innerhalb einer Website verweisen, sind unproblematisch, da das Ziel der Referenz nach der Migration dasselbe ist wie zuvor. Wird allerdings bei einer Referenz auf ein Objekt innerhalb der Website verwiesen, so ist es wahrscheinlich, dass durch strukturelle Änderungen der Website, oder gar die Veränderung der Internet-Adresse, die dort angegebene Referenz nicht mehr aktuell ist.

Um zu gewährleisten, dass alle internen Links der GXL-Website auch nach der Migration noch auf die richtigen Objekte verweisen, muss ein Abgleich der Adressen stattfinden. Hierfür werden die HTML-Inhalte, die in das Zielsystem übertragen werden sollen, während der Transformation auf ihre Referenzen überprüft und diese, soweit möglich, automatisch aktualisiert. Dies kann durch eine Tabelle erreicht werden, die die zu ersetzenden Bestandteile der Internet-Adressen auflistet und die Ersetzungen zuordnet.

Bei diesen internen Links unterscheidet man zwei Arten von Referenzierungen. Zum einen gibt es *absolute Links*, die die vollständige Adresse ihrer Referenz enthalten. Ein Beispiel hierfür ist *http://www.gupro.de/GXL*, wobei durch Angabe dieser Adresse direkt ersichtlich ist, wo sich das referenzierte Objekt befindet. Zum anderen existieren *relative Links*, die statt der vollständigen Adresse lediglich den Pfad angeben, von dem man relativ zur aktuellen Adresse der Webseite zum Ziel findet. Übliche Adressierungen sind z.B.:

- *background.html*: um auf ein Dokument zu verweisen, dass sich im gleichen Ordner wie das aktuelle Dokument befindet
- *../background.html*: um ein Dokument in einem übergeordneten Verzeichnis zu referenzieren

Im Rahmen der Transformation ist die Kenntnis dieser beiden Sorten von Referenzen insofern wichtig, als dass eine Funktion existieren muss, die eine Umwandlung der relativen in absolute Links durchführt. Damit ist schließlich auch die eindeutige Zuweisung für Aktualisierung der Referenzen möglich.

Im Fall der zu migrierenden Website sind die einzelnen statischen Webseiten des Alt-Systems der hauptsächliche Gegenstand der Transformation. Sie dienen als Eingangswerte zur Übertragung von Daten zur Befüllung der in Abschnitt 4.2 definierten Content-Typen. Deshalb wird dieser Prozess ausgehend von den Transformationsregeln beschrieben, die für jede Webseite formuliert werden müssen. Diese Beschreibungen enthalten nicht nur die Nennung der betroffenen Content-Typen, sondern auch eine implementierungsnahe detaillierte Beschreibung der Transformationsabläufe in Wort und Bild.

Um die Abläufe innerhalb der Transformationsregeln besser zu visualisieren, wurden diese durch Aktivitätsdiagramme modelliert. Hierbei werden besonders die Kontrollstrukturen deutlich, die bei der Umwandlung der Webseiten notwendig sind. Die Aktivitätsdiagramme stellen allerdings aus Gründen der Übersichtlichkeit die Transformationsregeln auf einem abstrahierten Niveau dar. Ihr Zweck ist es nicht, den genauen Ablauf jeder einzelnen Transformation zu zeigen.

Die aufgeführten Transformationen wurden in Python implementiert. Bei der Installation des Produkts *GXL_Website* werden die meisten der beschriebenen Transformationsroutinen automatisch ausgeführt. Der Quelltext für die Transformationen ist in der beigefügten CD zu finden.

4.6.3 Transformation der Navigationsbereiche

Es wurde beschlossen, die Navigationsbereiche, wie sie auf der zu migrierenden Website zu finden sind, weitestgehend in die Ziel-Website zu übertragen. Lediglich die Beispiele wurden wegen ihrer herausragenden Stellung zusätzlich als eigener Navigationsbereich der Ziel-Website definiert. Somit sind die zu transformierenden Navigationsbereiche: *Overview*, *Examples*, *Definition*, *Schemas*, *Tools* und *Advance*. Ziel ist, dass die genannten Navigationsbereiche als eigene Navigationspunkte in der Ziel-Website erscheinen.

In Plone kann dieses Problem dadurch gelöst werden, dass die Elemente hierarchisch in der nächsten Unterstufe zum Oberordner GXL stehen. Diese werden dann in der Navigationsleiste automatisch angezeigt. Um die Bereiche in die Navigation einzubeziehen, werden Instanzen des Content-Typs *GXLFolder* erzeugt. Diese Ordner können als Container für Content-Typen des Produkts *GXL_Website* verwendet werden und erscheinen dann in der Navigationsleiste.

Die Umsetzung der Navigationsbereiche in das Zielsystem ist in Abbildung 59 dargestellt. Hierbei, wie auch bei allen weiteren Abbildungen zu den Transformationsregeln, besitzen die dort dargestellten Elemente folgende Semantik: die linken Kästchen stehen für das Legacy-

System, das je nach Bereich aus Webseiten, XML-, XSL-Dokumenten oder, wie hier, aus logischen Bestandteilen wie den Navigationsbereichen besteht. Wo die Darstellung es zulässt, führen Pfeile in ein Kästchen mit der Beschriftung „Transformation“, das die Anwendung der Transformationsregeln versinnbildlicht.

Von diesem Kästchen führen die Pfeile zu den Bestandteilen des Zielsystems, die hier in der Regel allgemeiner als Content-Typen und nicht als Instanzen zu verstehen sind. Die Transformation der Navigationsbereiche bildet hier eine Ausnahme. Von diesen Content-Typen können mehrere Instanzen entstehen. Die fett gedruckten Eigenschaftsfelder der Content-Typen zeigen an, dass sie durch die Transformation tatsächlich mit Werten befüllt werden, während die anderen zunächst leer bleiben.

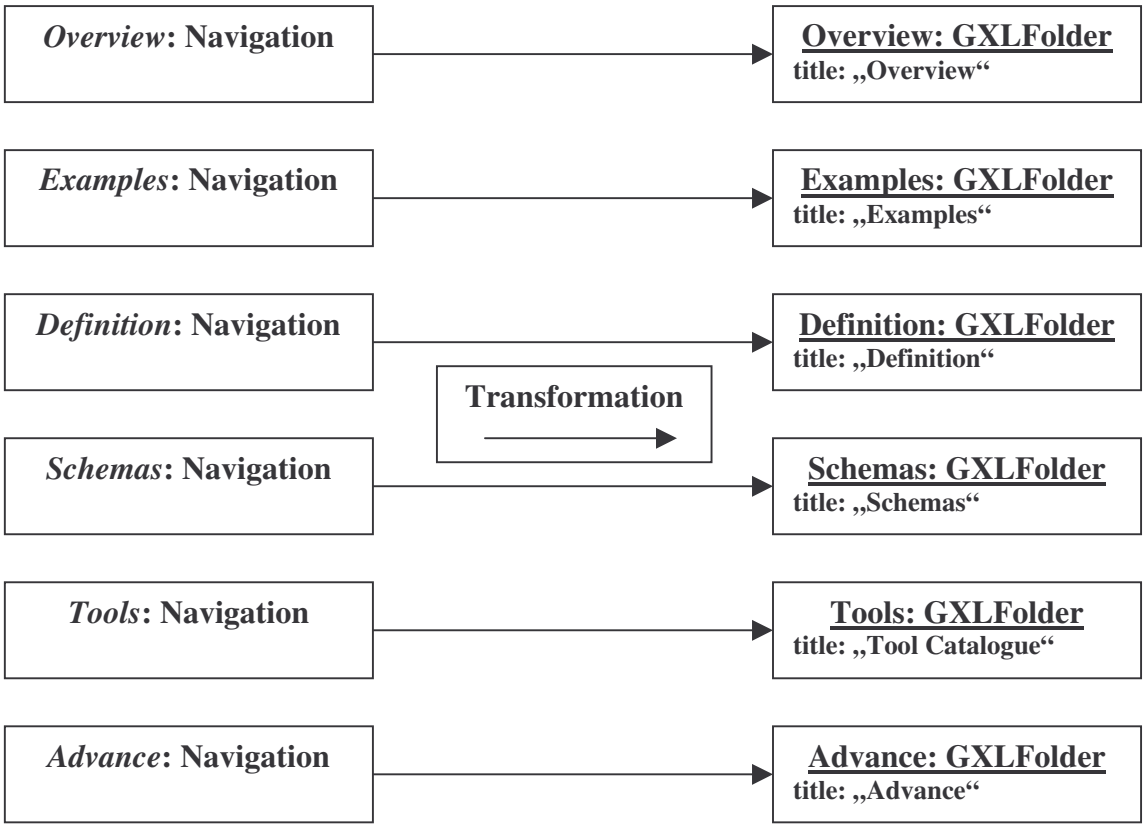

Abbildung 59: Abbildung der Navigationsbereiche in das Zielsystem

Ablauf der Transformation (siehe Abbildung 60)

Aktivität 1 und 2

Eine Übersicht für die vorhandenen Navigationsbereiche befindet sich, außer im Bereich der GXL-Beispiele, auf jeder Webseite der zu migrierenden Website. Deshalb kann die Webseite zur Extraktion der Navigationsbereiche recht frei gewählt werden (Aktivität 1). In Aktivität 2 wird schließlich die Navigationsleiste gesucht und einzelne Navigationsbereiche, die an Hand ihrer Formatierung erkannt werden können, ausgelesen.

Aktivität 3

Für jeden einzelnen Navigationsbereich wird nun eine Instanz des Content-Typs *GXLFolder* erstellt und sein Eigenschaftsfeld *GXLFolder.title* mit dem Namen des Navigationsbereichs ausgefüllt. Damit ist der Nachvollzug der Struktur der Navigationsbereiche im Zielsystem gewährleistet. Die Transformation terminiert, sobald alle Navigationsbereiche durchlaufen wurden.

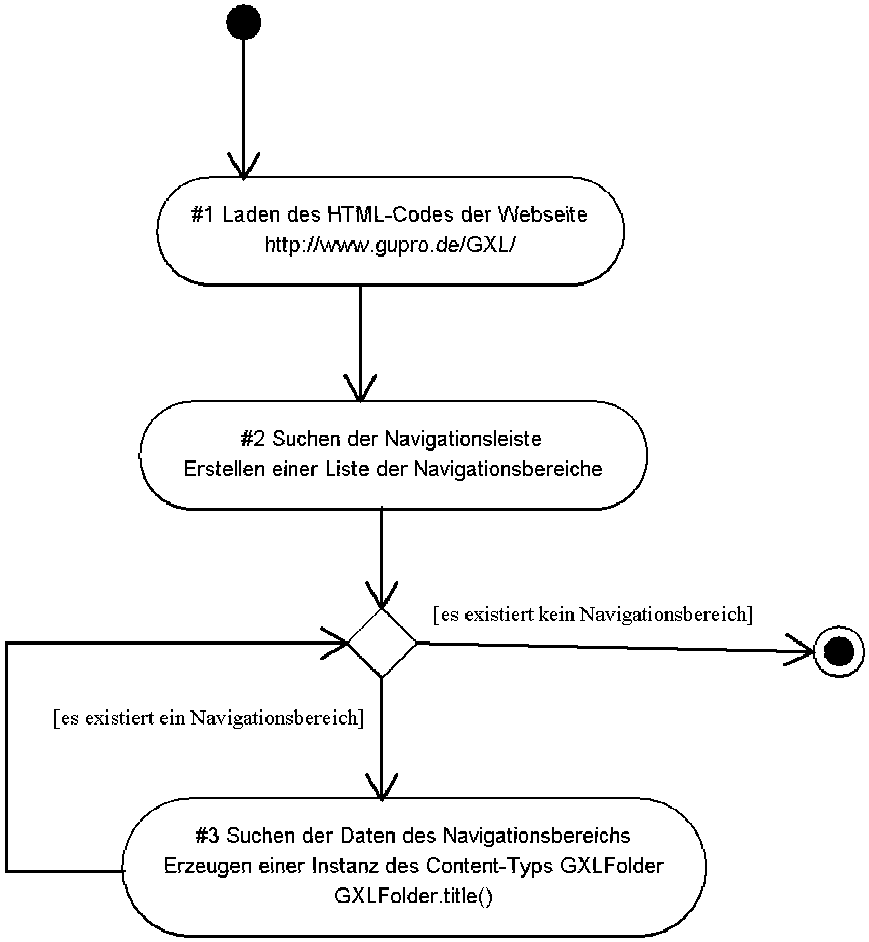

Abbildung 60: Aktivitätsdiagramm für Transformationsregeln für die Navigationsbereiche

4.6.4 Transformation der Webseite *Background*

Die Webseite *Background* ist die Ausgangsseite der zu migrierenden Website. Sie ist in dem Navigationsbereich *Overview* beinhaltet. In ihr sind viele Daten enthalten, die zur Zuweisung für verschiedene Content-Typen verwendet werden. Hierbei handelt es sich insbesondere um die Content-Typen *Project* und *Conference*, als auch um *Organisation* und *OrganisationalUnit*.

Fast alle verfügbaren Informationen zur Instanziierung des Content-Typs *Project* sind auf dieser Webseite enthalten. Dort werden neben den nur zu Project gehörenden Eigenschaften auch Referenzen zu Instanzen anderer Content-Typen gebildet. Hierzu gehören *Conference* und *Organisation.* Auch bei *Conference* sind die Informationen sämtlich auf dieser Webseite enthalten. Nur teilweise werden die beiden anderen Content-Typen befüllt. Die genaue Belegung der Content-Typen durch die Transformationen der Webseite ist in Abbildung 61 zusammengefasst. Hier sind die Content-Typen mit allen Feldern aufgelistet. Die Felder, die von der Transformation betroffen sind, sind fett gedruckt.

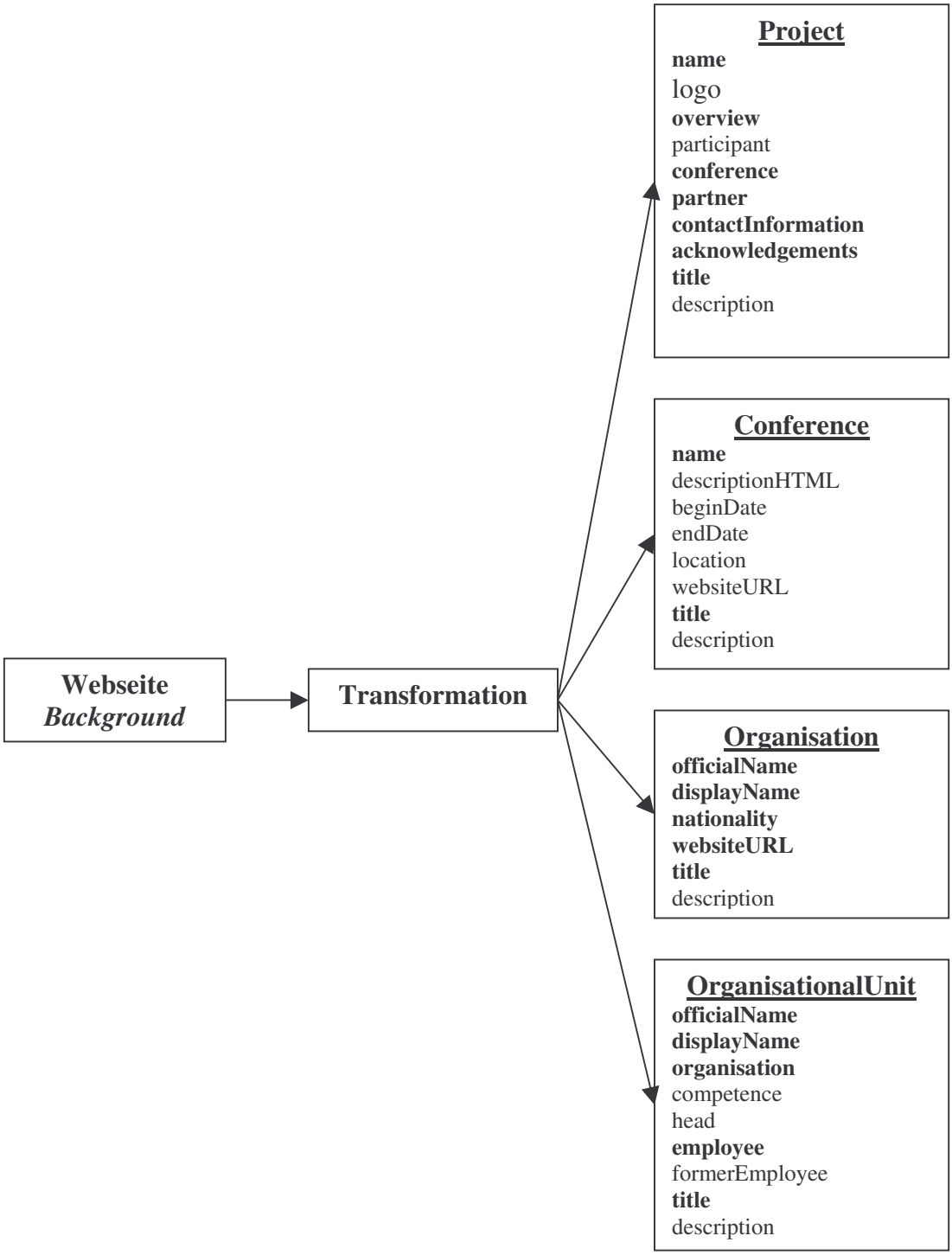

Abbildung 61: Belegung der Content-Typen durch die Webseite *Background*

Das genaue Ablaufmodell der Transformation der Webseite „Background“ ist in Abbildung 62 als Aktivitätsdiagramm dargestellt. Dort ist die Abfolge der Transformationsschritte der Daten der Webseite zu den Content-Typen wichtig. In den folgenden Absätzen sind diese genauer beschrieben.

Ablauf der Transformation (siehe Abbildung 62)
Aktivitäten 1 und 2

Zunächst muss, wie in Aktivität 1 gezeigt, die betroffene Webseite als HTML-Text geladen werden. Anschließend wird in Aktivität 2 nach dem einleitenden Text zum Projekt GXL gesucht. Dieser wird durch die Überschrift „Background“ und seine Formatierung

erkannt. Wurde eine Einleitung gefunden, so kann eine Instanz des Content-Typs *Project* erzeugt werden. Wird keine Einleitung gefunden, so muss davon ausgegangen werden, dass beim Auslesen ein Fehler unterlaufen ist und die Transformationsregel bricht den Vorgang ab. Die Einleitung wird schließlich als Eigenschaft *Project.overview* gespeichert.

Aktivität 3

Auf der Webseite erscheint anschließend eine Übersicht zu den Konferenzen, die dem Projekt GXL zugeordnet werden können. Deshalb wird in Aktivität 3 nach dem entsprechenden Abschnitt der Webseite gesucht. Er wird durch die Zeichenkette „Presentations and Discussion of former GXL versions“ und seine Formatierung gekennzeichnet. Wird der Bereich der Konferenzen gefunden, so wird nach einer einzelnen Konferenz gesucht. Diese wird dadurch erkannt, dass sie innerhalb eines Absatzes den ersten Hyperlink aufweist. Wird eine Konferenz gefunden, so wird in eine Instanz des Content-Typs *Conference* generiert und die Eigenschaftsfelder *Conference.name* und *.title* belegt. Zudem kann über die aufgefundene Konferenz eine Referenz zwischen ihr und dem aktuellen Projekt hergestellt werden, indem sie zu *Project.conference* hinzugefügt wird. Die Suche nach weiteren Konferenzen wird so lange durchgeführt, bis das Ende des Bereichs der Konferenzen erreicht ist.

Aktivität 4

Nachdem die Konferenzdaten aus der Webseite extrahiert wurden, werden nun die GXL-Partnerorganisationen gesucht. Der Beginn des Bereichs der Partnerorganisationen ist durch der Zeichenkette „GXL-Partners“ und ihrer Formatierung markiert. Wird der Bereich gefunden, so wird in Aktivität 4 nach den einzelnen Organisationen gesucht. Diese sind durch HTML-Listen-Elements (*li*) eingefasst. Wird schließlich eine Organisation gefunden, wird eine Instanz des Content-Typs *Organisation* erzeugt und die Felder *Organisation.officialName*, *.displayName*, *.title* und *.nationality* belegt.

Aktivität 5

In Aktivität 5 wird nach den in Klammern eingefassten Organisationseinheiten der Organisationen gesucht. Für die meisten Organisationen wurden entsprechende Abteilungen eingetragen. Wird eine Organisationseinheit gefunden, erzeugt dies eine Instanz des Content-Typ *OrganisationalUnit* und die Felder *OrganisationalUnit.officialName*, *.displayName*, *.title* werden befüllt. Zugleich wird die existierende hierarchische Beziehung zwischen der Organisation und ihrer Einheit im *OrganisationalUnit.organisation* referenziert. Auch hier durchläuft das Programm eine Schleife, bis alle Organisationen und Organisationseinheiten ausgelesen wurden.

Aktivitäten 6

Aktivität 6 befasst sich zunächst mit dem Auffinden der auf der Webseite enthaltenen Kontaktinformationen. Hierzu wird nach der Zeichenkette „Contact“ und ihrer speziellen Formatierung gesucht. Ist der entsprechende Abschnitt gefunden, wird dieser im Feld *Project.contactInformation* gespeichert. Ähnlich wie zuvor ist anschließend nach dem Abschnitt mit den Danksagungen für Beteiligte an diesem Projekt zu suchen. Wird der Abschnitt gefunden, wird der dort gehaltene HTML-Text in das Eigenschaftsfeld *Project.acknowledgements* eingefüllt. Der Endzustand der Transformationsregel für die Webseite *Background* ist erreicht.

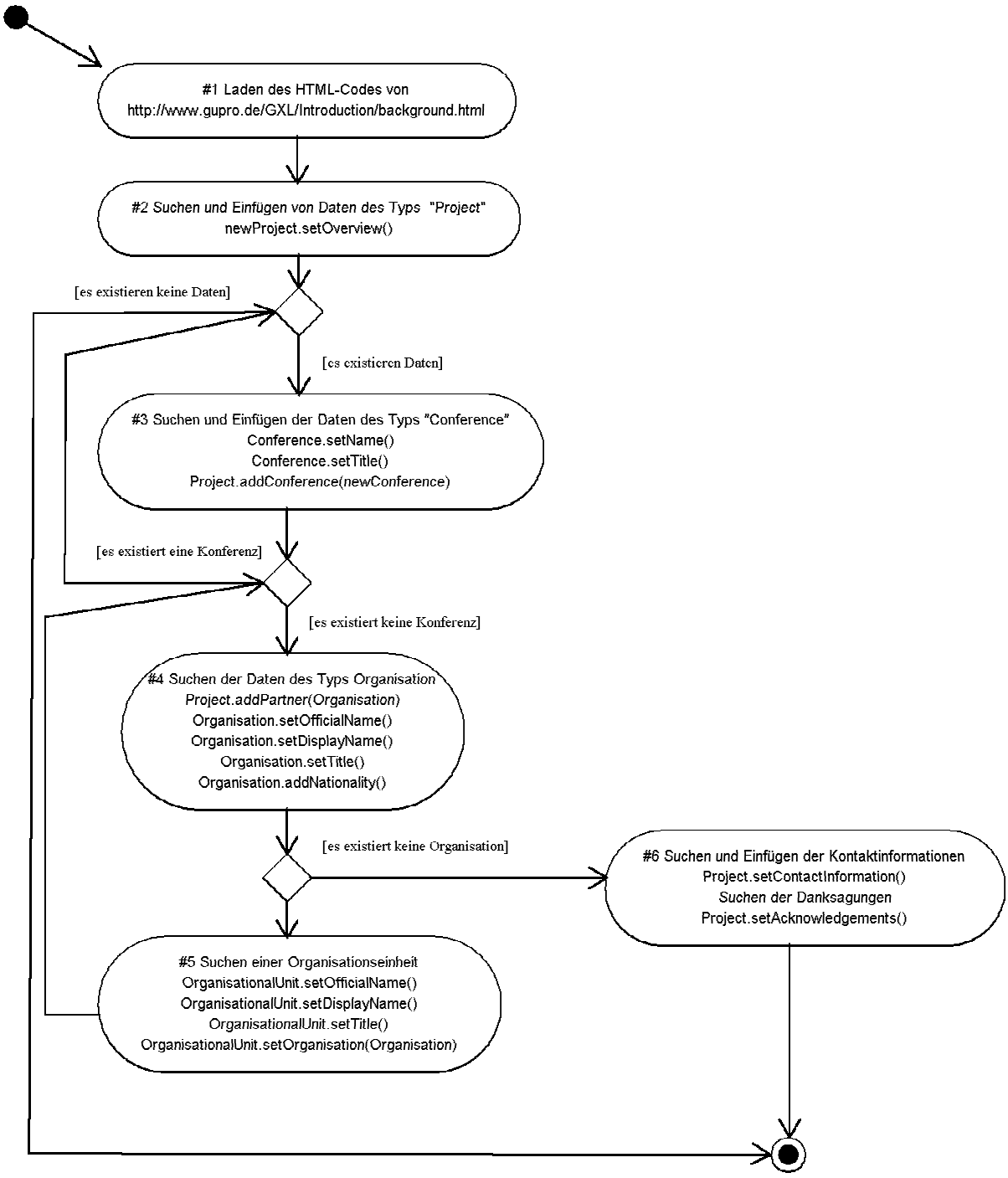

Abbildung 62: Aktivitätsdiagramm für Transformationsregeln für die Webseite *Background*

4.6.5 Transformation der Webseiten *Introduction*

Die unter dem Navigationspunkt *Introduction* aufzufindenden Webseiten enthalten einen Artikel, der eine einleitende Übersicht zu dem Thema GXL gibt. Dieser Artikel ist auf mehrere Webseiten verteilt, zwischen denen unter Nutzung von Hyperlinks navigiert werden kann. Im Rahmen der Verfeinerung des Ziel-Designs wurde beschlossen, diese Webseiten inhaltlich unverändert zu übernehmen. Um allerdings die Wartbarkeit der Webseiten mit automatischer Anpassung der Navigationslinks zu gewährleisten, müssen die Webseiten in einen Content-Typen transformiert werden, der dies leisten kann. Hierfür wurde *PloneArticle* ausgewählt.

PloneArticle besteht aus zwei Content-Typen, die für die Transformation wichtig sind: *PloneArticleMultiPage* und *PloneArticle*, wobei *PloneArticleMultiPage* als Container für *PloneArticle* fungiert und die Navigation zwischen den einzelnen Instanzen von *PloneArticle* erstellt. Außerdem enthalten die Webseiten Bilder, die zur vollständigen Migration der Website ebenfalls in das Zielsystem übertragen werden müssen. Diese werden als Objekte des Content-Typs *Image* gespeichert. Wie die Inhalte der Webseiten des Navigationspunkt *Introduction* auf die Content-Typen abgebildet werden, kann in Abbildung 63 nachvollzogen werden.

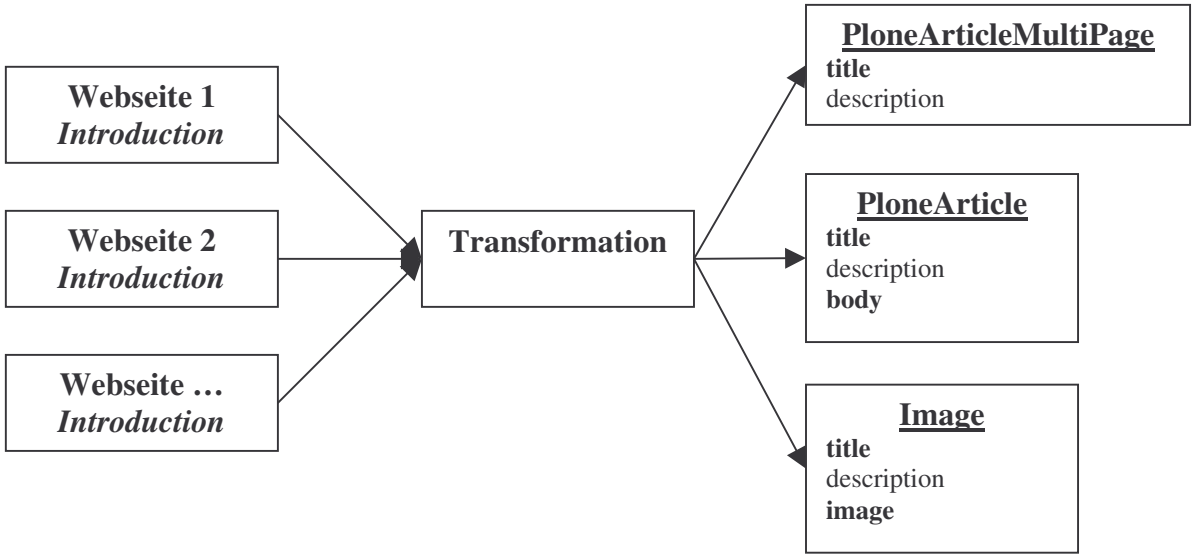

Abbildung 63: Belegung der Content-Typen durch die Webseiten *Introduction*

Ablauf der Transformation (siehe Abbildung 64)
Aktivität 1 und 2

Zunächst wird in Aktivität 1 der HTML-Quelltext der Start-Webseite dieses Navigationspunkts geladen. Danach erfolgt in Aktivität 2 für die Verwaltung der Inhalte der einzelnen Webseiten die Erzeugung einer Instanz des Content-Typs *PloneArticleMultiPage* und das Eigenschaftsfeld *PloneArticleMultiPage.title* wird befüllt.

Aktivität 3 und 4

In der aktuell geladenen Webseite befindet sich neben dem Abstract des Dokuments auch ein Inhaltsverzeichnis mit den Hyperlinks zu den weiteren Abschnitten dieses Artikels. Das Inhaltsverzeichnis wird in Aktivität 3 an Hand der Formatierung gesucht und nachfolgend für die Navigation zu den anderen Abschnitten des Artikels verwendet. Die einzelnen Webseiten der Abschnitte werden aufgerufen und deren Inhalte ausgelesen. Zur Speicherung der Daten wird innerhalb des Containers *PloneArticleMultiPage* eine Instanz des Content-Typs *PloneArticle* erzeugt und dessen Eigenschaftsfelder *PloneArticle.title* und *.body* mit den Inhalten befüllt. Da bei den Webseiten auch Bilder verwendet werden,

wird ebenso nach deren Vorkommen im Quelltext der Abschnitte gesucht. Wird ein Bild gefunden, wird innerhalb des Ordners zur Verwaltung der Bilder eine Instanz des Content-Typ *Image* erzeugt und seine Eigenschaften *Image.title* und *.image* gesetzt.

Es wird solange nach Abschnitten des Artikels gesucht, bis keine mehr gefunden werden. Damit ist die Transformation abgeschlossen und der Vorgang geht in den Endzustand über. Das Vorgehen bei der Transformation kann in Abbildung 64 nachvollzogen werden.

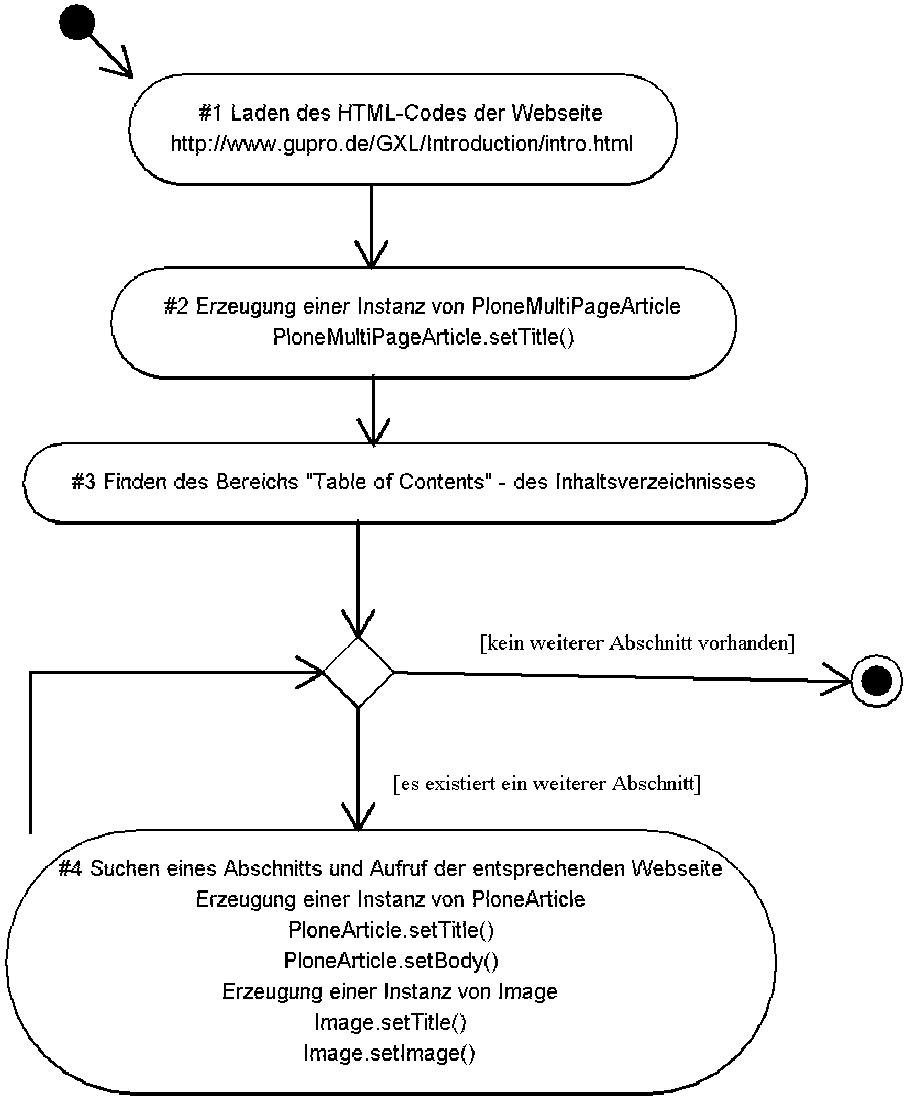

Abbildung 64: Aktivitätsdiagramm für Transformationsregeln für die Webseiten *Introduction*

4.6.6 Transformation der Webseite *FAQ*

Im Vergleich zu der eingangs beschriebenen Webseite „Background“, enthält die Seite *FAQ* nur Daten für wenige Content-Typen, nämlich *FAQSet* und *FAQ*. Allerdings können mit den für sie formulierten Transformationsregeln alle Instanzen der zu migrierenden Website in das Zielsystem überführt werden. Die Webseite *FAQ* wurde in der zu migrierenden Website aus einem XML-Dokument generiert. Da die Auswertung der Daten auf Grundlagen von XML einfacher durchzuführen ist, wird die Transformation mit diesem Dokument vorgenommen. Die genauen Abläufe dieses Vorgangs zeigt Abbildung 66, während Abbildung 65 die betroffenen Content-Typen und die befüllten Eigenschaften darstellt.

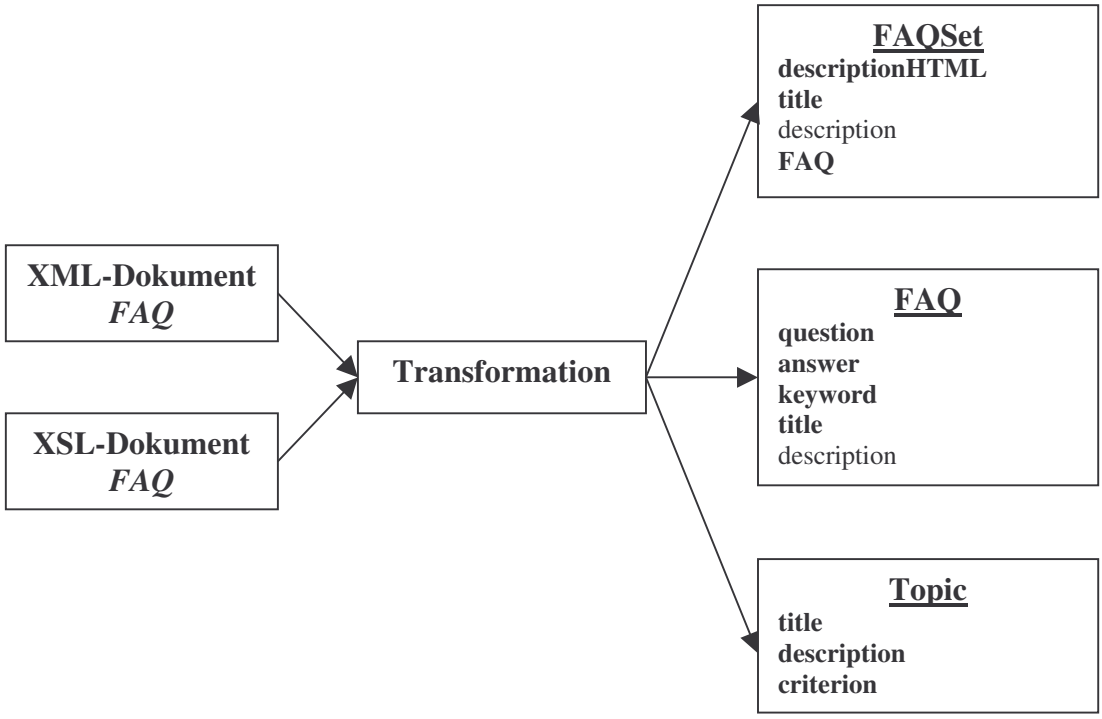

Abbildung 65: Belegung der Content-Typen durch die Webseite *FAQ*

Ablauf der Transformation (siehe Abbildung 66)
Aktivitäten 1 und 2

Zunächst wird in Aktivität 1 das XML-Dokument mit den FAQs geladen und geparst. In Aktivität 2 wird eine Instanz des Content-Typs *FAQSet* erzeugt und deren Felder *FAQSet.title* und *.descriptionHTML* befüllt. Die Daten für die Eigenschaft *FAQ_Set.descriptionHTML* liegen nicht im XML-Dokument vor und wurden zuvor aus der XSL-Datei kopiert.

Aktivität 3 bis 5

Innerhalb des XML-Dokuments sind die Elemente zunächst nach dem Typ *topic* geordnet. Um diese auslesen zu können, wird in Aktivität 3 eine Liste erstellt, die alle *topic*-Elemente enthält. Diese entsprechen den thematischen Überschriften auf der *FAQ*-Webseite und bestehen aus den einzelnen FAQs. Aus dem Wert des *name*-Attributs der *topic*-Elemente wird in Aktivität 4das Schlüsselwort *strKeyword* abgeleitet.

Schließlich wird für jedes *topic*-Element in Aktivität 5 eine Liste der in ihr enthaltenen Elemente des Typs *question* erzeugt. Diese *question*-Elemente enthalten die eigentlichen FAQs. Zur Speicherung der Daten wird für jedes *question*-Element eine Instanz des Typs *FAQ* innerhalb des Containers *FAQSet* erzeugt und durch Auslesen seines *text*-Attributs die Eigenschaftsfelder *FAQ.question* und *.title* befüllt und das Schlüsselwort *FAQ.keyword* zugewiesen. Innerhalb der *question*-Elemente befinden sich noch die verschachtelten Elemente *answer* und *link*. Sie enthalten die Daten für das Eigenschaftsfeld *FAQ.answer*.

Nachdem alle Eigenschaften eingetragen wurden, werden weitere *question*-Elemente ausgelesen. Sind innerhalb des aktuellen *topic*-Elements keine entsprechenden Elemente mehr vorhanden, wird das nächste *topic*-Element durchsucht. Sobald keine weiteren *topic*-Elemente mehr vorhanden sind, gilt die Eintragung der FAQs als beendet.

Aktivität 6

Um die FAQs in die Gruppen einzuteilen, wie sie durch die *topic*-Elemente vorgegeben wurden, werden in Aktivität 5 für jedes Schlüsselwort Instanzen des Content-Typs *Topic*

erzeugt. Dem logischen Ordner wird schließlich jeweils ein Schlüsselwort im Feld *Topic.criterion* zugewiesen.

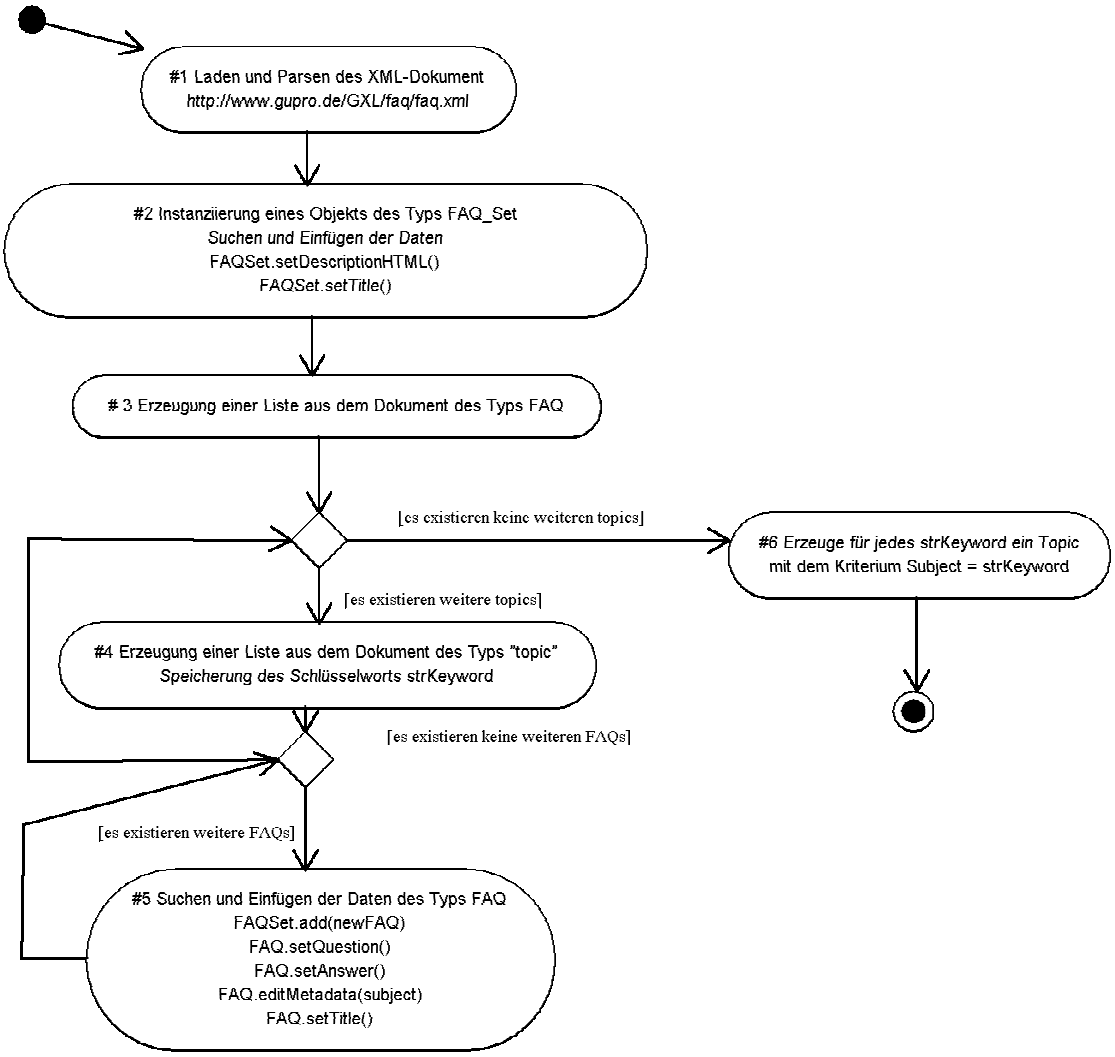

Abbildung 66: Aktivitätsdiagramm für Transformationsregeln für die Webseite *FAQ*

4.6.7 Transformation der Webseiten *Examples*

Die Webseiten des *Example*-Bereichs bilden eine Ausnahme im Vergleich zu fast allen anderen Webseiten. Inhalte zu den Content-Typen *Example* und *ExampleSet* sind über mehrere Webseiten verteilt und besitzen zudem eine eigene Navigation. Deshalb muss bei der Formulierung der Transformationsregeln ein besonderes Augenmaß auf die Navigation zwischen den einzelnen Seiten gelegt werden. Das Vorgehen ist in Abbildung 68 beschrieben, während Abbildung 67 die betroffenen Content-Typen und Eigenschaften zeigt.

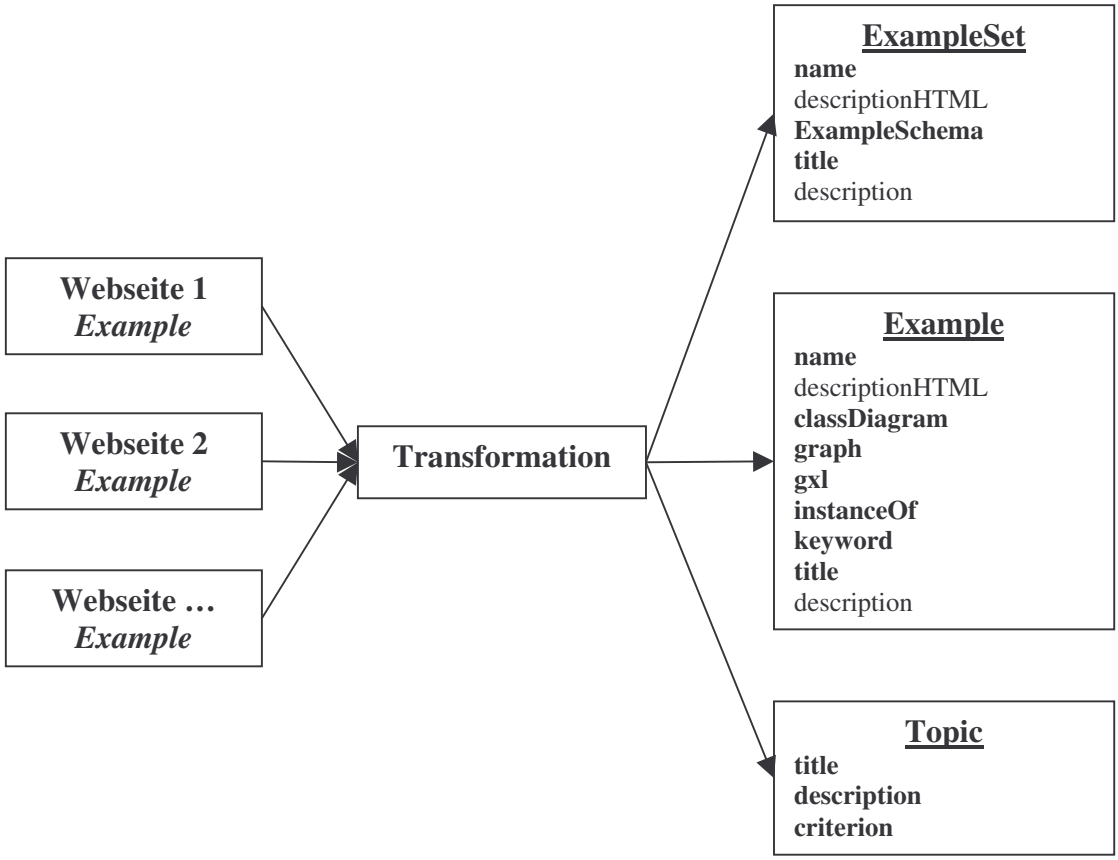

Abbildung 67: Belegung der Content-Typen durch die Webseiten *Examples*

Ablauf der Transformation (siehe Abbildung 68)
Aktivitäten 1 bis 3

Zunächst wird nicht die eigentliche *Examples*-Webseite mit ihren Frames, sondern die Seite mit der allgemeinen Navigation zur Untersuchung geladen (Aktivität 1). Auf Grundlage der dort angegebenen Links ist es möglich, zu den später angesprochenen Beispieluntergruppen zu navigieren. In Aktivität 2 wird eine Instanz des Content-Typs *ExampleSet* erzeugt und diese mit zweckmäßigen Inhalten für die Felder *ExampleSet.name* und *.title* gefüllt, die nicht von der Webseite stammen. Da jedes Schema, das auf den Beispielseiten erläutert wird, eine Instanz des Metaschemas ist, muss in Kenntnis dieser Tatsache in Aktivität 3 für dieses Metaschema eine Instanz des Content-Typs *Example* generiert werden. Damit können die später ausgelesenen Schemata immer auf diese Instanz referenzieren. Die Daten für dieses Metaschema werden über Navigation zu den die Daten enthaltenen Seiten basierend auf der Formatierung ausgelesen und in *Example.graph*, *.gxl* und *.classDiagram* eingefügt. Das Schlüsselwort *strKeywordLevelM2* markiert in *Example.keyword*, dass das Metaschema auf der Abstraktionsebene M2 angesiedelt ist. Da das erzeugte Metaschema eine Instanz von sich selbst ist, wird im Eigenschaftsfeld *Example.metaschema* auf das Metaschema verwiesen.

Aktivität 4

An Hand der Daten auf der allgemeinen Navigations-Webseite wird nach einer Beispieluntergruppe gesucht, die durch die Formatierung markiert ist. Nur solange in dieser Aktivität noch Beispieluntergruppen gefunden werden, wird der Transformationsvorgang fortgeführt. Wird eine Untergruppe gefunden, wird in aus der Überschrift ein Schlüsselwort generiert. Schließlich kann in der folgenden Aktivität im Bereich der Untergruppe nach Schemabeispielen gesucht werden.

Aktivität 5

Existiert in der Untergruppe ein Schemabeispiel, so wird in Aktivität 7 der HTML-Code der eigenen Navigationsseite des Schema-Beispiels geladen. In Aktivität 8 wird innerhalb des Containers *ExampleSet* eine Instanz des Content-Typs *Example* erzeugt und die Eigenschaftsfelder *Example.name* und *.title* belegt und dem Beispiel die Schlüsselwörter *strKeywordKindOfExample* für die Beispielgruppe und *strKeywordLevelM1* für die Abstraktionsebene in *Example.keyword* zugewiesen. Außerdem wird durch *Example.instance* die Referenz zwischen dem Schemabeispiel und dem Metaschema hergestellt.

Aktivität 6

Nun beginnt das abwechselnde Laden der Webseiten mit ihren Inhalten für die Beispiele. Zunächst wird der HTML-Code für das Graph-Beispiel geladen. Nachdem dort nach dem entsprechenden Bild gesucht wurde, wird dieses im Eigenschaftsfeld *Example.graph* abgelegt. Anschließend erfolgt das Laden der Webseite des Klassendiagramm-Beispiels. Auch hier wird das Bild gesucht und dann im Feld *Example.classDiagram* des Schemabeispiels abgelegt. Im Anschluss daran wird der HTML-Code des GXL-Beispiels geladen. Dort wird der gesuchte GXL-Beispielstext eingegrenzt, bevor er in *Example.gxl* eingelesen wird.

Aktivität 7

Nach den genannten Schritten sind die Daten für das Schemabeispiel vollständig. Nun wird in Aktivität 7 ein zugeordnetes Instanzbeispiel gesucht. Ist ein solches gefunden, wird innerhalb des Containers *ExampleSet* eine Instanz des Content-Typs *Example* erzeugt und die Eigenschaftsfelder *Example.name* und *.title* befüllt. Gleichzeitig wird im Feld *Example.instance* die Zugehörigkeit des Instanzbeispiels zu seinem Schemabeispiel festgelegt.

Aktivität 8

Analog zum Vorgehen beim Schemabeispiel wird schließlich der HTML-Code für das Graphbeispiel des Instanzbeispiels geladen und auf Vorhandensein des Bilds des Graphen untersucht. Das gefundene Bild wird in das Feld *Example.graph* eingetragen. Es fehlt noch der GXL-Text für das Instanzbeispiel. Hierfür wird die entsprechende Webseite geladen, auf den Text untersucht und in *Example.gxl* eingetragen. Abschließend erfolgt die Zuweisung der Schlüsselwörter *Example.keyword(strKeywordKindOfExample)* für die Beispielgruppe und *Example.keyword(strKeywordLevelM0)* für die Ebene M0 zum Instanzbeispiel.

Aktivität 9

Um die einzelnen Beispiele zu gruppieren, wird in Aktivität 9 für jedes Schlüsselwort für die Abstraktionsebene eine Instanz des Content-Typs *Topic* erzeugt und das Feld *Topic.criterion* mit dem Schlüsselwort belegt. Innerhalb dieser Instanzen werden weitere Instanzen von *Topic* generiert, die zusätzlich die Beispiele nach den Schlüsselwörtern der Beispielgruppen sortieren, wofür zusätzlich zum Schlüsselwort für die Ebene *Topic.criterion* noch das Schlüsselwort für die Beispielgruppen zugewiesen bekommt.

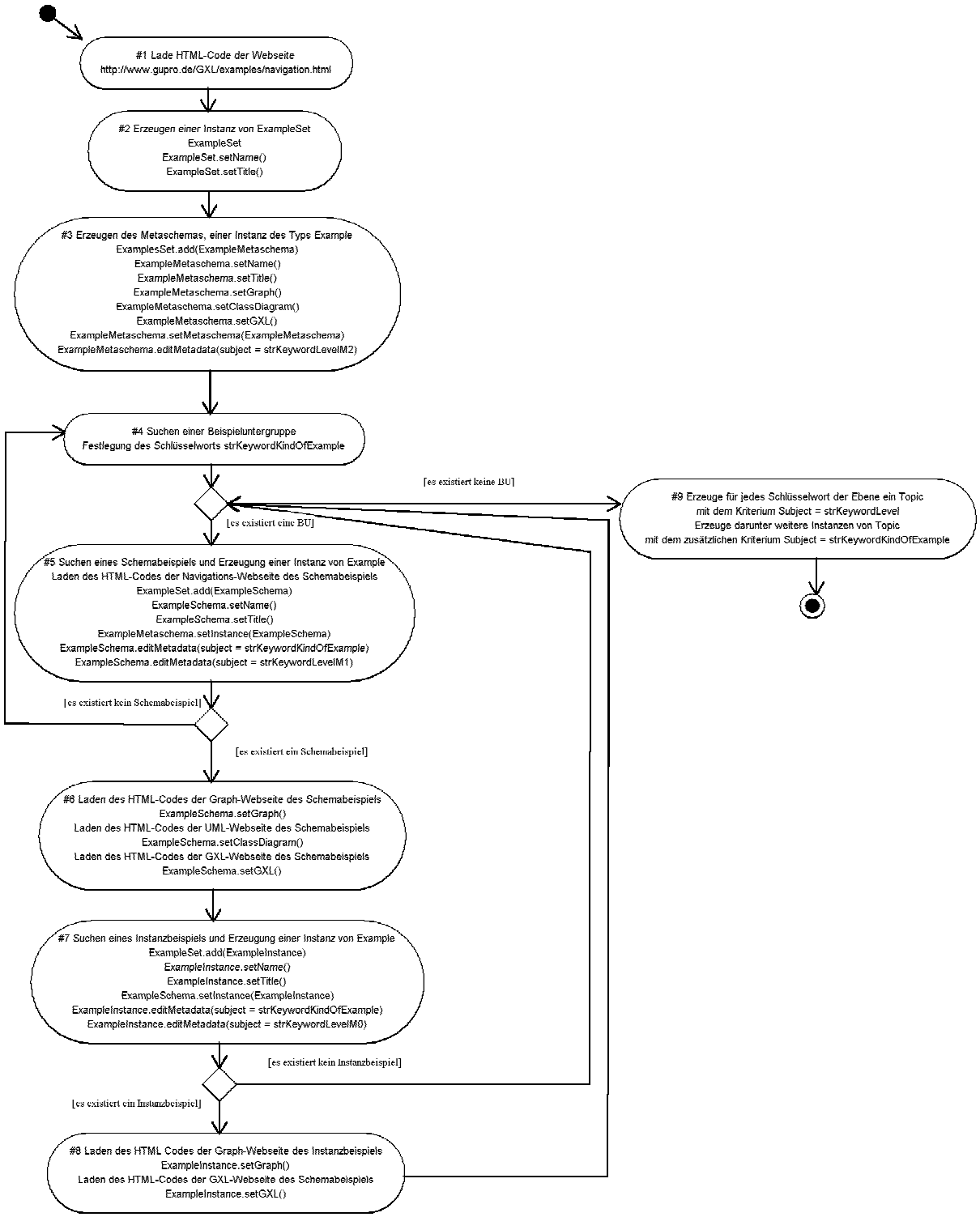

Abbildung 68: Aktivitätsdiagram für Transformationsregeln für die Webseiten *Examples*

4.6.8 Transformation der Webseite *Publications*

Die Transformationsregeln für die Webseite *Publications* unterscheiden sich wesentlich von den bisher beschriebenen. Der entscheidende Unterschied liegt darin, dass die Daten der Webseite nicht direkt in dem Zielsystem Plone gespeichert werden, sondern in der Literaturdatenbank des Fachbereichs 4 der Universität Koblenz-Landau *LitDB*. Die für die Formulierung der Transformationsregeln relevante Struktur der Literaturdatenbank ist in Abschnitt 4.4 beschrieben.

Von der Transformation der Webseite *Publications* sind die Datentypen *InProceedings* und *TechReport* betroffen. Alle Daten der betreffenden Webseite sind sowohl in HTML- als auch in XML-Code vorhanden. Die Abbildung der extrahierbaren Daten aus den Dokumenten ist in Abbildung 69 dargestellt. Wie die Daten von der Webseite über die Nutzung des XML-Dokuments in diese Datentypen übertragen werden, ist in Abbildung 70 nachzuvollziehen.

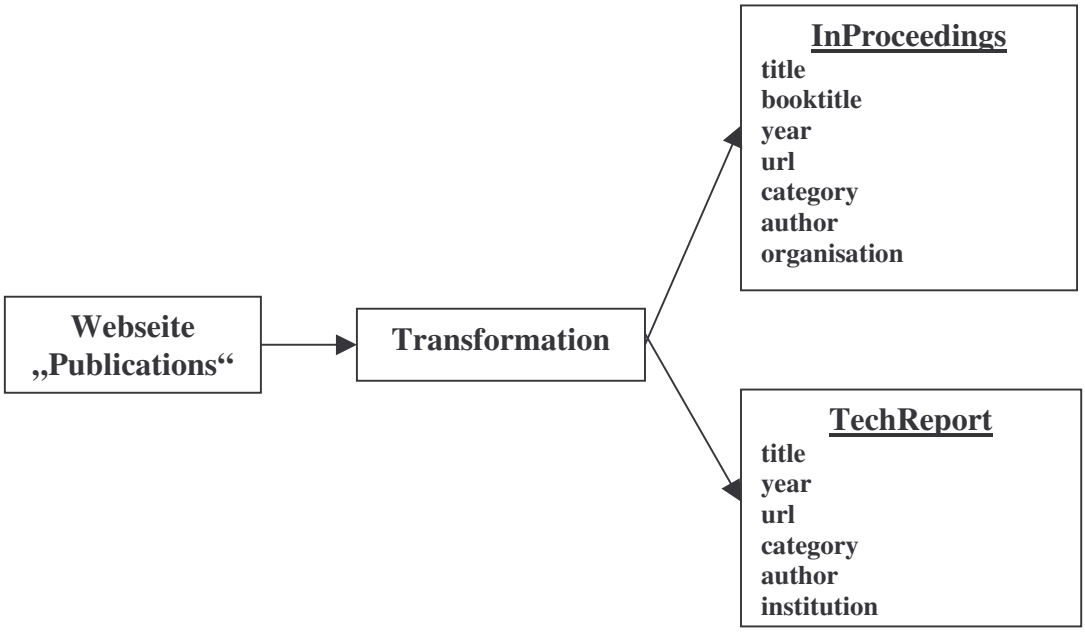

Abbildung 69: Belegung der Content-Typen durch die Webseite *Publications*

Ablauf der Transformation (siehe Abbildung 70)
Aktivitäten 1 bis 2

In Aktivität 1 wird das XML-Dokument *Publications* von der Website geladen und die darin enthaltenen Daten durch Parsen ausgelesen. Innerhalb des Dokuments ist das Element *Publications* die Wurzelinstanz. Wird diese gefunden kann mit der Transformation fortgefahren werden, existiert sie hingegen nicht, so wird der Vorgang abgebrochen.

Aktivität 3

Innerhalb des Wurzelelements des Dokuments befinden sich die Publikationseinträge in Form von *Pub*-Elementen. So lange in dem *Publications*-Element Elemente des Typs *Pub* enthalten sind, werden diese ausgelesen. Sobald kein Publikationseintrag in Form eines *Pub*-Elements gefunden wird, geht die Transformation in den Endzustand über. Schließlich muss unterschieden werden, ob es sich bei dem Publikationseintrag um ein Element des Typs *InProceedings* oder *TechReport* handelt. Dabei ist es nicht möglich, Kriterien für eine automatisierte Durchführung dieser Unterscheidung zu definieren. Die Auswahl des Typs erfolgt manuell.

Aktivität 4 bis 7

Wurde entschieden, dass der Publikationseintrag vom Typ *InProceedings* ist, so wird innerhalb von *LitDB* eine Instanz von *InProceedings* erzeugt (Aktivität 4). Hierbei muss verglichen werden, ob nicht bereits ein solcher Publikationseintrag in der Datenbank vorhanden ist. In Aktivität 5 werden die in dem *Author*-Element enthaltenen Daten zu *Author.surname*, *.firstName* und *.url* ausgelesen und in einer Instanz von *Author* gespeichert. Auch hier ist es wichtig, dass verglichen wird, ob der Autor bereits in der Datenbank erfasst ist. Analog dazu wird in Aktivität 6 mit den Daten für *OrganisationLitDB* verfahren, die aus verschiedenen Elementen innerhalb der *Pub-*

Elemente ausgelesen werden. Hierbei handelt es sich um die Felder *OrganisationLitDB.fullName*, *.shortName* und *.url*. In Aktivität 7 werden schließlich die Kerndaten für die Instanz von *InProceedings* eingetragen. Dazu gehören *InProceedings.title*, *.booktitle*, *.year* und *.url*, sowie die Referenzen zu den bereits eingetragenen Autoren und Organisationen. Zur Markierung des Literatureintrags wird das Feld *InProceedings.category* mit *„GXL_Pub“* belegt. Nach Eingabe aller Daten wird nach weiteren Publikationseinträgen gesucht.

Es wird darauf hingewiesen, dass es bei vielen Hyperlinks, die in *InProceedings.url* eingetragen werden, auf Grund des Datenmodells von *LitDB* unweigerlich zu Datenverlusten kommt. Hierbei geht es insbesondere um beschreibenden Text in der Umgebung des Hyperlinks, sowie um den Text, der für den Hyperlinks angezeigt wird (in HTML-Code bei *Text*). Innerhalb der Literaturdatenbank können nur einfache URLs ohne Kommentare eingegeben werden.

Aktivitäten 8 bis 11

Ganz ähnlich zum Vorgehen bei *InProceedings* werden Publikationseinträge des Typs *TechReport* verarbeitet. Kann ausgeschlossen werden, dass sich in der Datenbank nicht bereits der Literatureintrag befindet, wird in Aktivität 8 eine neue Instanz des Typs *TechReport* erzeugt. Anschließend werden in Aktivität 9 die Daten für *Author.surname*, *.firstName* und *.url* in eine Instanz von *Author* eingetragen. In Aktivität 10 werden die Daten *Institution.fullName*, *.shortName* und *.url* für die Institutionen festgehalten, die den technischen Bericht veröffentlicht haben. Zuletzt werden in Aktivität 11 alle Daten zu *TechReport* in die Literaturdatenbank überführt, dazu gehören die Felder *TechReport.title*, *.year*, *.url*, die Referenzen zu Autoren und Institutionen sowie die Zuweisung der Kategorie *TechReport.category = „GXL_Pub“*. Ebenso wie bei *InProceedings* gehen in dieser Transformation beschreibende Daten zu den Hyperlinks eines Publikationseintrags verloren.

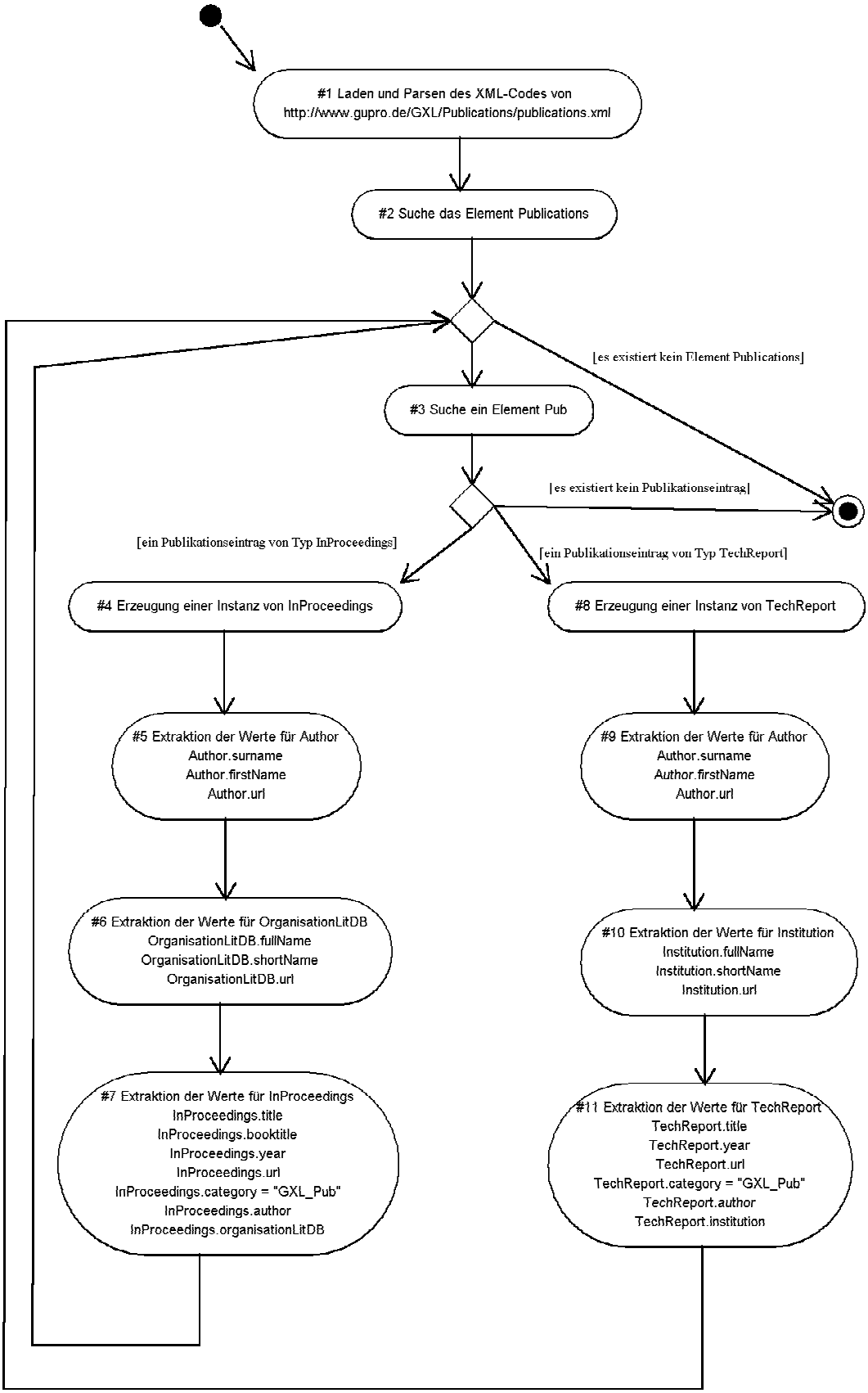

Abbildung 70: Aktivitätsdiagram für Transformationsregeln für die Webseite *Publications*

4.6.9 Transformation der Webseiten *DTD* und *XML Schema*

Die hier zusammengefassten Webseiten enthalten Quelltext in DTD und XML. Die Übertragung dieser Daten ist besonders wichtig, da es innerhalb der zu migrierenden Website eine Vielzahl von Hyperlinks gibt, die auf Bereiche dieser Webseiten verweisen. Eine frühzeitige Migration der dort gehaltenen Daten ist für die Überprüfung und Aktualisierung der Hyperlinks auf den anderen Webseiten notwendig. Zur Speicherung der Inhalte wird der für Quelltexte vorgesehene Content-Typ *Listing* verwendet. Welche Eigenschaftsfelder von Listing bei der Transformation der Webseiten betroffen sind, ist in Abbildung 71 nachzulesen.

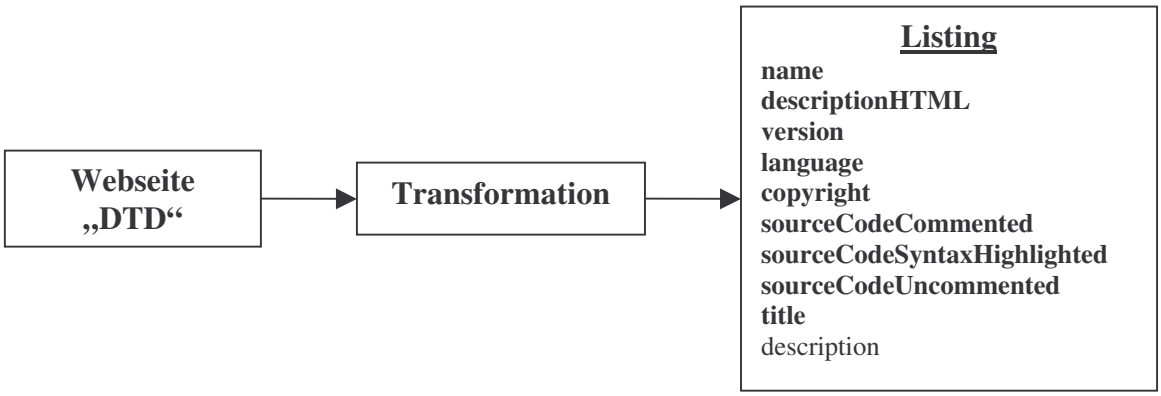

Abbildung 71: Belegung der Content-Typen durch die Webseite *DTD* und *XML Schema*

Ablauf der Transformation (siehe Abbildung 72)
Aktivitäten 1 bis 2

Zu Beginn wird der Quellcode der zu untersuchenden Webseite heruntergeladen und schließlich in Aktivität 2 nach der Überschrift durchsucht. Wird eine Überschrift gefunden, wird eine Instanz des Content-Typs *Listing* erzeugt und die Überschrift in den Eigenschaften *Listing.name* und *.title* eingetragen. Wenn an dieser Stelle keine Daten gefunden werden, dann geht der Vorgang in den Endzustand über.

Im weiteren Verlauf dieser Aktivität wird an Hand von charakteristischen Zeichenfolgen innerhalb des Quellcodes nach den Eigenschaften von Listing gesucht. Auf diese Weise können die Werte der Felder *Listing.version*, *.language* und *.copyright* extrahiert werden. Je nach Vorhandensein der einzelnen Quelltextvarianten wird aus den Webseiten der kommentierte Quelltext *sourceCodeCommented* und / oder der Syntax hervorgehobene Quelltext *sourceCodeSyntaxHighlighted* gepeichert. Zuletzt wird die Datei mit dem unkommentierten Quelltext in dem Feld *Listing.sourceCodeUncommented* heruntergeladen. Mit Abschluss dieser Aktivität geht die Transformation in den Endzustand über.

Die durchzuführenden Aktivitäten zur Transformation dieser Webseiten sind in Abbildung 72 zusammengefasst.

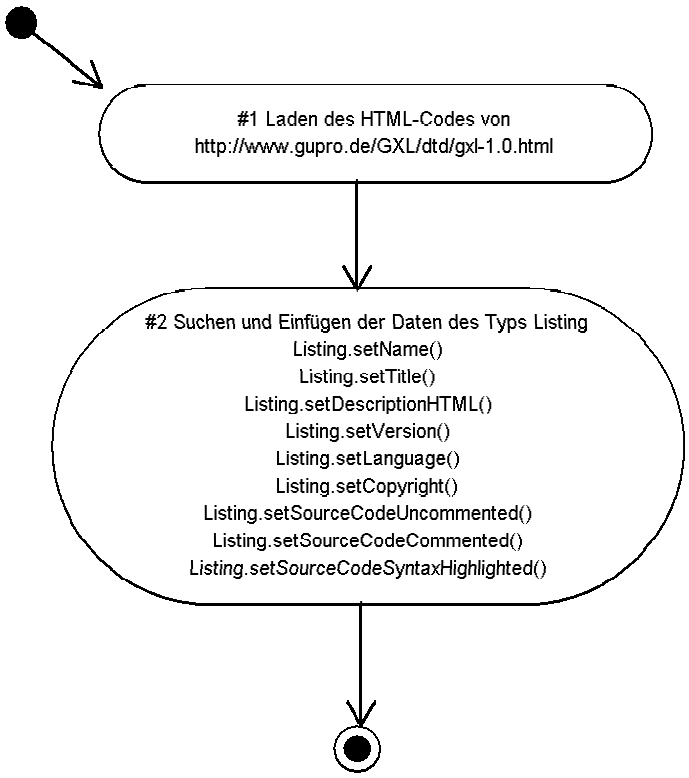

Abbildung 72: Aktivitätsdiagramm für Transformationsregeln für die Webseite *DTD* und *XML Schema*

4.6.10 Transformation der Webseiten *Graph Model* und *Metaschema*

Einen relativ einfachen Aufbau besitzen die Webseiten „Graph Model“ und „Metaschema“. Für die Speicherung der Inhalte dieser Webseiten wurde kein expliziter Content-Typ entwickelt. Stattdessen wird hier auf den generischen Content-Typ *Document* zurückgegriffen, der zur Verwaltung von einfachen Webseiten verwendet wird. Die Übertragung der Daten zu diesem Content-Typ ist in Abbildung 73 dargestellt.

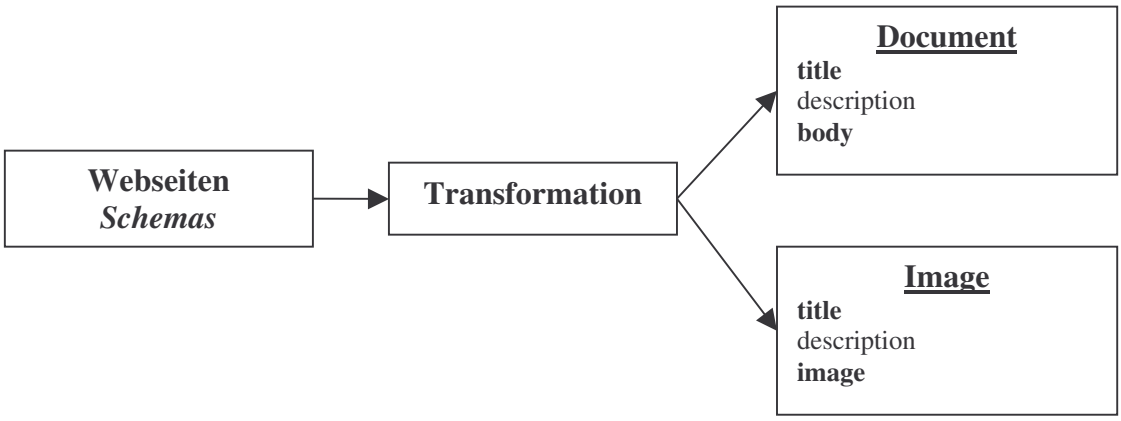

Abbildung 73: Belegung der Content-Typen durch die Webseite *Graph Model* und *Metaschema*

Ablauf der Transformation (siehe Abbildung 74)
Aktivität 1 und 2

Zunächst muss der Quellcode der Webseite geladen werden. Im Anschluss daran wird in Aktivität 2 die Überschrift auf der Webseite gesucht und, wenn vorhanden, in das Eigenschaftsfeld *Document.title* eingetragen. Der Quellcode, der sich innerhalb des *body*-Elements der Webseite befindet, wird in die entsprechende Eigenschaft *Document.body*

eingetragen. Da die Webseiten des Navigationsbereichs „Schemas“ über Bilder verfügen, ist es wichtig, dass diese Bilder in das Zielsystem übertragen werden. Diese Bilder werden im HTML-Quelltext gesucht und in den Ordner zur Bilderverwaltung der Webseite abgespeichert. Hierzu wird innerhalb des Bilderordners eine Instanz des Content-Typs *Image* erzeugt und die Eigenschaftsfelder *Image.title* und *.image* befüllt. Das Aktivitätsdiagramm in Abbildung 74 zeigt die hier beschriebenen Abläufe.

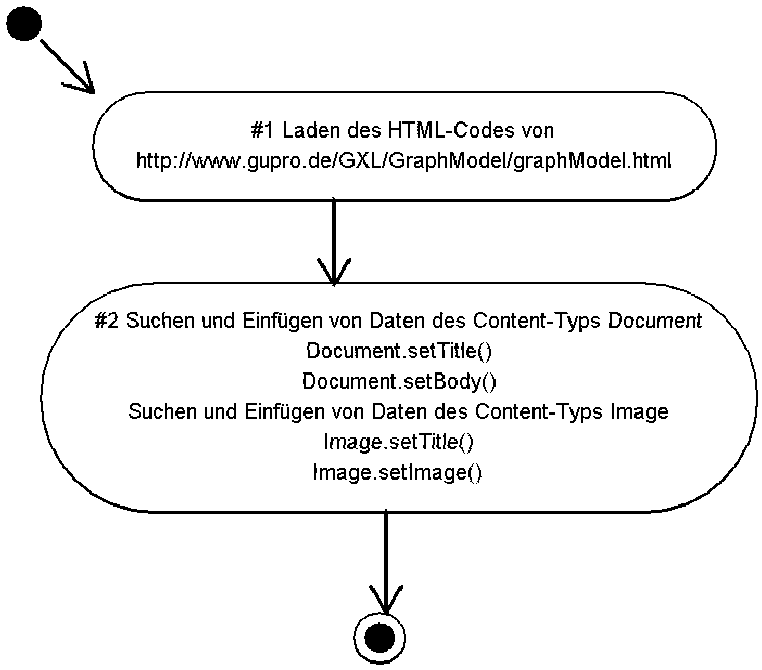

Abbildung 74: Aktivitätsdiagramm für Transformationsregeln für die Webseite *Graph Model* und *Metaschema*

4.6.11 Transformation der Webseite *Tool Catalogue*

Die Webseite *Tool Catalogue* enthält eine Vielzahl von Daten, die für das Zielsystem wichtig sind. Sie stellt Daten für die Befüllung von mehreren Content-Typen bereit. Dazu gehören:

- *ToolSet*
- *Tool*
- *Person*
- *Organisation*
- *OrganisationalUnit*

Sämtliche Daten für die Instanzen der Content-Typen *ToolSet* und *Tool* sind auf dieser Webseite enthalten. Für *Person*, *Organisation* und *OrganisationalUnit* hingegen ist diese Seite nur eine von mehreren, von denen sich Daten gewinnen lassen. Auch im Fall der Webseite *Tool Catalogue* sind die Daten in Form eines XML-Dokuments vorhanden. Zur einfacheren Verarbeitung der dort befindlichen Inhalte wird deshalb dieses Dokument für die Transformation als Datenquelle verwendet. Der Ablauf der Transformationsregeln kann in Abbildung 76 nachvollzogen werden, während Abbildung 75 eine datenorientierte Sicht der Transformation zeigt.

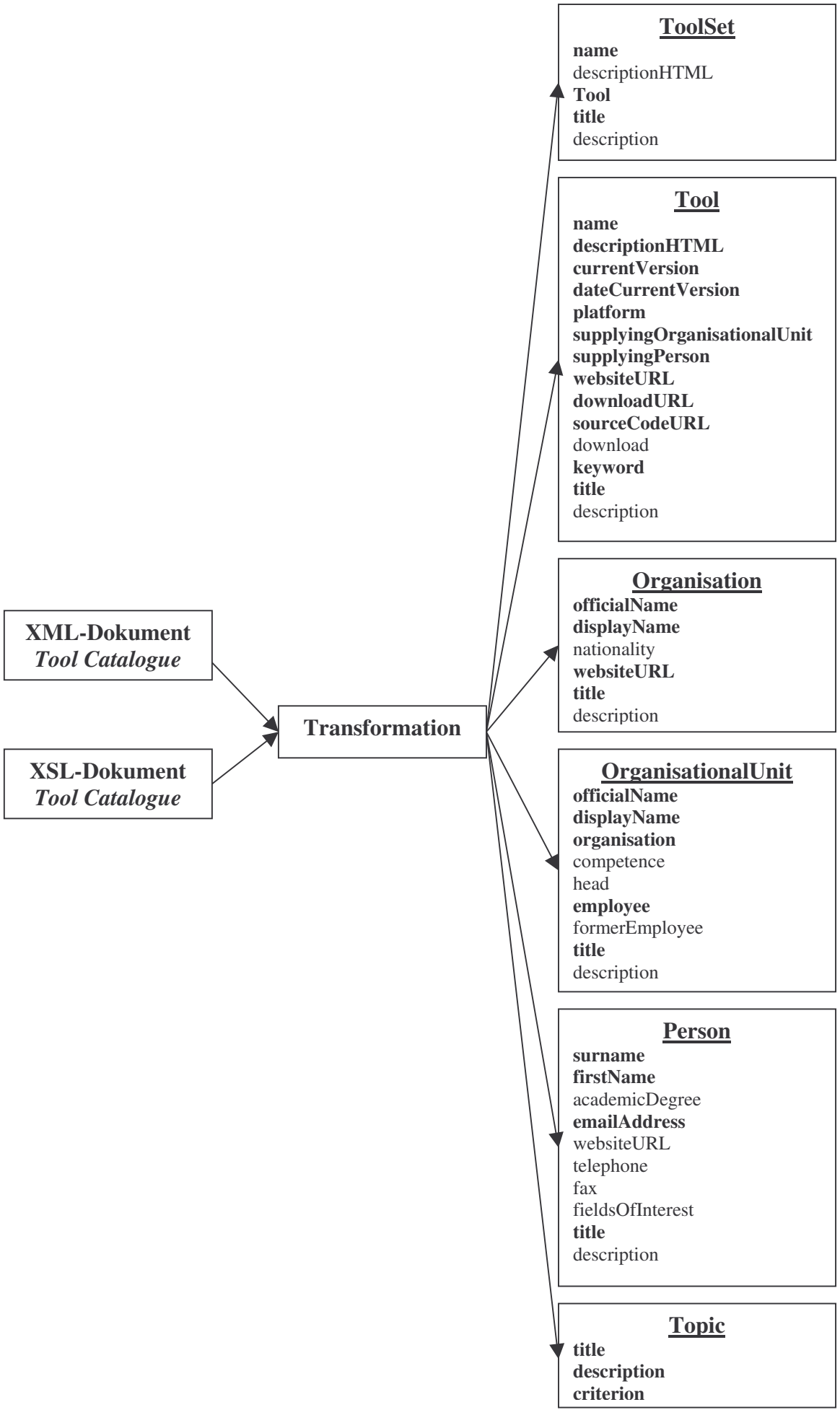

Abbildung 75: Belegung der Content-Typen durch die Webseite *Tool Catalogue*

Ablauf der Transformation (siehe Abbildung 76)
Aktivitäten 1 und 2

In Aktivität 1 wird das die Daten der Webseite *Tool Catalogue* enthaltende XML-Dokument heruntergeladen und geparst. In Aktivität 2 wird der Container *ToolSet* für die hinzuzufügenden Instanzen des Content-Typs *Tool* erzeugt und die Werte der Eigenschaftsfelder *ToolSet.name* und *.title* gesetzt.

Aktivität 3 und 4

Die in dem Dokument beschriebenen Werkzeuge sind als Elemente mit dem Namen *tool* eingetragen. Um auf alle Werkzeuge Zugriff zu erhalten, wird in Aktivität 3 eine Liste mit allen Knotenelementen des Typs *tool* erzeugt. Diese Liste wird zum Auslesen der Informationen in einer Schleife durchlaufen. In Aktivität 4 wird vom Container *ToolSet* für jedes *tool*-Element eine Instanz des Content-Typs *Tool* erzeugt. Die Werte der Attribute der *tool*-Elemente beinhalten die Eigenschaftsdaten für *Tool.title*, *.name*, *.websiteURL*, *.downloadURL*, *.sourceCodeURL*, *.currentVersion*, *.dateCurrentVersion*, *.platform* und *.descriptionHTML*. Die Zugehörigkeit zu einer bestimmten Klasse von Werkzeugen kann ebenfalls durch ein Attribut des *tool*-Elements abgeleitet werden und wird als *Tool.keyword* eingetragen.

Aktivität 5 und 6

Neben den Daten, die ausschließlich die Werkzeuge betreffen, enthält das XML-Dokument auch Informationen zu den Organisationen, deren Organisationseinheiten und Personen, die für das Werkzeug zuständig sind. Diese sind in dem Unterelement *supplier* eines jeden *tool*-Elements beinhaltet. Dieses Element vermischt somit Informationen zu den drei Content-Typen *Organisation*, *OrganisationalUnit* und *Person*. In Aktivität 5 werden die Attribute des *supplier*-Elements ausgelesen, die sich auf die Organisation beziehen. Sollten Informationen vorhanden sein, wird vom Container *OrganisationSet* eine Instanz des Content-Typs *Organisation* erzeugt und, soweit es möglich ist, die Eigenschaftsfelder *Organisation.officialName*, *.displayName*, *.title* und *.websiteURL* zugewiesen.

Teilweise enthalten die Attribute auch Informationen über die Organisationseinheiten der Organisationen. In Aktivität 6 wird nach den Daten gesucht und diese eingetragen. Hierfür erzeugt der Container *Organisation* eine Instanz des Content-Typs *OrganisationalUnit* und befüllt die Felder *OrganisationalUnit.officialName*, *.displayName* und *.title* mit den verfügbaren Daten. Falls keine Daten zu einer Organisationseinheit gefunden werden können, wird eine künstliche Organisationseinheit erstellt, damit ihr Personen zugeordnet werden können, die auch in dem *supplier*-Element aufgelistet sind. Hierfür werden in die Felder *OrganisationalUnit.officialName*, *.displayName* und *.title* im Unterschied zu vorhandenen Daten mit den Werten der gleichnamigen Felder ihrer Organisation gefüllt.

Aktivität 7

In jedem *supplier*-Element können sich auch Daten zu Personen befinden. In Aktivität 7 wird in den entsprechenden Attributen nach einer Person gesucht, die dem Werkzeug zugeordnet ist. Beim Finden einer Person, generiert der Container *PersonSet* eine Instanz des Content-Typs *Person* und weist ihm die Felder *Person.firstName*, *.surname*, *.title* und *.emailAddress* zu. In derselben Aktivität wird die in *Tool.supplyingPerson* die Verbindung zwischen Person und Werkzeug hergestellt. Wurden auch Daten zu einer Organisation und Organisationseinheit gesammelt, zu der die Person gehört, so wird diese in dem Eigenschaftsfeld *OrganisationalUnit.employee* eingetragen.

Aktivität 8

Zur Gruppierung der Werkzeuge werden für jedes Schlüsselwort der Instanzen von Tool Instanzen des Content-Typs *Topic* erzeugt. Deren Feld *Topic.criterion* wird jeweils ein Schlüsselwort zugewiesen.

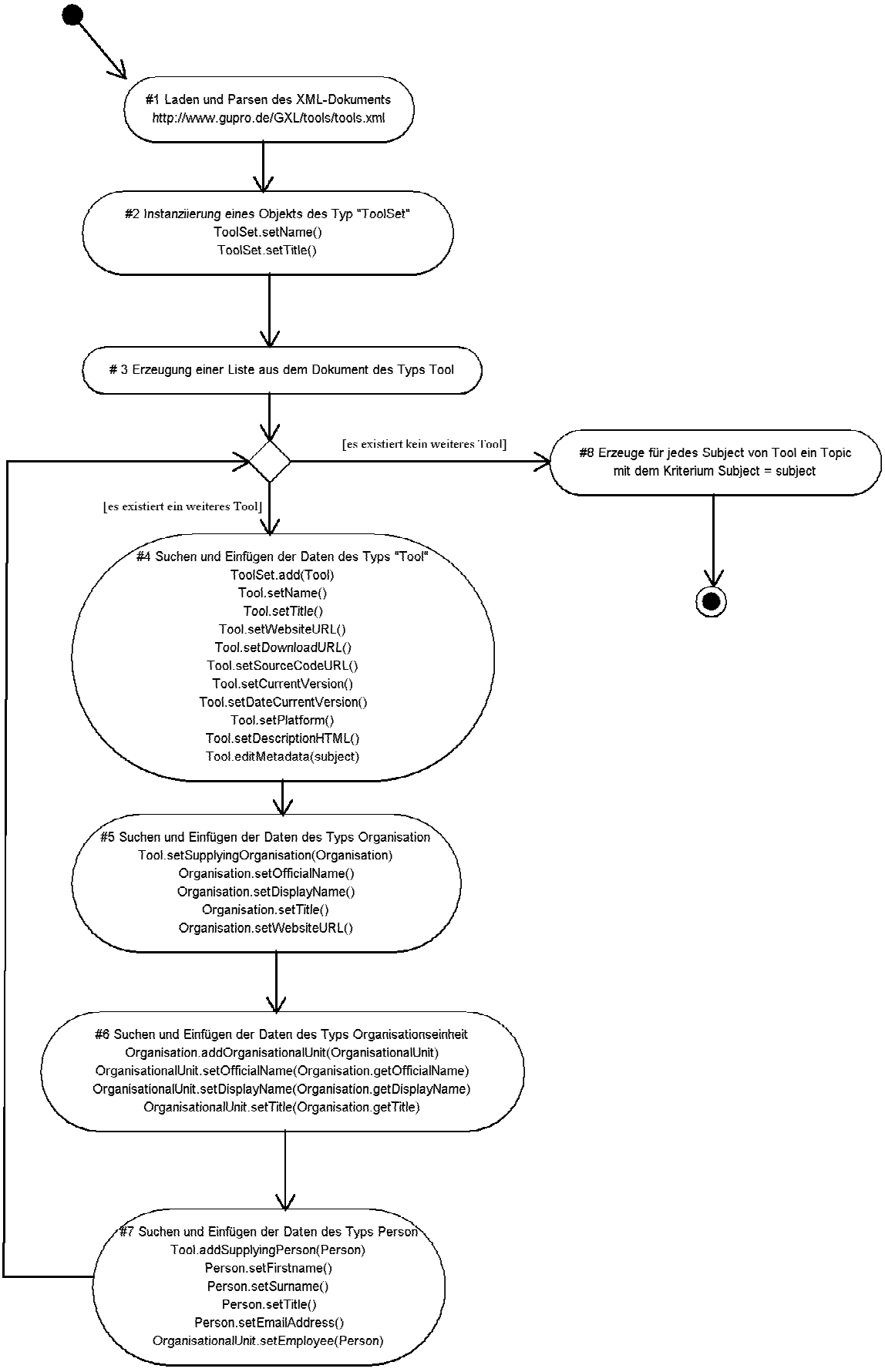

Abbildung 76: Aktivitätsdiagramm für Transformationsregeln für die Webseite *Tool Catalogue*

5 Strategieauswahl

Der Abschnitt „Strategieauswahl“ spricht mehrere Kernbereiche einer Migration an. Dabei handelt es sich um die Wahl der Transformations-, Umstellungs- und Übergabestrategie (vgl. Abschnitt 1.2.2 und [Ackermann (2005), S. 201]). Die Strategieauswahl wirkt sich ganz wesentlich auf alle Bereiche der Migration aus.

Transformationsstrategie
Die Transformationsstrategie beschreibt das geplante Vorgehen bei der Transformation, d.h. wie wird das Legacy-System oder Teile davon in das Zielsystem übertragen. Wichtige Begriffe sind dabei *Neuentwicklung*, *Konversion* und *Kapselung*. Grundlage der Auswahl der Transformationsstrategie bildet die globale Systembewertung der Legacy-Analyse (vgl. Abschnitt 3.2). Bei der globalen Systembewertung der zu migrierenden Website wurde neben der Eignung der Website für eine Migration auch festgestellt, dass die Qualität des Alt-Systems so hoch ist, dass sich neben der Neuentwicklung sogar eine Konversion oder Kapselung anbietet.

Zunächst zur Festlegung der Begrifflichkeiten. Sneed formuliert Neuentwicklung, Konversion und Kapselung als drei Alternativen der Migration [Sneed (1999), S. 20]. Als Unterscheidungskriterien zwischen den drei Varianten zieht er die Behandlung der Daten und der Programme heran.

Bei der Migration durch *Neuentwicklung* handelt es sich hinsichtlich der Daten um eine vollständige Neuzuordnung derselben. Die Veränderungen der Datenstruktur können beliebig umfangreich sein, solange gewährleistet ist, dass sich die Semantik der Daten nicht verändert [Sneed (1999), S. 20]. In der Dimension der Programme erfolgt allerdings eine vollständige Ablösung. So werden üblicherweise die Programme komplett neu geschrieben, der alte Programmcode bleibt unangetastet.

Im Rahmen einer *Konversion* wird hingegen versucht, einen möglichst großen Anteil des Legacy-Systems in das Zielsystem zu überführen. Dabei werden nicht nur die Daten transformiert, sondern auch die Programme [Sneed (1999), S. 20]. Ein besonderer Faktor bei der Konversionsstrategie ist die Wiederverwendbarkeitsrate des Alt-Systems. Als unteren Grenzwert gibt Ackermann eine entsprechende Rate von 75% an [Ackermann (2005), S. 55]. Systeme, die eine geringere Wiederverwendbarkeitsrate aufweisen, sollten nicht mittels einer Konversion transformiert werden.

Die letzte Alternative zur Durchführung der Transformation ist die *Kapselung*. Im Gegensatz zu den zuvor beschriebenen Transformationsstrategien verbleiben Daten und Programme im Legacy-System [Ackermann (2005), S. 51]. Die Integration des Alt-Systems in das Zielsystem erfolgt über die Programmierung eines Wrappers, der das Alt-System umhüllt und Schnittstellenfunktionalität bereitstellt [Sneed (1999), S. 22]. Dadurch ergibt es sich, dass das Zielsystem zwar oberflächlich eigenständig arbeitet (im Sinne eines Frontends), tatsächlich aber lediglich auf Dienste des Alt-Systems zurückgreift [Sneed (1999), S. 23].

Durch diesen minimalen Eingriff auf das Legacy-System erscheint die Kapselung als unkomplizierteste Migrationsstrategie. Allerdings birgt auch sie gewisse Risiken, da es notwendig ist, das Alt-System in genügendem Maße verstehen zu können [Ackermann (2005), S. 51]. Daraus ergibt sich, dass auch hier eine gewisse Wiederverwendbarkeitsrate eingehalten werden muss, damit dies gewährleistet ist. Ackermann gibt hierfür 50% an [Ackermann (2005), S. 55].

Umstellungsstrategie
Dem hingegen legt die Umstellungsstrategie fest, in welchen Schritten das Alt-System zu migrieren ist. Abhängig von dieser Entscheidung wird das gesamte Legacy-System entweder komplett zu einem Zeitpunkt („Big Bang“) oder in Einzelschritte unterteilt inkrementell migriert. Bei der inkrementellen Umstellung ist die Bildung von Paketen innerhalb des Alt-Systems und die Festlegung einer Migrationsreihenfolge von Bedeutung [Ackermann (2005), S. 201].

Übergabestrategie
Schließlich erfolgt durch die Übergabestrategie die Entscheidung bezüglich der Art der Übergabe des migrierten Zielsystems. Hierbei ist eine Festlegung hinsichtlich eines eventuellen parallelen Betriebs von Alt- und Zielsystem oder einer vollständigen Umstellung zu einem bestimmten Termin zu treffen [Ackermann (2005), S. 202].

5.1 Migrationsstrategien für Zielsystem-Kandidat erarbeiten

Bei der Erarbeitung der Migrationsstrategien erfolgt die Sammlung möglicher Migrationsstrategien für jeden einzelnen Zielsystem-Kandidaten, wie sie in der Aktivitätsgruppe „Zielsystem-Kandidaten definieren“ innerhalb des Ziel-Designs bestimmt wurden (siehe hierzu Abschnitt 4.1). In der vorliegenden Migration ist der Arbeitsaufwand hierfür reduziert, da Plone bereits als Zielsystem ausgewählt wurde. Somit müssen die Strategien lediglich auf Plone abgestimmt werden.

Die Migrationsstrategien umfassen sämtliche oben angesprochenen Bereiche, d.h. Transformations-, Umstellungs- und Übergabestrategie. Ebenso ist zu analysieren, welche Paketbildungsstrategien zur Durchführung der Migration eingesetzt werden.

Transformationsstrategie
Die globale Bewertung leistete bereits einen wichtigen Beitrag zur Ausarbeitung der Transformationsstrategie (siehe Abschnitt 3.2). Das Legacy-System wurde unter Berücksichtigung technischer und betriebswirtschaftlicher, bzw. wissenschaftlicher Aspekte bewertet und als migrationswürdig befunden. Bei der Portfolio-Analyse wurde seine Position innerhalb des Abschnitts für *Konversion, bzw. Kapselung* festgelegt. Dabei wies das Zielsystem allerdings keinen großen Abstand zur Alternative der Neuentwicklung auf.

Zur Erarbeitung von anwendbaren Strategien ist zunächst eine Definition notwendig, wie im Zielsystem Daten und Programme verstanden werden sollen. Die zu migrierende Website besteht im Wesentlichen aus einer Menge von Webseiten, XML- und deren XSL-Dokumenten, sowie *Makefile*-Anweisungen zur Verwaltung der Website.

Die Webseiten können als Daten aufgefasst werden, da sie keinerlei Funktionalität enthalten (vgl. Abschnitt 3). Im Zielsystem Plone sind die Webseiten-Inhalte in HTML-Blöcke untergliedert und so den Feldern der Content-Typen zugewiesen (vgl. Abschnitt 4.6). Dort werden sie als Daten behandelt und gliedern sich so in das Content Management System ein. Somit ist die Interpretation der Webseiten als Datenquellen gerechtfertigt.

Innerhalb und zwischen den Webseiten existieren Hyperlinks als Navigationselemente. Damit zeigen sie Beziehungen zu sich und anderen Webseiten auf. Da die Webseiten als Daten interpretiert werden, können deren Hyperlinks als gespeicherte Beziehungen zwischen den Daten definiert werden. Sie gliedern sich somit in das Verständnis von Webseiten als Datenquellen ein.

Noch unmissverständlicher ist der Zusammenhang bei den XML-Dokumenten. Sie beinhalten in strukturierter Form die Daten, die später durch Aufruf des Befehls *make* und die

Transformation durch die XSL-Anweisungen in Webseiten umgewandelt werden. Bei den XML-Dokumenten handelt es sich ganz eindeutig um Daten.

Die in den XSL-Dateien gehaltenen Formierungsanweisungen für die Transformation der XML-Dokumente nach HTML sind nicht so klar zu interpretieren. In der Umgebung der zu migrierenden Website werden sie als Bestandteil der Programmierung angesehen, da die in ihnen enthaltenen Anweisungen bei der Erzeugung der HTML-Seiten ausgeführt werden.

Die Verwendung von *make* und den *Makefiles* hingegen wird eindeutig der Programmierung zugerechnet. Die dort vorhandenen Anweisungen legen fest, wie die Webseiten der zu migrierenden Website generiert werden und sind damit Bestandteil eines einfachen Dokumentenverwaltungssystems.

In Hinblick auf das Zielsystem Plone werden die Daten vollständig migriert. Durch die Übertragung der Inhalte der Webseiten in das Konzept der Content-Typen in Plone, bzw. das Datenmodell der Literaturdatenbank des Fachbereichs 4 *LitDB*, wird dabei aber auch eine umfangreiche Restrukturierung vorgenommen. Damit deckt sich das Vorgehen bei den Daten mit den Migrationsstrategien der Neuentwicklung und der Konversion.

Anders verhält sich der Umgang mit den Programmen. Als funktionale Bestandteile des Alt-Systems wurden die Bereiche der XSL-Dokumente sowie die Verwendung von *make* und der *Makefiles* identifiziert. Hierbei dienen die angesprochenen Programme der Verwaltung, bzw. genauer gesagt der Generierung, der Webseiten im Alt-System.

In Plone ist die Funktionalität zur Verwaltung der Webseiten allerdings bereits implementiert. Insofern wird nicht auf die Implementierung mit *make* zurückgegriffen und es besteht keine Wiederverwendung. Eine Möglichkeit für die Wiederverwendung von Funktionalität in Plone besteht auf Seiten der XSL-Dateien. Da diese allerdings ganz wesentlich von der Struktur der XML-Dokumente abhängen und in Plone das Datenmodell verändert wurde, ergibt sich auch bei XSL keine Wiederverwendungsmöglichkeit. Die Aspekte der Programme finden sich also im Rahmen der Migration kaum wieder.

Damit legt der Zustand der zu migrierenden Daten die Wahl der Transformationsstrategie fest. Da, wie in Abschnitt 3.2.2 erarbeitet, die Datenqualität recht hoch (hierfür kann neben den Kriterien der Datenhaltung und der Aktualität der Daten auch die HTML / XML-Konformität herangezogen werden) ist und damit ein hohes Wiederverwendungspotenzial aufweist, wird die *Transformationsstrategie einer Konversion* ausgewählt. Genauer gesagt handelt es sich um eine Datenkonversion.

Umstellungsstrategie
Die Behandlung der Umstellungsstrategie dient der Festlegung, in welchen und wie vielen Schritten die Transformation durchzuführen ist. Die Einteilung der Schritte hängt von der Größe des Systems und der Beherrschbarkeit der Risiken ab, die mit der Migration einhergehen.

Bezüglich der Größe des Legacy-Systems weist die zu migrierende Website den Vorteil auf, dass sie nicht sehr groß und nicht komplex ist. Damit ist es möglich, das gesamte Alt-System als überschaubares Paket zu beschreiben und darauf die Umstellungsstrategie aufzubauen. Damit hängt zusammen, dass auch die Risiken einer fehlerhaften Transformation klein gehalten werden. Angesichts dieser Kriterien spricht vieles für eine Umstellung in nur einem Iterationsschritt.

Die Aufteilung des Systems in einzelne Pakete, die nacheinander mit zeitlichem Abstand migriert werden, stellt sich nicht als sinnvolle Alternative dar. Die Unterteilung in Migrationspakete und Planung der einzelnen Migrationsschritte verkompliziert die Durchführung der Migration nur unnötig.

Übergabestrategie
Die Übergabestrategie hängt eng mit der Umstellungsstrategie zusammen. In ihr muss festgelegt werden, in welcher Form das migrierte Zielsystem übergeben wird. Dabei kommt die Übergabe zu einem Zeitpunkt oder auch die Übergabe von Teilbereichen des migrierten Systems mit Parallelbetrieb in Frage. Angesichts der geringen Größe und Komplexität des Legacy-Systems ergibt sich auch hier, ähnlich zur Umstellungsstrategie, die Übergabe des vollständigen Zielsystems zu einem bestimmten Zeitpunkt an.

Definition potenzieller Paketbildungsstrategien
Die Zerlegung des Alt-Systems in isolierbare Pakete ist vor allem Bestandteil der inkrementellen Umstellungsstrategie. Obwohl diese für die vorliegende Migration keine Anwendung findet, ist die Bildung von Paketen für die Vorbereitung der Migration von Bedeutung.

Die Paketbildung macht vor allem bei der Einteilung der zu migrierenden Webseiten Sinn, da diese miteinander durch Hyperlinks verbunden sind. Bei der Migration verändern sich allerdings die Referenzen, da die Hyperlinks nicht mehr auf die alte Website, sondern auf das Zielsystem mit seinen neuen Adressen verweisen sollen (siehe Abschnitt 4.6). Die in dem Quelltext der Webseiten festgelegten Referenzen müssen also neu gesetzt werden. Damit dies zum größten Teil automatisiert geschehen kann, ist die Identifizierung der Verlinkung zwischen den Webseiten und die Definition einer Migrationsreihenfolge notwendig.

Durch die geschickte Festlegung einer Migrationsreihenfolge werden zuerst Webseiten migriert, die selbst auf keine oder wenige Seiten der Legacy-Website verlinken. Später können dann Webseiten migriert werden, die möglichst nur bereits migrierte Webseiten referenzieren, deren neue Adresse im Zielsystem bereits bekannt ist. Auf diese Weise minimiert sich der manuelle Aufwand bei der Aktualisierung der Hyperlinks. Für diesen Zweck wurde in Abschnitt 4.2 eine Einteilung der Webseiten des Zielsystems in Pakete vorgenommen.

5.2 Migrationsstrategien für Zielsystem-Kandidat bewerten

Der Aktivitätsbereich „Migrationsstrategien für Zielsystem-Kandidat bewerten“ ist durch die Reduzierung der Alternativen auf genau ein Zielsystem, nämlich Plone, wesentlich vereinfacht. Zwar wird an dieser Stelle eine Bewertung der für den Zielsystem-Kandidaten gewählten Strategie vorgenommen, eine Auswahl aus einer Menge von verschiedenen Migrationsstrategien entfällt allerdings.

Die technische und organisatorische Durchführbarkeit dieser Migrationsstrategie ist gewährleistet. Die begrenzten Mittel, die für die Migration zur Verfügung stehen (siehe Abschnitt 3.2) sind angesichts der überschaubaren Größe des Legacy-Systems ausreichend. Ebenso steht die Migrationsstrategie nicht in Widerspruch mit den in Abschnitt 2 definierten Migrationsanforderungen.

5.3 Migrationsstrategie auswählen

Die Erarbeitung und Bewertung der Migrationsstrategie erfolgte vor allem unter Berücksichtigung technischer Voraussetzungen. In der Aktivitätsgruppe „Migrationsstrategie auswählen“ kommen vor allem ökonomische Aspekte hinzu. Zur verbesserten Schätzung der Kosten und Risiken, die durch die Migration verursacht werden, wird auch eine Verfeinerung der Migrationsstrategie durchgeführt.

Die Kosten der Migration werden vor allem nach Zeitaufwand bemessen. Dadurch, dass für das Zielsystem keine Lizenzkosten entstehen und durch die nutzbare technische Infrastruktur

der Universität die Entwicklung fast kostenfrei erfolgen kann, liegt der einzige Faktor in der investierten Arbeitskraft für die Migration. Hierfür wurden im Verlauf der Anfertigung dieser Arbeit Zeitpläne erstellt und deren Einhaltung überprüft. Die Verfeinerungen der Migrationsstrategie können im Abschnitt 4 zum Ziel-Design nachvollzogen werden.

Im Zuge der Verfeinerung der Migrationsstrategie wurde auch ein vertretbares Paket für das Outsourcing entdeckt. Hierbei handelt es sich um den Bereich der Webseite *Publications*. Diese Daten sollten nach Absprache mit dem Website-Betreiber in *LitDB* eingetragen werden. Es existierte allerdings noch keine Schnittstelle, um die Daten aus der Literaturdatenbank in Plone auszulesen.

Die Existenz dieser Schnittstelle wurde zwar als notwendig für die Migration erkannt, sie stellt allerdings keinen originären Bestandteil dar. Dies liegt darin begründet, dass es sich bei der Anbindung der Datenbank um eine Erweiterung der Funktionalität der Website handelt. So wurde mit dem Website-Verantwortlichen eine Absprache getroffen, dass dieser die Entwicklung dieser Schnittstelle übernimmt und diese schließlich nur in das Zielsystem integriert wird.

Die Migrationsstrategie für die zu migrierende Website lässt sich kurz zusammenfassen. Es handelt sich um eine Migration zum Zweck einer *Konversion* mit *Umstellung und Übergabe* des vollständigen Zielsystems *zu einem bestimmten Termin („Big Bang"-Umstellung)*. Zur Erleichterung der Transformation werden die *Webseiten des Alt-Systems in Pakete* eingeteilt, die eine weitestgehend automatisierte Aktualisierung der Hyperlinks der Webseiten erlauben. Die *Transformation* soll *überwiegend automatisiert* durch die eigene Entwicklung von Transformationsprogrammen verlaufen. An Stellen, wo die Programmierung im Vergleich zur manuellen Ausführung einen zu großen Aufwand bedeutet, soll *auch manuell* transformiert werden.

6 Transformation

Der Kernbereich der Transformation beschäftigt sich mit der praktischen Überführung eines Migrationspakets des Legacy-Systems in das Zielsystem [Ackermann (2005), S. 214]. In diesem Bereich kommen die in bei den Transformationsregeln (vgl. Abschnitt 4.6) und bei der Migrationsstrategie (vgl. Abschnitt 5) getroffenen Entscheidungen zum Einsatz.

Ackermann beschreibt die Transformation als klassisches Gebiet für den Einsatz von Werkzeugen zur automatisierten Überführung des Alt-Systems. Durch die Spezialisierung der Transformationsaktivitäten ergeben sich häufig Möglichkeiten zum Outsourcing an externe Projektpartner [Ackermann (2005), S. 214]. Bei der vorliegenden Migration ist das systematische Outsourcen dieser Aktivitäten allerdings sowohl auf Grund der Überschaubarkeit des Projekts nicht notwendig, als auch im Sinne der Erstellung der Abschlussarbeit nicht gewollt.

Die Migrationsstrategie gibt vor, dass das Legacy-System im Ganzen zu migrieren ist, also eine „Big Bang"-Umstellung geplant ist. Daraus ergibt sich, dass der Kernbereich der Transformation im Gegensatz zum inkrementellen Ansatz nur einmal durchlaufen wird.

6.1 Migrationspaket isolieren

Die Aktivitätsgruppe „Migrationspaket isolieren" stellt sicher, dass das zur Transformation ausgewählte Paket in die Migrationsumgebung übernommen und damit vom Alt-System isoliert wird. Gleichzeitig dürfen an den davon betroffenen Teilen des Legacy-Systems keine oder nur geringfügige Anpassungen vorgenommen werden. Kommt es während der

Transformation zu Änderungen des Migrationspakets, muss, wie später in diesem Abschnitt beschrieben, eine Deltamigration vorgenommen werden [Ackermann (2005), S. 219].

Durch die Wahl des „Big Bang“-Vorgehens bei der Umstellungsstrategie erfolgt die Isolierung des gesamten Legacy-Systems zum Zeitpunkt der Transformation. Der Begriff der Isolierung ist in diesem Fall nicht ganz zutreffend. Tatsächlich wird die Transformation ausgehend von der operativen Website durchgeführt, wie sie auch zu jedem Zeitpunkt über das Internet abgerufen werden kann. Die Isolierung erfolgt durch Reduzierung der Änderungsarbeiten an der gesamten Website für den Zeitraum der Migration.

Dieses Vorgehen ist im vorliegenden Fall unproblematisch. Zum Zeitpunkt der Transformation kann die alte Website nach wie vor über das Internet erreicht werden. Der Normalbetrieb ist somit gewährleistet. Lediglich dürfen während der Migration keine inhaltlichen und strukturellen Änderungen an der Website vorgenommen werden. Dies wäre bei größeren und komplexeren Systemen problematisch. Da die Legacy-Website allerdings nur sehr selten bearbeitet wird und die Struktur sehr stabil ist, kann rigoros verfahren werden.

6.2 Migrationspaket transformieren

Innerhalb der Transformation des Migrationspakets wird die tatsächliche Übertragung des Pakets in das Zielsystem durchgeführt. Der Vorgang geschieht zum großen Teil automatisiert durch ein Skript bei der Installation des GXL_Website-Produkts (siehe Abschnitt 4.6). Hierbei handelt es sich ausschließlich um eine Datenmigration. Durch das Zielsystem Plone, das bereits eine vollständige Content Management-Funktionalität bereitstellt, ist die Übertragung die Migration der Funktionalität zur Verwaltung der Website nicht mehr notwendig.

Ackermann erläutert in Rückgriff auf [Sneed (1999), S. 189], dass die vollständig automatisierte Durchführung der Transformation in der Realität nicht umzusetzen ist. Es wird immer die Notwendigkeit bestehen, an bestimmten Stellen manuell nachzubessern. Bei der zu migrierenden Website handelt es sich hierbei um die Aktualisierung der Hyperlinks. Durch Paketierung ist es möglich, einen großen Teil der website-internen Hyperlinks während der automatischen Transformation zu aktualisieren. Bei Webseiten, die sich gegenseitig verlinken, ist die Paketierung allerdings wirkungslos. In diesen Fällen, die bekannt sind (siehe Abschnitt 3.4), müssen nach dem automatischen Durchlauf der Transformation die Hyperlinks korrigiert werden.

Die automatisierte Durchführung der Transformation nimmt regulär lediglich wenige Minuten in Anspruch. Nimmt man noch die manuellen Nachbesserungen ohne Testen der Transformation hinzu, ist die Transformation binnen einer Stunde abgeschlossen.

Zur Anwendung der Empfehlungen des ReMiP wurde die Mehrzahl der Transformationen zur automatisierten Ausführung programmiert. Nur im Fall der Übertragung der Publikationseinträge der zu migrierenden Website in die LitDB wurde auf Grund mangelnder mangelnden Kenntnis der technologischen Bedingungen und in Rücksicht auf den Zeitplan die Transformation manuell durchgeführt. Besonders effizient war die Transformation der Website-Bereiche, die in XML-Dokumenten gespeichert waren. Im Vergleich zur Transformation von Webseiten in Form von HTML-Dokumenten kann man zu deren Inhalte wesentlich leichter navigieren. Tabelle 5 verdeutlicht die Wahl der Mittel bei der Transformation.

Transformation	Umsetzung	Bemerkungen
Navigationsbereiche	Programmierung	
Webseite *Background*	Programmierung	
Webseiten *Introduction*	Programmierung	
Webseite *FAQ*	Programmierung	Erhöhte Automatisierung durch Auslesen eines XML-Dokuments
Webseiten *Examples*	Programmierung	
Webseite *Publications*	manuell	
Webseiten *DTD* und *XML Schema*	Programmierung	
Webseiten *Graph Model* und *Metaschema*	Programmierung	
Webseite Tools	Programmierung	Erhöhte Automatisierung durch Auslesen eines XML-Dokuments

Tabelle 5: Art der Umsetzung der Transformation

6.3 Deltamigration durchführen

Eine Deltamigration entspricht der Transformation eines Migrationspakets nach bereits erfolgter Transformation. Dies wird notwendig, wenn während der Transformation weiterhin Änderungen am Migrationspaket durchgeführt wurden, sei es nun durch Wartungsarbeiten oder Veränderungen des Datenbestands des Pakets. Während der Durchführung der Deltamigration werden allerdings sämtliche Aktivitäten am Migrationspaket eingestellt. Sie stellt die endgültige Transformation dar.

Bei der zu migrierenden Webseite wurde in der Aktivitätsgruppe „Migrationspaket isolieren“ festgelegt, dass während der Transformation keine Änderungen des Datenbestands oder der Struktur der Website mehr zugelassen sind. Damit entfällt die Notwendigkeit einer Deltamigration.

6.4 Komponenten implementieren

Die Aktivitätsgruppe „Komponenten implementieren“ beschreibt die Neuentwicklung von Software-Komponenten für das Zielsystem. Dies ist der Fall, wenn auf Grund einer zu geringen Qualität der alten Systembausteine keine Konversion oder Kapselung vorgenommen werden kann [Ackermann (2005), S. 221]. Die Implementierung von Komponenten ist damit ein wesentlicher Bestandteil der Neuentwicklung.

Bei der Neuimplementierung der Komponenten kommt die Entwicklung von Verwaltungsfunktionalität der Webseiten in Frage, wie sie zuvor mit *make* realisiert war. Die Implementierung dieser Funktionalität entfällt allerdings auf Grund der Datenkonversionsstrategie. Plone hat diese als Content Management System bereits integriert und man kann sich der standardmäßigen Funktionen ohne eigene Programmierung bedienen.

7 Test

Der Kernbereich „Test“ überprüft die einwandfreie Übertragung des Migrationspakets in das Zielsystem. Dabei ist die Überprüfung der funktionalen Äquivalenz des Zielsystems mit dem Legacy-System wichtig, so genannte Regressionstest, sowie die Einhaltung der Migrationsanforderungen [Ackermann (2005), S. 222].

Das Testen der Migration nimmt einen erheblichen Teil des Zeitaufwands für die Migration in Anspruch. Neben der Validierung der tatsächlich migrierten Komponenten gehört dazu auch die Überprüfung der Entwicklungsarbeit bei der Erstellung der Transformationen im Vorfeld. Im klassischen Fall wird versucht, den Testaufwand durch Ausnutzung von Automatisierung beherrschbar zu machen. Wie später in diesem Abschnitt zu sehen sein wird, ist im Fall der zu migrierenden Website das werkzeuggestützte Testen nur begrenzt möglich.

7.1 Globale Teststrategie definieren

Die Definition der globalen Teststrategie legt die geplanten Tests und deren Ziele sowie die Art der Testumgebung fest [Ackermann (2005), S. 226]. Damit werden den in den späteren Schritten konkretisierten Testfälle bereits grobe Vorgaben gemacht und Automatisierungspotenziale ausgemacht.

Zunächst wird geklärt, was getestet werden soll. Bei der Transformation der Legacy-Website sind vor allem Daten betroffen. Diese werden sowohl im Legacy- als auch im Zielsystem in Form von Webseiten angezeigt. Der Vergleich der beiden Systeme lässt sich sehr leicht ausgehend von den Webseiten durchführen. Die Testobjekte zur Überprüfung der Äquivalenz von Legacy- und Zielsystem sind die Webseiten.

Beim Testen kommt es nicht darauf an, dass jede Webseite „eins zu eins“ in das Zielsystem übertragen wurde. Durch die Wahl des Ziel-Designs, das sich auf eine Modellierung nach Content-Typen und nicht Webseiten aufbaut, ist dies auch gar nicht vollständig möglich.

Als Kriterien für die richtige Umsetzung der Migration werden folgende Punkte beachtet:

- Verständliche optische Darstellung der Webseiten-Elemente, d.h. von Text, Bildern, Tabellen, etc.
- Vorhandensein der für die Migration vorgesehenen Daten, d.h. von Text, Bildern, Dateien, etc.
- Korrekte Setzung der Verweise
- HTML-Validität
- Lauffähigkeit auf Browsern

Darstellung der Daten
Die Testgruppe zur Untersuchung der Darstellung der Daten beschäftigt sich mit der korrekten Umsetzung der Legacy-Website ins Zielsystem. Bei dieser augenscheinlichen Untersuchung wird jeder Webseite des Alt-Systems die entsprechende Website des Zielsystems gegenübergestellt und untersucht. Dabei ist es wichtig, dass *alle Elemente* angezeigt werden, also *optisch präsent* sind. Die Art der Speicherung, also z.B. ob ein angezeigtes Bild noch im Legacy-System oder im Zielsystem gespeichert ist, ist hierbei irrelevant. Falls durch die Migration die Inhalte auf mehrere Webseiten übertragen wurden, müssen diese durch entsprechende Navigationslinks erreichbar sein und werden in die Bewertung mit einbezogen.

Hinzu kommt die Überprüfung, dass die *Elemente verständlich und im Sinne der Anzeige im Alt-System übertragen* worden und angezeigt sind. Es wird nicht untersucht, ob die Darstellung identisch ist (siehe oben), sondern lediglich, ob aus der Darstellung im Zielsystem die Intention der Webseite hervorgeht. Erscheint die Webseite durch die Umstrukturierung der Migration wie aus dem Zusammenhang gerissen und zu unübersichtlich, so ist sie bei diesem Test durchgefallen.

Des Weiteren wird darauf geachtet, dass die *Formatierung der Elemente korrekt übertragen* wurde. Ein Beispiel ist, dass durch die Transformation eventuell die Struktur einer Tabelle verloren gegangen ist und diese falsch angezeigt wird.

Bei diesem Test gibt es keine Automatisierungspotenziale, da jede Webseite durch Betrachten auf Einhaltung dieser Testvorgaben überprüft wird. Als Browser zur Überprüfung der Darstellung der Daten wird der Internet Explorer verwendet. Der Test wird im Wesentlichen durch den Programmierer ausgeführt und in Stichproben von dem Website-Betreiber überprüft.

Vorhandensein der Daten
Bei der Überprüfung auf Vorhandensein der Daten wird gewährleistet, dass alle ausgewählten Elemente der zu migrierenden Website tatsächlich in das Zielsystem übertragen wurden. Die *Bilder* der zu migrierenden Website müssen sich physisch im Zielsystem befinden, analog dazu die *Dateien*. Auch hier werden Webseiten des Alt-Systems, die im Zielsystem eventuell auf mehreren Webseiten verteilt sind, als Gesamtes untersucht. Dabei müssen klar ersichtliche Navigationslinks zwischen den Webseiten vorhanden sein.

Im Gegensatz zu der oben genannten optischen Präsenz wird hier die *physische Präsenz der Daten im Zielsystem* verlangt. Zum Abschluss der Migration muss das Legacy-System entfernt werden können und das Zielsystem sämtliche Daten für die Anzeige der Website enthalten. Ebenso wie der Testvorgang bezüglich der Darstellung der Daten im Zielsystem, wird der Test auf Vorhandensein der Daten manuell durchgeführt. Auch übernimmt der Programmierer die Durchführung des Tests. Der Website-Betreiber kontrolliert in Einzelfällen die Einhaltung der Vorgaben.

Korrekte Setzung der Verweise
Der Test auf die korrekte Setzung der Verweise betrifft ausschließlich diejenigen Verweise, die auf Website-interne Webseiten referenzieren. Damit wird deren teil-automatisierte Aktualisierung im Verlauf der Transformation überprüft. Zu den Verweisen gehören nicht nur Hyperlinks, sondern auch Referenzen innerhalb des *src*-Attributs des *img*- oder *map*-Elements. Sobald das Zielsystem keine Verweise mehr auf die Legacy-Website enthält, ist der Testvorgang erfolgreich. Der Programmierer ist für die Durchführung des Tests verantwortlich.

HTML-Validität
Innerhalb des Abschnitts 3.2 zur Bestimmung der HTML-Validität des Quelltexts der Webseiten wurden Qualitätsmängel entdeckt. Die Verbesserung der Validität des Quellcodes wurde auch als Anforderung aufgenommen.

Das Zielsystem Plone verfügt durch das Produkt Kupu über eine Funktionalität zur Überprüfung und Aufbereitung von HTML-Code nach XHTML. Nach Anzeige einer Webseite in der Bearbeitungsansicht wird der HTML-Code automatisch überarbeitet (vgl. Abschnitt 3.4). Bei der Durchführung der vorhergehenden Tests bietet es sich an, diesen Schritt zusätzlich als Bestandteil der Quellcode-Sanierung durchzuführen.

Sobald eine Webseite über den Mechanismus von Kupu überarbeitet worden ist, kann sie über die Nutzung eines XHTML-Validators auf Validität überprüft werden. Der Test auf HTML-Validität wird durch den Programmierer durchgeführt.

Lauffähigkeit auf Browsern
Als eine explizite Anforderung an das Zielsystem wurde in Abschnitt 2.2 die Lauffähigkeit der Website auf den gängigen Browsern formuliert. Dies bedeutet, dass die einzelnen Webseiten möglichst fehlerfrei und auf die gleiche Weise angezeigt werden. Zur Überprüfung dieses Verhaltens werden die Webseiten in verschiedenen Browser aufgerufen und davon Screenshots gemacht. Ausgehend von diesen Bildern ist es dann möglich, die Art der Anzeige zu vergleichen. Dabei werden als Auswahl aller auf dem Markt befindlichen Browser die in

der Anforderungsanalyse definierten, meistgenutzten Programme verwendet, nämlich Internet Explorer, Firefox, Mozilla und Opera.

7.2 Test für Migrationspaket planen und vorbereiten

Innerhalb der Aktivitätsgruppe für die Planung und Vorbereitung der Tests für das Migrationspaket wird die Feinplanung für die einzelnen Testfälle vorgenommen [Ackermann (2005), S. 226]. Da es sich bei der Migration nur um ein einziges Migrationspaket handelt, das das komplette Legacy-System in das Zielsystem überführt, werden alle Testfälle für die Website definiert.

Testgruppe „Darstellung der Daten“
Wie weiter oben bereits dargestellt, umfasst die Testgruppe zur Darstellung der Daten die drei Tests auf optische Präsenz, verständliche Darstellung der Inhalte und korrekte Formatierung. Für jeden der drei Tests wird jede korrespondierende Webseite des Legacy-Systems und des Zielsystems aufgerufen und durch Betrachtung überprüft.

Zur Überprüfung der optischen Präsenz wird getestet:

- Ist der zu migrierende *Text* vollständig sichtbar? Wenn der Text nicht komplett angezeigt wird, ist der Test nicht bestanden. Der Test ist beendet, sobald der zu migrierende Text aus dem Alt-System mit dem im Zielsystem übereinstimmt.
- Werden die zu migrierenden *Bilder* vollständig angezeigt? Wenn ein Bild innerhalb einer Webseite im Zielsystem im Vergleich zum Alt-System nicht vorhanden ist, so ist der Test nicht bestanden. Der Test endet, wenn alle Bilder der Webseite des Alt-Systems in der Webseite des Zielsystems zu sehen sind.

Der Test auf verständliche Darstellung der Inhalte überprüft:

- Sind die *Inhalte* der migrierten Webseite so zu verstehen, wie sie im Alt-System vorlagen? Dabei wird lediglich untersucht, ob durch die Anordnung der Daten auf der Webseite für den Betrachter die gleiche Aussage erzielt wird, wie auf der Webseite des Alt-Systems. Wenn die migrierte Webseite dem Betrachter im Sinne der Intention der alten Webseite nicht verständlich ist, so ist der Test nicht bestanden. Er endet, sobald dem Betrachter ersichtlich ist, dass bei Webseiten die gleichen Inhalte enthalten.
- Ist die *Navigation* zwischen den Webseiten transparent? Es muss durch die lokale Navigation der Webseite möglich sein, zu den gleichen Webseiten im Zielsystem zu gelangen, wie auf der Webseite des Alt-Systems. Die globale Navigation wird dabei nicht auf Übereinstimmung überprüft. Wenn es nicht möglich ist, sich über die Navigation zu den gleichen Webseiten zu bewegen, ist der Test nicht bestanden. Er endet, wenn die Navigation des Zielsystems äquivalent mit der des Legacy-Systems ist.

Der Test auf korrekte Formatierung untersucht:

- Ist der auf der migrierten Webseite angezeigte *Text* im Vergleich zur Webseite des Alt-Systems richtig *angezeigt*? Dies umfasst die Untersuchung des Schriftgrads, Schriftschnitt (z.B. kursiv, fett) und die Farbe. Die Schriftart muss nicht mit der der Legacy-Webseite übereinstimmen. Der Test endet, wenn der Text der Zielwebsite unter den genannten Kriterien mit dem der Alt-Website übereinstimmt.
- Sind die *Elemente richtig positioniert*? Dabei wird die Positionierung der Elemente durch Mittel wie Tabellen, Absätze und Tabulatoren überprüft. Hierbei werden nur die Elemente des Alt-Systems miteinander verglichen, die sich im Zielsystem innerhalb eines Felds eines Content-Typs befinden (wie z.B. der Inhalt des Felds *sourceCodeCommented* in einem *Listing*-Content-Typ). Das liegt darin begründet, dass durch die automatisch generierte Ansicht von Archetypes die Positionierung zwischen den Elementen immer etwas von der Legacy-Webseite abweicht und darauf innerhalb

der Migration kein Einfluss genommen wird. Entspricht also die Positionierung der Elemente in der Ziel-Webseite nach den genannten Kriterien denen der Alt-Webseite, so endet der Test.

Testgruppe „Vorhandensein der Daten“
Die Überprüfung auf Vorhandensein der Daten der Legacy-Website innerhalb des Zielsystems zielt darauf ab, dass die Daten, unbeachtet der konkreten Darstellung, sich im Zielsystem befinden. Ist diese Bedingung gewährleistet, kann die Ziel-Website ohne Rückgriff auf Daten der Legacy-Website alleine bestehen.

Nachdem durch die Testgruppe „Darstellung der Daten“ bestätigt wurde, dass die Inhalte korrekt auf der Ziel-Website angezeigt werden, kann nun überprüft werden, ob die angezeigten Elemente tatsächlich auch im Zielsystem gespeichert sind. Für diesen Test werden die Webseiten aufgerufen und im Zielsystem überprüft, ob die angezeigten Objekte korrekt übertragen wurden. Diese Untersuchung umfasst:

- Wurden die *Bilder* korrekt übertragen? Hierbei wird getestet, ob die angezeigten Bilder sich im Zielsystem befinden. Dazu gehören lediglich die Bilder der Website, die zu den Inhalten gehören. Ausgenommen sind graphische Elemente wie horizontale Linien oder Navigationspfeile, die nur zur einheitlichen Gestaltung der Website verwendet wurden, aber selbst keinen Beitrag zu den Inhalten leisten. Der Test ist bestanden, wenn das angezeigte Bild im Zielsystem gespeichert ist.
- Wurden die zu migrierenden *Dateien* korrekt übertragen? Weniger offensichtlich als die Bilder, die häufig direkt auf der Webseite zu sehen sind, wird auch das Vorhandensein der Dateien überprüft, die zum Download zur Verfügung gestellt werden. Davon ausgenommen sind also gewöhnliche HTML-Dateien oder Dateien, die bei der Generierung von Webseiten mit dem Programm *make* benötigt werden. Der Test ist bestanden, wenn die auf den Webseiten referenzierte Datei sich physisch im Zielsystem befindet. Tabelle 6 listet die Dateien auf, die im Legacy-System zum Download zur Verfügung gestellt wurden und auch im Zielsystem vorhanden sein müssen.

#	Datei	Beschreibung
1	http://www.gupro.de/GXL/examples/gxl-1.0-examples.zip	Beispiel-Webseiten als ZIP-Ordner
2	http://www.gupro.de/GXL/examples/gxl-1.0-examples.tar.gz	Beispiel-Webseiten im TAR-Archivformat
3	http://www.gupro.de/GXL/xmlschema/gxl-1.0.xsd	XML-Schema von GXL 1.0
4	http://www.gupro.de/GXL/gxl-1.0.dtd	DTD von GXL 1.0

Tabelle 6: Download-Dateien der Legacy-Website

Testgruppe „Korrekte Setzung der Verweise“
Beim Test zur korrekten Setzung der Verweise erfolgt die Überprüfung der Referenzen, wie sie in Hyperlinks, aber auch in Bildern vorkommen. Ziel ist dabei die Gewährleistung, dass alle Referenzen, die innerhalb der Legacy-Website verwiesen haben, jetzt wie vorgesehen nur noch innerhalb der Ziel-Website verweisen. Dabei werden abgeklärt:

- Sind die *Referenzen* korrekt? Es wird bei jedem *href*-Attribut der HTML-Elemente der Ziel-Webseite der Verweis auf das Ziel überprüft. Es dürfen keine Referenzen auf die Legacy-Website mehr vorhanden sein. Diese Bedingung wird über die Nutzung eines Web Crawlers überprüft. Der Test endet, sobald keine Referenzen mehr auf die Legacy-Website existieren und die externen Referenzen wie auf der Legacy-Website korrekt verweisen.
- Sind die *Quellangaben für die Bilder* korrekt? Für die Anzeige von Bildern wird das *src*-Attribut innerhalb des *img*-Elements verwendet. Dadurch wird auf die Quelle des Bildes verwiesen. Das *src*-Attribut darf keinen Quellverweis auf die alte Website

beinhalten. Der Test endet, sobald die Quellverweise nicht mehr auf die Legacy-Website referenzieren.

Testgruppe „HTML-Validität“
Bei der Überprüfung der HTML-Validität wird ein Vergleich der Validitätsfehler des Quellcodes der Ziel-Webseiten durchgeführt. Für die Untersuchung der Validität wird, wie in Abschnitt 3.4, ein Validator eingesetzt. Da der Quelltext im Zielsystem in XHTML gespeichert wird, wird ein Test auf XHTML-Validität durchgeführt. Beim Test werden folgende Aspekte untersucht:

- Wie hoch ist die *Anzahl der Fehler*? Gemessen an der Ausgabe der Anzahl der Fehler durch den Validator wird verglichen, wie sich die Menge der Fehler im Vergleich zum Alt-System verändert hat. Da das Zielsystem automatisch die Überprüfung und Anpassung des Quellcodes der Webseiten nach XHTML durchführt, und darauf kein Einfluss genommen werden kann, dient dieser Test nur der Feststellung der Zielerreichung der Fehlerreduzierung. Die Ergebnisse werden tabellarisch zusammengefasst. Der Test endet mit dem Vergleich der Fehleranzahl.
- Welche *Art von Fehlern* werden gemeldet? Ähnlich wie bei dem Vorgehen zur Vorbereitung des Legacy-Systems ist nicht nur die Anzahl der Fehler für die Messung des Erfolgs maßgeblich. Es wird auch berücksichtigt, um welche Art von Fehlern es sich handelt. Daraus können Rückschlüsse auf ihre Schwere und die Art der Entstehung gemacht werden. Auch hier endet der Test mit der Durchführung des Vergleichs der Ergebnisse der Webseiten des Legacy-Systems.

Testgruppe „Lauffähigkeit auf Browsern“
Innerhalb der Testgruppe „Lauffähigkeit auf Browsern“ wird untersucht, ob die Webseiten vollständig auf den gängigen Browsern angezeigt werden. Hierfür werden zur Aufwandsreduzierung nicht alle Webseiten gestestet, sondern nur einige exemplarisch. Die Kriterien bei der Untersuchung sind:

- Korrekte Anzeige aller Elemente: der Webseiten-Inhalt muss vollständig und richtig formatiert im Browser erscheinen. Wenn diese Kriterien erfüllt sind, gilt der Test als bestanden.
- Gleichartige Anzeige der Webseiten: im Vergleich mit den verschiedenen Browsern wird überprüft, ob das Verhalten der getesteten Webseite überall gleich ist. Wird festgestellt, dass die Webseiten in gleicher Weise angezeigt werden, dann gilt der Test als bestanden.

7.3 Test für Migrationspaket durchführen und auswerten

Die Durchführung der Tests für das Migrationspaket stellt den Kern des Kernbereichs „Test“ dar [Ackermann (2005), S. 228]. Innerhalb dieser Aktivität werden die zuvor spezifizierten Testfälle ausgeführt und die Daten zur späteren Auswertung gesammelt. Ackermann unterscheidet weiterhin noch die Aktivitätsgruppe „Testergebnisse auswerten“, die sich mit dem Abgleich der Ergebnisse zwischen Legacy- und Zielsystem beschäftigt [Ackermann (2005), S. 229]. In dieser Arbeit wurden beide Aktivitätsgruppen gemeinsam beschrieben, um den Zusammenhang zwischen der Durchführung und der Auswertung zu bewahren.

Die Testgruppen „Darstellung der Daten“ und „Vorhandensein der Daten“ wurden manuell ausgeführt durch den augenscheinlichen Vergleich der Legacy- mit den Ziel-Webseiten. Bei der Ausführung der Tests wurde möglichst gründlich vorgegangen. Zur Darstellung der Webseiten wurde der Internet Explorer 6.0 verwendet.

Testgruppe „Darstellung der Daten“

Die Testgruppe „Darstellung der Daten“ wurde manuell ausgeführt durch den augenscheinlichen Vergleich der Legacy- mit den Ziel-Webseiten. Zur Darstellung der Webseiten wurde der Internet Explorer 6.0 verwendet. Die so gewonnenen Erkenntnisse müssen vor allem in Hinsicht auf die Darstellung der Bilder durch die Ergebnisse der Testgruppe „Lauffähigkeit auf Browsern“ relativiert werden.

Die Prüfung der Sichtbarkeit des migrierten Texts konnte erfolgreich abgeschlossen werden. Es muss eingeräumt werden, dass durch den nicht automatisierten Vergleich eine garantiert lückenlose Kontrolle nicht möglich gewesen ist. Dennoch wurde die Mehrzahl der Webseiten so getestet und eine ausreichende Testabdeckung erreicht.

Bei der Untersuchung der Bilder wurden alle Webseiten, die Bilder enthalten, aufgerufen und deren Anzeige überprüft. In diesem Arbeitsschritt wurde gleichzeitig das physische Vorhandensein der Bilder getestet. Für jedes Bild musste die Bedingung erfüllt sein, dass die im Quelltext angegebene Internet-Adresse (der Wert des Attributs *src* im *img*-Element) sich im Adressraum der Ziel-Website befindet. Zum Zeitpunkt der Überprüfung war dies die Internet-Adresse *http://wwww.uni-koblenz.de:14080/GXL/*.

Wie Tabelle 7 zeigt, war der Test sowohl auf die Sichtbarkeit als auch auf das Vorhandensein der Bilder erfolgreich. In allen Fällen waren die Bilder sichtbar und auch korrekt gespeichert.

#	Webseite	Anzahl Bilder	sichtbar	physisch vorhanden
1	http://wwww.uni-koblenz.de:14080/GXL/gxl/overview-gxl/project-gxl	1	1	1
2	http://wwww.uni-koblenz.de:14080/GXL/gxl/overview-gxl/introduction-to-gxl/section2.html	1	1	1
3	http://wwww.uni-koblenz.de:14080/GXL/gxl/overview-gxl/introduction-to-gxl/section3.html	4	4	4
4	http://wwww.uni-koblenz.de:14080/GXL/gxl/overview-gxl/introduction-to-gxl/section4.html	5	5	5
5	Examples-Webseiten (alle Dateien unter http://wwww.uni-koblenz.de:14080/GXL/gxl/examples/examples-gxl/)	32	32	32
6	http://wwww.uni-koblenz.de:14080/GXL/gxl/schema-gxl/gxl-graph-model/	1	1	1
7	http://wwww.uni-koblenz.de:14080/GXL/gxl/schema-gxl/gxl-metaschema	1	1	1

Tabelle 7: Testergebnisse für die Sichtbarkeit und das Vorhandensein der Bilder

Die Überprüfung der Nachvollziehbarkeit der Inhalte, der Navigation, der Formatierung des Texts und der korrekten Position der Elemente erfolgte durch die Einschätzung des Website-Programmierers. Zwar stimmen, wie durch die vorhergegangenen Testschritte bereits erwiesen, die Inhalte größtenteils überein, durch das geänderte Layout im Zielsystem hat sich teilweise die Struktur der Webseiten stark verändert. Das Ergebnis konnte nicht so wie bei der Darstellung der Bilder quantifiziert werden und bleibt somit schwer rekonstruierbar. Es soll dennoch festgehalten werden, dass eine subjektive Überprüfung der genannten Eigenschaften erfolgt ist und nach Meinung des Verfassers der Arbeit die Testziele erreicht wurden.

Testgruppe „Vorhandensein der Daten“

Innerhalb der Testgruppe „Vorhandensein der Daten“ wurde bereits durch Tabelle 7 dargestellt, dass die physische Übertragung der Bilddateien vom Legacy- in das Zielsystem erfolgreich durchgeführt wurde. Hierbei unbeachtet blieb der Aspekt der kopierten Dateien, die zum Download zur Verfügung gestellt werden. Während der manuellen Überprüfung auf deren Vorhandensein im Zielsystem wurden alle Dateien, wie sie in Tabelle 6 definiert sind, an der vorgesehenen Stelle gefunden. Damit ist diese Testgruppe erfolgreich abgeschlossen.

Testgruppe „Korrekte Setzung der Verweise“

Bereits innerhalb der Testgruppe „Darstellung der Daten“ wurden die Bilder und deren physische Speicherung untersucht. Als Nebenprodukt dieses Test ergab sich auch, dass die

src-Attribute der Bilder korrekt gesetzt waren und dieser Test der Quellenangaben erfolgreich abgeschlossen werden kann.

Die Überprüfung der Gültigkeit der Referenzen, genauer also der *href*-Attribute bestimmter Elemente des HTML-Quelltexts, erfolgt über die Nutzung des Web Crawlers WebLoupe. Dieser Web Crawler versucht alle Ziele der *href*-Attribute aufzurufen und gibt seine Resultate in einer Tabelle aus. Der Test kann dann als bestanden angesehen werden, wenn es keine Referenzen mehr gibt, die entweder auf ein nicht vorhandenes Ziel oder ungewollt auf die alte Website verweisen.

Der Test wurde nach mehreren Durchläufen erfolgreich abgeschlossen. Es existieren keine „toten Links" mehr auf Webseiten außerhalb der GXL-Website. Ebenso verweisen die Referenzen innerhalb der Website alle auf existierende Dokumente und nicht mehr auf die Legacy-Website. Die Ergebnisse dieses Tests können nachvollzogen werden, indem man das mitgelieferte Programm WebLoupe installiert und damit die lokale Version der migrierten GXL-Website analysiert.

Testgruppe „HTML-Validität"
Die Testgruppe „HTML-Validität" vergleicht die Fehler der Ziel-Website mit der Legacy-Website sowohl quantitativ als auch qualitativ. Die Ergebnisse des Tests der HTML-Validität wurden in Tabelle 8 den Ergebnissen aus der Legacy-Analyse gegenüber gestellt (vgl. auch Tabelle 2). Die Spalte „Fehler Legacy" listet die Fehler der alten Website auf, die Spalte „Fehler Ziel-Website" die entdeckten Validitätsfehler der migrierten Website, während die Spalte „Bemerkungen" an passender Stelle Erläuterungen zu der Art der Fehler oder der Testmethode gibt.

#	Webseite	Fehler alt	Fehler neu.	Bemerkung
1	http://www.gupro.de/GXL	1	2	Fehler: syntax of attribute value does not conform to declared value.
2	http://www.gupro.de/GXL/Introduction/background.html	40	2	siehe oben
3	http://www.gupro.de/GXL/Introduction/intro.html	60	18	document type does not allow element "ul" here; missing one of "object", "applet", "map", "iframe", "button", "ins", "del" start-tag.
4	http://www.gupro.de/GXL/Introduction/section1.html	40	3	
5	http://www.gupro.de/GXL/Introduction/section2.html	145	6	
6	http://www.gupro.de/GXL/Introduction/section3.html	100	6	
7	http://www.gupro.de/GXL/Introduction/section4.html	84	5	
8	http://www.gupro.de/GXL/Introduction/section5.html	45	2	
9	http://www.gupro.de/GXL/Introduction/section6.html	43	2	
10	http://www.gupro.de/GXL/Introduction/references.html	236	8	
11	http://www.gupro.de/GXL/dtd/dtd.html	35	3	
12	http://www.gupro.de/GXL/xmlschema/xmlschema.html	42	3	
13	http://www.gupro.de/GXL/GraphModel/graphModel.html	46	67	Fehler: required attribute "alt" not specified. value of attribute "shape" cannot be "RECT"; must be one of "rect", "circle", "poly", "default".
14	http://www.gupro.de/GXL/MetaSchema/metaSchema.html	87	185	siehe oben
17	Simple example Schema (es existieren mehrere Legacy-Webseiten und nur eine Webseite im Zielsystem)	39	2	Legacy-Fehler zusammengerechnet aus Navigation, GXL, Graph, Klassendiagramm: 2 + 11 + 13 + 13 = 39 Fehler.
18	Tools (es exisitert eine Legacy-Webseite und mehrere Webseiten im Zielsystem)	11	2	die Fehler der Ziel-Webseite werden als Durchschnitt aller Webseiten berechnet, da jede konstant immer die gleichen zwei Fehler hat

#	Webseite	Fehler alt	Fehler neu.	Bemerkung
19	FAQ (es existiert eine Legacy-Webseite und mehrere Webseiten im Zielsystem)	13	2	siehe Tools
20	http://www.gupro.de/GXL/Publications/publications.html	39	15	

Tabelle 8: Validitätsfehler der Ziel-Website im Vergleich zur Legacy-Website

Die Ergebnisse zeigen, dass in fast allen Fällen die Webseiten des Zielsystems deutlich weniger Fehler als die Legacy-Website aufweisen. Nur im Fall der Webseiten *Graph Model* und *Metaschema* erhöhte sich die Fehleranzahl. In diesen Fällen handelte es sich um systematische Fehler, da der in Plone integrierte HTML-Editor Kupu bestimmte Attribute standardmäßig in Großschrift umwandelt. Dieser Mangel ist zwar unerwünscht, dennoch beeinflusst er nicht die Anzeige und Funktion der Webseite, so dass er toleriert werden kann.

Die restlichen Arten der Fehler beeinflussen ebenfalls nicht die Anzeige der Seite. Es handelt sich entweder um wenige systematische syntaktische Fehler wie das Auslassen eines nicht optionalen Attributs oder um Fehler, die aus der nicht erlaubten Verwendung eines Elements resultieren, wie sie auch schon auf der Legacy-Website bestanden. Angesichts der deutlichen Reduzierung der Fehlerzahl und der Anzahl der systematischen Fehlerarten wird der Test erfolgreich abgeschlossen. Es wurde zwar keine vollständige XHTML-Validität erreicht, diese war gemäß der Testdefinition auch nicht verlangt.

Testgruppe „Lauffähigkeit auf Browsern“
Der Test auf Lauffähigkeit auf Browsern wurde überwiegend durch Nutzung einer Dienstleistung im Internet ausgeführt. Für alle getesteten Browser wurde auf das Angebot einer Website zurückgegriffen, die kostenlos durch Eingabe der Internet-Adressen Screenshots der Webseiten anfertigt und sie zum Download zur Verfügung stellt.[29]

Die exemplarisch getesteten Webseiten decken folgende Bereiche ab:

- Webseite *GXL - Graph eXchange Language* (entspricht der Legacy-Webseite *Background*): Sie ist eine besonders wichtige Webseite, da sie den einführenden Überblick zum Thema GXL gibt.
- Webseite *Introduction*, „Abschnitt 2“: Für die Untersuchung der Anzeige des einleitenden Artikels zu GXL wurde exemplarisch nur die Webseite des zweiten Abschnitts untersucht.
- Webseite *Examples*, „Complex example Schema“: Repräsentativ für die Anzeige der Examples-Webseiten wurde die Webseite „Complex example Schema“ untersucht. Im Vergleich zu einer Webseite, die nur ein Instanz-Beispiel zeigt, verfügt sie über mehr angezeigte Elemente (ein Bild des Klassendiagramms).
- Webseite *Tools*, „FAMIX“: Die Webseite „Famix“ dient exemplarisch zur Überprüfung der Anzeige für alle Werkzeuge in der Werkzeug-Liste.
- Webseite *Definition*, „GXL DTD“: An Hand der Untersuchung der Webseite „GXL DTD“ wurde die korrekte Anzeige aller Instanzen vom Content-Typ *Listing* überprüft.

Das Ergebnis der Testgruppe „Lauffähigkeit auf Browsern“ war nicht zufrieden stellend. Bei der Überprüfung der Anzeige der Webseiten wurde durchgängig festgestellt, dass es Probleme mit der Anzeige von Bildern gab. Ohne nachvollziehbaren Grund wurden die Bilder manchmal angezeigt und das andere Mal nicht. Dieses Problem trat nur in den Fällen auf, bei denen die Bilder als eigenes Feld modelliert wurden, deren Ansicht durch das Standard-Template von Archetypes generiert wurde. Bilder, die in HTML-Feldern (*richtext*) eingebunden wurden, konnten stets angezeigt werden.

[29] Das Angebot ist unter http://browsershots.org/ abzurufen.

Damit ist die Ziel-Website hinsichtlich dieses Kriteriums durch den Test gefallen. Besonders stabil bei der Darstellung der Bilder haben sich Firefox, Opera und Mozilla erwiesen. Probleme gab es mit beiden aktuellen Versionen des Internet Explorers. Abbildung 77 zeigt die fehlerhafte Darstellung der Webseite *Background* auf dem Internet Explorer und die korrekte Darstellung auf Opera. Zur Überprüfung, ob es sich um ein spezifisches Problem des Internet Explorers handelt, wurde zusätzlich noch der Browser Safari 2.0 für das Betriebssystem Mac OS X. Aber auch bei Safari stellten sich die gleichen Probleme ein, so dass die Darstellungsproblematik verallgemeinert werden kann.

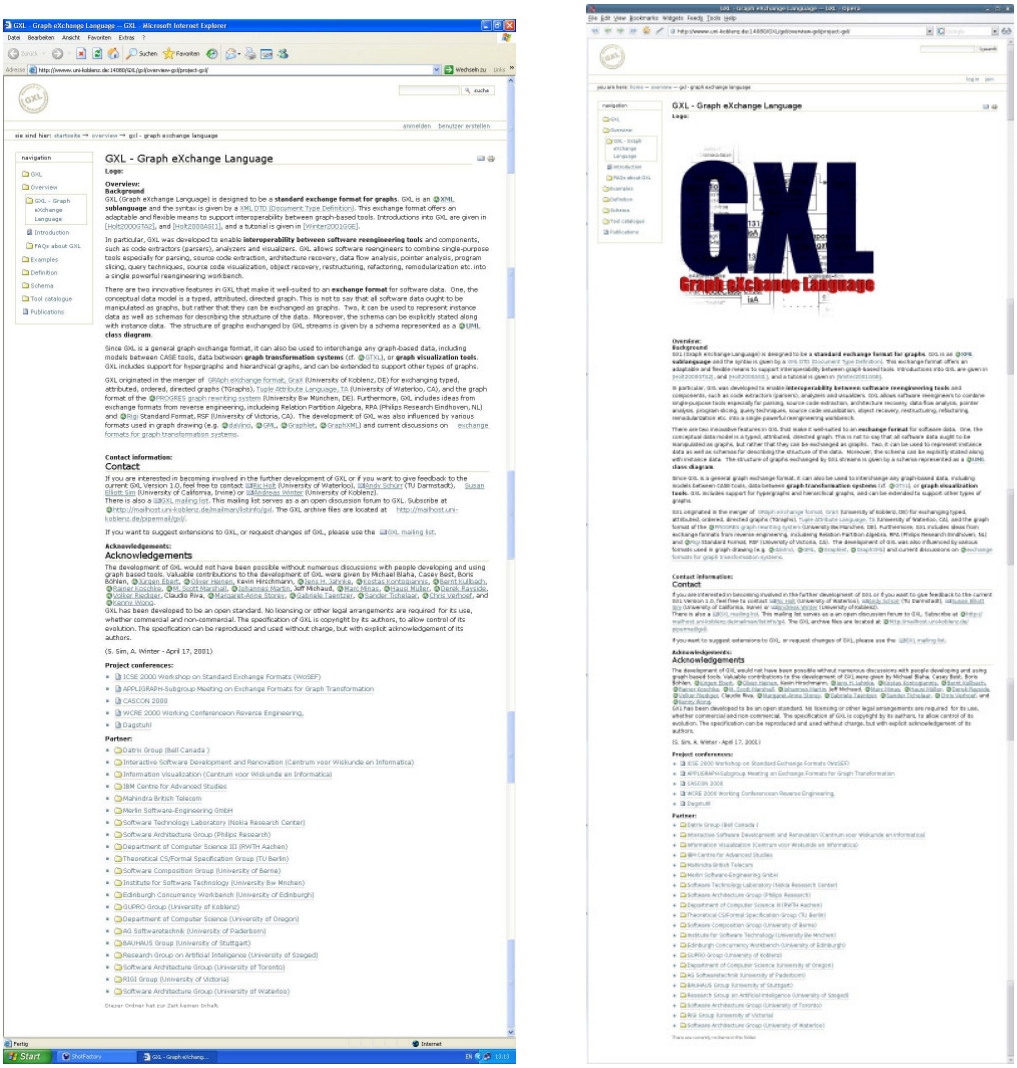

Abbildung 77: Darstellung der Webseite *Background* auf Internet Explorer 6.0 und Opera 9.01

Hinsichtlich der Einheitlichkeit und Korrektheit der Darstellung wurden, ausgenommen von den Bildern, die restlichen Kriterien eingehalten. Da die Darstellungsfehler bei den Bildern recht schwerwiegend sind, konnte dieser Test nicht als bestanden erklärt werden. Es handelt sich offenbar um ein systematisches Problem von Plone, dass Felder des Typs *Image* im manche Browsern nicht richtig angezeigt werden. Dieser Mangel muss in einem separaten Projekt außerhalb dieser Migration weiter verfolgt werden. Die Ergebnisbilder dieses Tests können im Ordner „Screenshots“ der mitgelieferten CD eingesehen werden.

8 Übergabe

Nach der Transformation eines Migrationspakets und seiner Überprüfung durch Regressionstest, wird im Kernbereich der Übergabe das migrierte Paket übergeben. Sobald

alle Migrationspakete erfolgreich migriert wurden und das Zielsystem vollständig erstellt wurde, kann die Ablösung des Legacy-Systems erfolgen [Ackermann (2005), S. 230].

8.1 Zielumgebung einrichten

Die Aktivitätsgruppe „Zielumgebung einrichten“ zielt auf die Bereitstellung und Konfiguration der für das Zielsystem zu verwendenden Soft- und Hardware [Ackermann (2005), S. 234 / 235]. Darin werden nur Komponenten beschrieben, die, abgesehen vom Legacy-System, zusätzlich installiert werden müssen.

Für die vorliegende Migration ist das Content Management System Plone die Grundlage des Zielsystems. Die Installationsdatei für Plone ist frei verfügbar. Die aktuelle Version von Plone lässt sich von der Website der Plone-Stiftung herunterladen. Für den Einsatz im Internet muss Plone auf einem Web Server installiert werden.

Neben der Kernsoftware Plone müssen auch die in Abschnitt 4 genannten Produktabhängigkeiten des Zielsystems berücksichtigt werden. Bevor das Zielsystem in Betrieb genommen und transformiert werden kann, müssen die dort genannten Produkte installiert worden sein.

Sind somit die technologischen Grundlagen geschaffen, kann die Legacy-Website über die Installation des Produkts *GXL_Website* migriert werden. Weitere Installationen sind für das Zielsystem nicht notwendig.

8.2 Übergabe planen

Auf die Planung der Übergabe entfallen mehrere Vorbereitungstätigkeiten. So werden Termine und Methoden zur Übergabe festgelegt. Für die vorliegende Migration wurde mit den Projektbeteiligten ein Termin vereinbart, wann die endgültige Übergabe durchzuführen ist. Es handelt sich um die Übergabe des vollständigen Systems zu einem festen Zeitpunkt. Zu diesem Termin kann das Legacy-System abgelöst werden, da alle Daten in das Zielsystem übertragen wurden.

Zu der Planung der Übergabe gehört auch die Entwicklung von Akzeptanztests. Für diese Migration wird angesichts der relativ unkomplizierten Struktur der zu migrierenden Website von umfangreichen Akzeptanztests abgesehen. Es ist aber eingeplant, dass die Funktionsweise des Zielsystems den zukünftigen Nutzern, dazu gehört insbesondere der Website-Betreiber, die Möglichkeit erhält, das Verhalten des Systems auf einem Testserver zu nutzen. Damit können die Nutzer das Verhalten der Ziel-Website untersuchen und dem Programmierer Rückmeldung geben.

Die Erstellung der Installationseinleitungen ist im vorliegenden Fall leicht. Wurde erst die Zielumgebung mit Plone und den benötigten Produkten eingerichtet, reicht die einfache Installation des Produkts *GXL_Website* für die Übergabe. Die Installation erfolgt über die Web-Oberfläche von Plone. Das Vorgehen hierfür ist wie folgt:

1. Zunächst muss der Ordner für das Produkt *GXL_Website* im *Products*-Ordner des Dateisystems von Plone gespeichert werden. Innerhalb des Programmordners von Plone ist dieser im Verzeichnis *Data* zu finden. Außerdem ist eine Internetverbindung erforderlich.
2. Aufruf der Installationszentrale innerhalb des Konfigurationsbereichs von Plone über einen Web-Browser. Hierfür muss der Anwender als Manager eingeloggt sein. Die Installationszentrale kann z.B. durch Anfügen von *„/prefs_install_products_form“* an die Adresse der Ziel-Website aufgerufen werden.

3. Anschließend kann das Produkt GXL_Website ausgewählt werden und installiert werden. Abbildung 78 stellt diesen Vorgang dar.

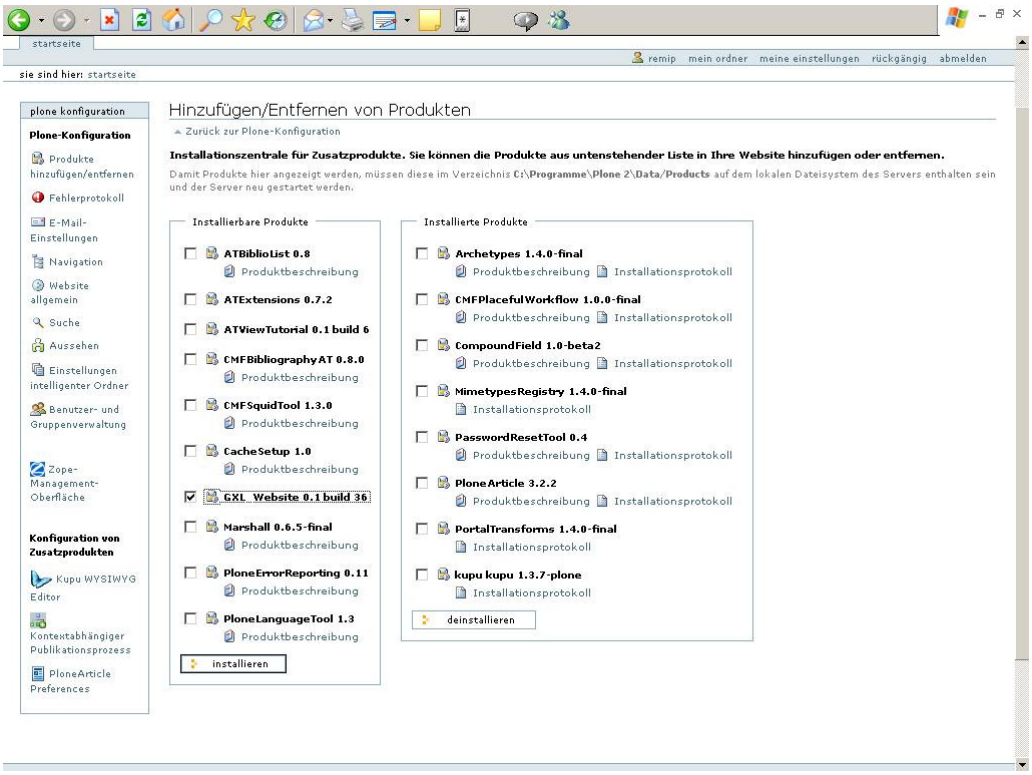

Abbildung 78: Installation des Produkts *GXL_Website*

Der Installationsvorgang dauert einige Minuten. Nach erfolgreichem Abschluss der Installation wird die gleiche Seite nochmals angezeigt, diesmal mit *GXL_Website* auf der rechten Seite bei den installierten Produkten.

8.3 Supportmaterial entwickeln

Die Erstellung einer Dokumentation für das neue Zielsystem ist für die Übergabe unerlässlich. Es muss gewährleistet sein, dass die zukünftigen Nutzer das Zielsystem auch richtig bedienen können. Dazu gehört einerseits die Entwicklung von Endanwender-Dokumentation als auch andererseits die Erstellung von Trainingsmaterial zur schnellen Eingewöhnung.

Innerhalb der Abschlussarbeit wurde bereits ausführlich auf den grundsätzlichen Aufbau von Plone und die Nutzung der wesentlichen Funktionen zur Verwaltung von Content eingegangen (vgl. Abschnitt 1.5 und 4.3). Somit liegt bereits eine grundlegende Dokumentation für Plone vor. Für Plone existieren zusätzlich noch zahlreiche Bücher, Tutorials und Newsgroups, die eine erweiterte Auseinandersetzung mit diesem Thema ermöglichen. Durch die detaillierte Beschreibung des Datenmodells und Benutzungsschnittstellen des Zielsystems existiert eine Dokumentation zur Durchführung von Wartungsarbeiten.

Auf die zusätzliche Erstellung von Trainingsmaterial wurde verzichtet, begründet durch die gut nachvollziehbare Nutzung von Plone über seine Benutzeroberfläche. Damit kann innerhalb der Trainingstreffen schnell der Umgang mit Plone erlernt werden. Außerdem existieren im Internet Demos, auf die zurückgegriffen werden kann.

8.4 Übergabe durchführen

Die Aktivitätsgruppe der Durchführung der Übergabe umfasst alle Aktivitäten zur konkreten Übergabe an die Endnutzer. Sie beinhaltet die Installation des Übergabepakets, die Durchführung und Auswertung der Akzeptanztests sowie das Training der Anwender [Ackermann (2005), S. 237].

Für die beschriebene Migration wurde an dieser Stelle die endgültige Installation des Zielsystems ausgeführt. Nach der Durchführung der Akzeptanztests der Nutzer wurden am Erscheinungsbild der Ziel-Website Anpassungen vorgenommen. Diese umfassten insbesondere Änderungen der standardmäßig von Plone angezeigten Elemente auf den Webseiten. In diesen Schritten wurden die Fußzeile, das Kolophon und, bis auf die Navigation, alle Makros der linken und rechten Spalte des main-Templates entfernt (vgl. Abschnitt 1.5.1). Sie enthielten zu viele Plone-spezifische Verweise oder unnötige zusätzliche Funktionalität für das Erscheinungsbild der neuen GXL-Website. Dieses Vorgehen erfolgte in Absprache zwischen Website-Betreiber und Programmierer.

Nach der Übergabe wurde der Website-Betreiber in die zukünftige Nutzung der Ziel-Website eingeführt. Dies geschah in einer interaktiven Trainingssitzung durch Verwendung des migrierten Zielsystems und konnte erfolgreich abgeschlossen werden.

8.5 Legacy-System ablösen

Nunmehr erfolgt die Überführung des gesamten Legacy-Systems in das Zielsystem und die Übergabe an einem bestimmten Termin an die Endnutzer. Es wurden alle Maßnahmen zur Gewährleistung der korrekten Nutzung der neuen Website durchgeführt. Das Zielsystem hat die Legacy-Website vollwertig ersetzt.

In der Aktivitätsgruppe „Legacy-System ablösen“ erfolgt die Abwicklung der verbliebenen Legacy-Komponenten. Hierfür wird zur Archivierung der alten Website die gesamte Ordnerstruktur kopiert und auf einem Datenträger abgelegt.

Für die endgültige Ablösung der Legacy-Website werden die über Internet verfügbaren Dateien gelöscht. Da die Ziel-Website sich nicht auf demselben Server befindet wie die alte GXL-Website, wird an Stelle der alten Adresse *http://www.gupro.de/GXL* eine Webseite *„index.html“* mit einer automatischen Weiterleitung zur neuen GXL-Website gespeichert. Damit ist gewährleistet, dass Besucher der Website, die noch die alte Internet-Adresse gespeichert haben, auf diese Weise zur neuen Website gelangen. Damit ist das Legacy-System abgelöst.

9 Konfigurations- und Änderungsmanagement

Das Konfigurations- und Änderungsmanagement beschäftigen sich mit den dynamischen Aspekten der Migration. Das Konfigurationsmanagement umfasst die Verwaltung der Änderungen der Artefakte, während das Änderungsmanagement Änderungswünsche im Verlauf der Migration erfasst [Ackermann (2005), S. 240]. Dabei werden auch Veränderungen des Legacy-Systems dokumentiert und auf die während der Migration erstellten Artefakte abgebildet.

9.1 Änderungsanfragen verwalten

Das Legacy-System stellte sich im Verlauf der Migration als sehr konstant heraus. In dieser Zeit waren keine strukturellen Anpassungen der Website notwendig. Damit entfiel die mitunter aufwändige Abstimmung zwischen Migrationsprojekt und Veränderungen des

Legacy-Systems. Diese Konstanz bezüglich des Legacy-Systems ist allerdings eher untypisch. Im Fall dieser Migration liegt es darin begründet, dass der Großteil der Inhalte sich schon seit mehreren Jahren auf der Website befindet und der Bedarf an Aktualisierung seitdem sehr gering war.

Allerdings ergibt sich bei der Fortentwicklung des Zielsystems stets die Notwendigkeit, bestimmte Stellen zu ändern. Zur Gewährleistung, dass alle Änderungsanfragen erfasst werden, sind diese in einem Dokument festgehalten und werden bei Bedarf an das Konfigurationsmanagement weitergeleitet.

9.2 Konfigurations- und Änderungskontrolle für das Projekt planen

Bei der Änderung und Versionierung der Artefakte des Projekts können vier grundlegende Gruppen unterschieden werden, die bei der Aktualisierung berücksichtigt werden müssen:

- Berichte
- Dokumentation
- Programmierung auf Ziel-Design-Ebene
- Programmierung auf Transformationsebene
- Programmierung auf Übergabeebene

Die während der Migration für die Treffen der Projektbeteiligten erstellten *Berichte* dienten vorrangig der Demonstration des Projektfortschritts (siehe Abschnitt 10). Die dabei erstellten Dokumente in textueller und bildhafter Form gingen allerdings auch in nicht unerheblichen Ausmaß in die Anfertigung der Artefakte ein, wie sie in der vorliegenden Arbeit zu finden sind. Die Berichte stellten nur eine Bestandsaufnahme dar und mussten nicht zur Gewährleistung der Integrität der Migration im Zeitverlauf angepasst werden. Sie stellen somit zwar einen Bestandteil der Artefakte dar, unterliegen allerdings nicht dem Änderungsmanagement. Für Berichte ist deren Archivierung im Rahmen des Konfigurationsmanagements ausreichend.

Die *Dokumentation* umfasst die Beschreibung der Migration auf Grundlage von Dokumenten. Innerhalb dieser Dokumente finden sich Festlegungen bezüglich der Migrationsstrategie, der Legacy-Analyse und des Designs. Sie stellen die Arbeitsgrundlage zur Verwaltung und Definition des Migrationsgegenstandes dar. Die Dokumentation erstreckt sich über alle Phasen und Bereiche der Migration. Somit besteht bei ihr ein hoher Aktualisierungsbedarf. Sobald es Änderungen innerhalb der Migration gibt, müssen diese Dokumente auf den neuesten Stand gebracht werden.

Die *Programmierung auf Ziel-Design-Ebene* umfasst die Modellbildung und die darauf folgende Implementierung des Zielsystems. Somit müssen bei Änderungen innerhalb der Programmierung im Ziel-Design alle davon betroffenen Elemente des Bereichs aktualisiert werden. Ebenso kann durch die Änderung der Dokumentation der Bedarf entstehen, dass das Ziel-Design und damit auch die Programmierung für das Zielsystem angepasst werden müssen.

Komplexer wird die Berücksichtigung der *Programmierung auf der Transformationsebene*. Neben den internen Abhängigkeiten und dem Einfluss der Dokumentation müssen an dieser Stelle bei Änderungen des Ziel-Designs im Allgemeinen auch Aktualisierungen der Transformationen vorgenommen werden. Damit erhöht sich innerhalb dieser Gruppe der Konfigurationsaufwand erheblich.

Am schwierigsten ist die Abstimmung innerhalb der *Programmierung auf Übergabeebene*. Da sich hier bereits migrierte Komponenten befinden, die transformiert und weiter entwickelt

wurden, wirken sich Änderungen in den vorgelagerten Phasen besonders gravierend aus. Sobald eine Änderung der Dokumentation, des Ziel-Designs oder der Transformationen vorgenommen wird, muss überprüft werden, ob diese sich auf das migrierte Paket auswirken.

Daraus ergeben sich zwischen den Artefakten Aktualisierungsabhängigkeiten, die in Abbildung 79 dargestellt sind.

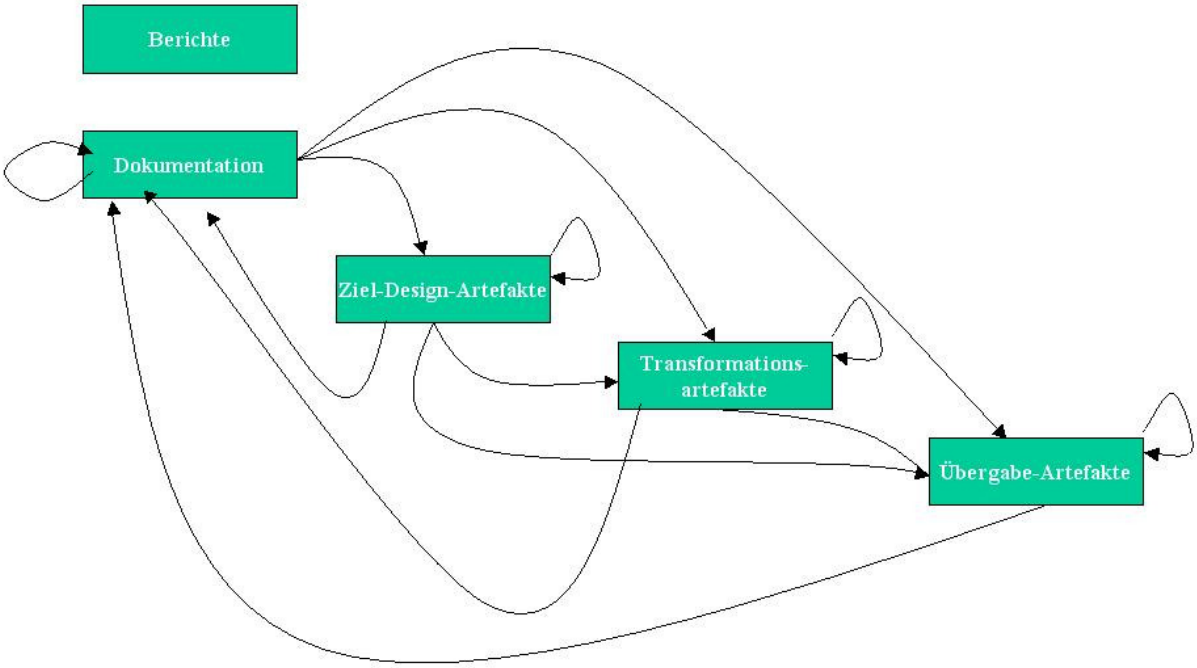

Abbildung 79: Aktualisierungsabhängigkeiten zwischen Migrationsartefakten

Der für die Durchführung der Migration zuständige Programmierer hat die Aufgabe, bei Änderungen die Aktualisierungsabhängigkeiten zu überprüfen und ggf. Anpassungen vorzunehmen. Es findet bis auf die Erfassung der Berichte keine durchgehende Versionierung statt. Stattdessen werden zu festgelegten Zeitpunkten Sicherheitskopien angefertigt, die die Wiederherstellung eines alten Migrationszustandes ermöglichen.

9.3 Umgebung für das Konfigurationsmanagement einrichten

Die technische Umgebung des Konfigurationsmanagements ist relativ einfach. Da nur eine Person, nämlich der Programmierer, die wesentliche Projektarbeit übernimmt, entfallen ansonsten aufwändige Koordinierungsmaßnahmen. Zur Durchführung des Konfigurationsmanagements kann nur er direkt auf die Artefakte zugreifen und diese zur Aktualisierung ändern.

9.4 Konfigurationsstatus überwachen und dokumentieren

Wie in Abschnitt 9.2 bereits erwähnt, wird keine für jeden Konfigurationsschritt nachvollziehbare Versionierung der Artefakte durchgeführt. An deren Stelle werden die Artefakte direkt aktualisiert und in regelmäßigen Abständen auf einem Datenträger gesichert.

9.5 Konfigurationselemente ändern und ausliefern

Nebenläufig zu den migrationspezifischen Aktivitäten wurden die Konfigurationselemente unter Berücksichtigung der Aktualisierungsbeziehungen geändert. Eine explizite Auslieferung der neu konfigurierten Elemente erfolgte durch die direkte Bearbeitung der Artefakte nicht.

10 Projektmanagement

Das Projektmanagement im Allgemeinen umfasst eine Vielzahl von Aktivitäten, die sich mit der Konzeption, Planung, Kontrolle und den Abschluss von Projekten befassen. Manche Aktivitäten, wie die Definition des Projekts, finden sich mit verändertem Schwerpunkt auch in den Kernbereichen des ReMiP wieder.

10.1 Neues Projekt konzipieren

Die Neukonzeption des Projekts der Migration der GXL-Website kann bereits in der Ausschreibung dieses Themas als Master-, bzw. Diplomarbeit gesehen werden. Zu diesem Zeitpunkt war die genaue Gestaltung des Themas noch nicht sehr detailliert. Dennoch war bereits damals eine Festlegung für die Zielumgebung der GXL-Website in Plone erfolgt.

Erst durch Interessensbekundung von Seiten des Verfassers dieser Arbeit wurde das Projekt weiterentwickelt. Durch Absprachen zwischen den Betreuern und ihm wurde das Ziel der Arbeit abgestimmt.

10.2 Projektumfang und Risiken bewerten

Im frühen Stadium des Projekts waren die Bedingungen dieser Arbeit noch nicht vollständig ausgehandelt. Zur Bewertung des Umfangs und der mit dem Projekt verbundenen Risiken war eine ausführliche Beschäftigung mit dem Arbeitsthema notwendig, das vor allem die Auseinandersetzung mit dem zukünftigen Zielsystem Plone und der Beurteilung der Legacy-Website umfasste.

Bei der Bewertung von Umfang und Risiko spielten bereits zahlreiche Elemente mit, die in der späteren Migrationsplanung von Bedeutung waren. Innerhalb dieser Teilaktivität des Projektmanagements wurden auch die Bereiche Legacy-Analyse, Ziel-Design und Mitarbeiterqualifizierung behandelt.

Auf Grundlage der ersten, globalen Bewertung des Legacy-Systems und der Abschätzung der Möglichkeiten des Zielsystems Plone konnten nun die Rahmenbedingungen der Arbeit bewertet und auf Machbarkeit evaluiert werden. Die wichtigsten Punkte bei dieser Festlegung finden sich im Abschnitt 3.2 zur Abgrenzung der organisatorischen Faktoren wieder.

Das größte Risiko bei der Projektdurchführung war die Einhaltung der Zeitvorgabe. Es musste gewährleistet sein, dass das Projekt innerhalb von sechs Monaten durchführbar ist, ansonsten kann es nicht den Bedingungen einer Master-, bzw. Diplomabschlussarbeit im Fach Informationsmanagement nach den Kriterien an der Universität Koblenz-Landau genügen.

Die bis dahin gewonnenen Erkenntnisse bezüglich des Aufwands und des (Zeit-)Risikos ließen darauf schließen, dass das Projekt zur Migration der Website nach Plone die Rahmenbedingungen einhalten würde. Die Machbarkeit wurde damit angenommen und der nächste Schritt innerhalb des Projektmanagements hinsichtlich der Planung des Projekts ausgeführt.

10.3 Projekt planen

Bis zur Aktivität „Projekt planen“ sind die Annahmen zur Durchführung des Projekts noch eher grundsätzlicher Natur. Es wurde festgestellt, dass das Projekt in Hinsicht auf die zur Verfügung stehenden Ressourcen durchführbar sein sollte. Nun erfolgt die Detaillierung des Projekts und die Überprüfung auf Machbarkeit durch weitergehende Planung der einzelnen Schritte.

Für die vorliegende Arbeit wurde die Projektplanung durch die Erstellung eines Zeitplans durchgeführt, der mit den Betreuern, und somit den Projektbeteiligten, abgesprochen wurde. Innerhalb von Zeitfenstern von zwei bis drei Wochen wurde die Bearbeitung thematischer Abschnitte und die Ablieferung von Artefakten festgelegt.

Wenn an dieser Stelle von Artefakten die Rede ist, so handelt es sich um vorläufige Ausarbeitungen von Abschnitten dieser Arbeit. Wie in Abschnitt 1.1 erwähnt, wurde der Ansatz gewählt, die Dokumente der Migration in den Textfluss der Arbeit einzubetten. Die Bearbeitung von bestimmten Abschnitten und logischen Einheiten deckt sich weitestgehend mit der von Artefakten.

Der erste Zeitplan stellte nur eine sehr grobe Zeitvorgabe dar. Die für die einzelnen Bereiche angesetzten Zeitfenster sollten einen idealen Verlauf darstellen und eine ungefähre Abschätzung des Aufwands ermöglichen. Die wichtigsten Informationen dieses initialen Zeitplans waren die Festlegung der direkt folgenden Schritte und die des geplanten Endzeitpunkts des Projekts.

10.4 Nächste Iteration planen

Die Planung der nächsten Iteration innerhalb des Projektmanagementzyklus umfasst die Definition des Umfangs und der in der Iteration folgenden Aktivitäten. Während der regelmäßigen Treffen der Projektbeteiligten zur Besprechung des Arbeitsfortschritts wurde zum Abschluss stets vereinbart, welche Abschnitte als nächstes zu bearbeiten sind. Dabei wurden die zu bearbeitenden Abschnitte nicht strikt nach Phasen im Sinne des ReMiP eingeteilt, sondern nach dem innerhalb der Zeitvorgabe von zwei bis drei Wochen durchführbaren Arbeitsumfangs. Die Phasen des ReMiP umfassten also häufig mehrere Iterationen innerhalb des Projektmanagements.

10.5 Iteration verwalten

Die Verwaltung des Fortschritts im Verlauf einer Iteration wurde durch den Programmierer durchgeführt. Er verfügte als Verfasser der Arbeit über den umfassendsten Einblick in die Vorgänge und die Entwicklung des Umfangs und der Risiken der Arbeit. Bei notwendigen Anpassungen der Iteration hat er Rücksprachen bei den Projektbeteiligten eingeleitet und in eigener Verantwortung die konkrete Ausgestaltung der Aktivitäten übernommen.

10.6 Phase abschließen

Zum Ende einer Iteration wurde, bei dem regelmäßigen Treffen zur Erläuterung des Arbeitsfortschritts, besprochen, ob eine Phase abgeschlossen werden kann. Die Ergebnisse wurden in Form eines Berichts an die Betreuer bzw. Projektbeteiligten abgegeben und auf Grundlage der eingereichten Dokumente der Abschluss der Phase erklärt.

10.7 Projekt abschließen

Der Abschluss des gesamten Projekts läuft parallel zu den Aktivitäten des Kernbereichs „Übergabe“. Während die Übergabe sich vor allem mit der technischen Abwicklung des Alt-Systems und deren Ersetzung durch das Zielsystem beschäftigt, wird in der Abschlussaktivität des Projektmanagements die Organisation des Projekts aufgelöst.

Dem Abschluss des Projekts ging ein letztes gemeinsames Treffen der Projektbeteiligten voraus, bei dem der vorläufige Entwurf des abschließenden Arbeitsdokuments besprochen wurde. Ebenso wurde festgestellt, dass die im Rahmen des Kernbereichs Übergabe

notwendigen Aktivitäten durchgeführt wurden. Erst nach Berücksichtigung der in der letzten Besprechung vorgebrachten Vorschläge wird durch die Ablieferung des Abschlussdokuments das Projekt abgeschlossen.

10.8 Projekt überwachen & kontrollieren

Neben den Tätigkeiten der Iterationsverwaltung, dem Abschluss der Phase oder des Projekts und der Planung der neuen Iteration muss der Fortschritt des Projekts ständig überwacht und kontrolliert werden. Während der Migration wurde diese Aktivität durch die regelmäßigen Treffen der Projektbeteiligten gewährleistet, bei denen zum Fortschritt Bericht erstattet wurde.

Die Evaluation des Fortschritts erfolgte dort auf verbaler Ebene. Bei der Berichterstattung der nächsten Iteration wurde allerdings nochmals explizit die Erreichung der vorgesehenen Ziele behandelt. Auf dieser Grundlage konnte bei den Sitzungen immer gezielt auf die Lage eingegangen werden. Da bei den Treffen die entscheidenden Projektbeteiligten anwesend waren oder zumindest durch die Berichterstattung informiert wurden, war die Projektkontrolle gewährleistet.

11 Mitarbeiterqualifizierung

Der Basisbereich der Mitarbeiterqualifizierung beschäftigt sich mit der Ausbildung der am Projekt beteiligten Mitarbeiter. Anders als in Abschnitt 8 zur Übergabe des Migrationspakets zur Planung und Durchführung der Trainingsmaßnahmen für die Endnutzer des Zielsystems, sind die Betroffenen hierbei das Migrationsteam und das Wartungsteam [Ackermann (2005), S. 248]. Es handelt sich also um die Programmierer innerhalb des Projekts.

11.1 Mitarbeitertraining vorbereiten

Für das Mitarbeitertraining ist die Ermittlung des Trainingsbedarfs und die Erstellung eines Trainingsplans wesentlich. Im Migrationsprojekt fungieren der Verfasser dieser Arbeit als Programmierer und der Website-Verantwortliche als Migrationsteam. Nach erfolgter Migration wird der Website-Verantwortliche Wartungsarbeiten übernehmen und steht damit stellvertretend für das Wartungsteam.

Erhebung des Trainingsbedarfs
In Hinsicht auf das Legacy-System war der Kenntnisstand beim Migrationsteam bereits sehr hoch. Beiden Beteiligten war die Funktionsweise der Website auf HTML-Basis ebenso wie der Umgang mit in XML gespeicherten Dokumenten bekannt. Auf Seiten des Programmierers gab es Kenntnisdefizite bezüglich des zur Verwaltung der Website verwendeten Werkzeugs *make*.

Der Kenntnisstand zum Zielsystem Plone vor allem beim Programmierer geringer. Während der Website-Betreiber bereits mit der Funktionsweise von Plone vertraut war und erste Anwendungsversuche durchgeführt hatte, war dem Programmierer Plone fast gänzlich unbekannt. Er konnte lediglich auf geringe Erfahrung im Umgang mit dem Web-Applikationsserver Zope zurückgreifen, auf dem Plone basiert. Ebenso war die Nutzung von *LitDB* zum Eintragen und Auslesen von Bibliographiedaten dem Programmierer nicht ausreichend bekannt. Der Website-Verantwortliche verfügte hingegen um tiefer gehende Kenntnisse dieser Datenbank, hatte allerdings noch keine Erfahrungen mit dem Auslesen der Daten nach Plone.

In Hinsicht auf die Anwendung der Migrationsumgebung und der Nutzung von ArchGenXML (siehe Abschnitt 1.5.6) wurde auch die Kenntnis von Klassendiagrammen und

der Umgang mit Werkzeugen zu deren Erstellung vorausgesetzt. Sowohl der Programmierer als auch der Website-Verantwortliche besitzen diese Kompetenzen.

Damit wurde klar, auf welchen Gebieten Lern-, bzw. Trainingsbedarf, bestand. Der Programmierer musste sich in die grundlegende Funktionsweise von *make* einarbeiten, um die Semantik der *Makefiles* nachvollziehen zu können. Darauf aufbauend konnte die Legacy-Analyse erfolgen.

Auch war es für den Programmierer notwendig, das Content Management System Plone zu erlernen und den Umgang mit diesem für die Entwicklung von eigenen Anwendungen mit Python und der Plone-eigenen Template-Sprache. Um die Nutzung der Literaturdatenbank für die Verwaltung der Bibliographieeinträge zu gewährleisten, benötigt der Programmierer auch auf diesem Gebiet Kenntnisse. Der Website-Verantwortliche musste sich lediglich das technische Know-How zum Auslesen der Bibliographiedaten nach Plone aus der Literaturdatenbank aneignen.

Zur Aneignung der erforderlichen Kenntnisse um das Legacy- und das Zielsystem wurde eine Kombination von Selbststudium und Austausch zwischen den Beteiligten des Migrationsteams angewandt. Auf Basis von Büchern und im Internet verfügbaren Dokumenten wurde im Verlauf der Migration das Wissen über die oben angesprochenen Komponenten aufgebaut. In unregelmäßigen, aber häufigen Treffen zwischen den Beteiligten des Migrationsteams wurden die Kenntnisse weiter vertieft.

Da das Wartungsteam eine Teilmenge des Migrationsteams darstellt und angesichts der Kenntnis des Zielsystems keine zusätzlichen Kompetenzen verlangt werden, wird von der zusätzlichen Beschreibung der Maßnahmen für das Wartungsteam abgesehen.

11.2 Migrationsteam trainieren

Die geplanten Maßnahmen zum Training des Migrationsteams werden in der Aktivitätsgruppe „Migrationsteam trainieren“ umgesetzt. Im Verlauf der Migration und dabei vor allem in den frühen Phasen wurde der Trainingsbedarf nachgeholt. Mit genauerem Kenntnisstand der zu Grunde liegenden Technologien und Werkzeuge, ergaben sich während des Trainings auch immer neue Gebiete, auf denen Bedarf für weiteres Studium entdeckt wurde.

Der Trainingserfolg wurde nicht durch Tests oder sonstige Kontrollen überprüft. Erst bei der Anwendung des Gelernten im Rahmen der Legacy-Analyse und des Ziel-Designs zeigte sich der Fortschritt. Durch regelmäßige Treffen zur Kontrolle des Projektfortschritts konnte auf diese Weise indirekt auf den Lernerfolg geschlossen werden. Insgesamt war dieses Verfahren des „Learning by Doing“ erfolgreich.

11.3 Wartungsteam trainieren

Die Aktivitäten in diesem Abschnitt sind analog zu Abschnitt 11.2.

12 Migrationsumgebung

Durch den Basisbereich „Migrationsumgebung“ wird die technische Infrastruktur beschrieben, die die Durchführung des Migrationsprojekts gewährleistet [Ackermann (2005), S. 250]. Die damit verbundenen Aktivitäten sorgen für die Beschaffung, die Installation und den Betrieb der dafür ausgewählten Systeme.

12.1 Projektumgebung vorbereiten

Im Rahmen der Vorbereitung der Projektumgebung werden die für die Durchführung des Projekts benötigten Soft- und Hardware-Voraussetzungen geschaffen. Für das vorliegende Migrationsprojekt stellen sich, abgesehen vom Vorhandensein einer Internetverbindung, keine Ansprüche an die Hardware, die über die Standardausstattung eines handelsüblichen PCs hinausgehen.

Die Software für verschiedene Zwecke eingesetzt. Im Abschnitt 3 zur Legacy-Analyse wurde auf Validatoren zur Überprüfung der Standardkonformität von HTML- und XML-Code zurückgegriffen. Die Werkzeuge leisten wertvolle Dienste zur quantitativen und qualitativen Analyse der Konformitätsfehler.

Für das Migrationsprojekt wurde der Validator des World Wide Web-Konsortiums (W3C) ausgewählt.[30] Die Nutzung dieses Validators ist kostenlos und er ermöglicht die Überprüfung der Gültigkeit sowohl von HTML als auch von XML-Code für die gängigen Standardversionen. Der Vorteil dieses Validators liegt auch darin, dass er neben der Ausgabe der Anzahl der Fehler auch zu jedem eine detaillierte Beschreibung liefert. Damit ist es möglich, die Art des Fehlers zu analysieren und davon Code-Verbesserungsmaßnahmen abzuleiten. Abbildung 80 zeigt exemplarisch den Anfang der Ausgabe des Validators für die *Background*-Webseite der GXL-Website.

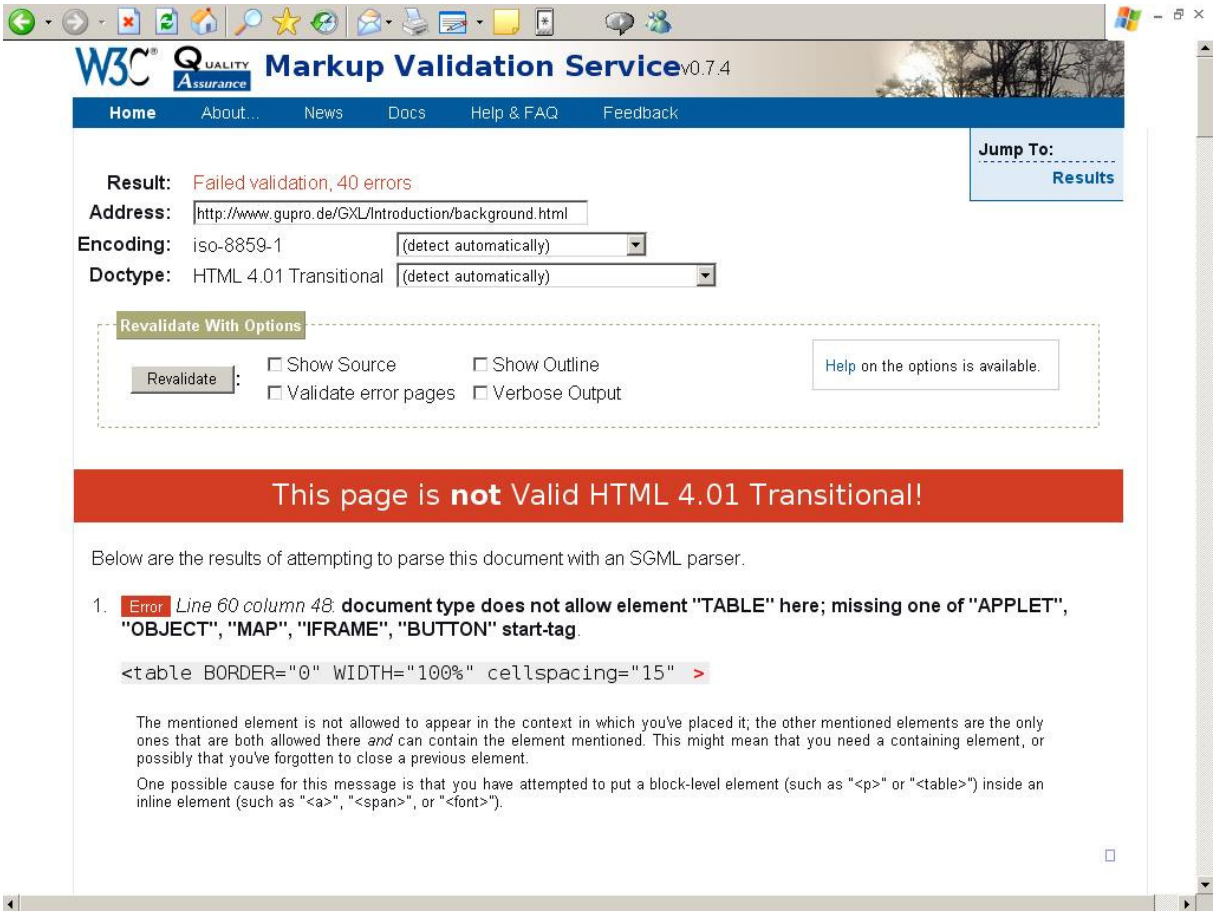

Abbildung 80: Ausgabe des W3C Validators für die *Background*-Webseite

[30] Zu finden unter http://validator.w3.org/.

Für die vereinfachte Analyse der Legacy-Website auf Grundlage der vorhandenen Hyperlinks wurde der Web Crawler *WebLoupe* eingesetzt.[31] Durch die Angabe von Parametern ist es möglich, die Hyperlinks innerhalb einer Webseite und der mit ihr verbundenen Webseiten tabellarisch aufzulisten und als Graph darstellen zu lassen. Für das Migrationsprojekt wurde auf die Funktionalität der Anzeige der Hyperlinks in tabellarischer Form zurückgegriffen. So werden durch einen einfachen Crawl-Vorgang (also das Durchsuchen der Hyperlinks einer Webseite) komfortabel alle Hyperlinks der interessierenden Webseiten aufgelistet und können anschließend ausgewertet werden. Abbildung 81 zeigt die Ausgabe aller Hyperlinks auf der *Background*-Webseite der GXL-Website mit WebLoupe.

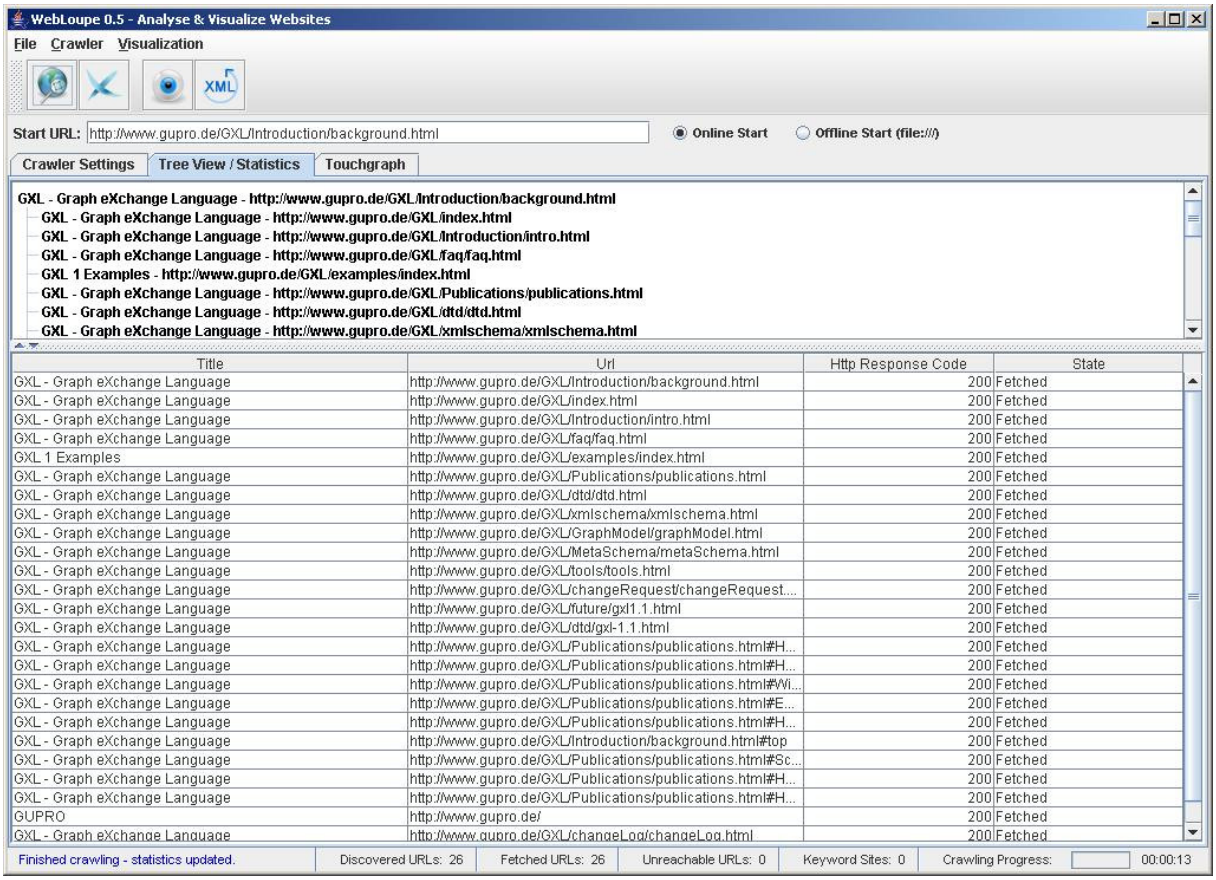

Abbildung 81: Ausgabe der Hyperlinks der Webseite *Background* durch WebLoupe

Ein wichtiges Werkzeug zur Beschreibung sowohl des Legacy- als auch des Zielsystems war das UML-Werkzeug *ArgoUML*.[32] Durch dieses Open Source-Tool können die wichtigsten Diagrammtypen der UML grafisch erstellt werden. Im Zusammenhang mit dem Migrationsprojekt wurden vor allem Klassen- und Aktivitätsdiagramme verwendet.

Besondere Bedeutung erlangt ArgoUML für die Entwicklung von Content-Typen für Plone unter Nutzung des Kommandozeilen-basierten Werkzeugs *ArchGenXML*.[33] Dieses Code-Generierungstool übersetzt u.a. Klassendiagramme mit festgelegter Semantik in die programmatorische Beschreibung von Content-Typen. Neben anderen Formaten kann ArchGenXML auch das Speicherformat ZARGO für UML-Diagramme, die in ArgoUML erstellt wurden, interpretieren. Damit stellen ArgoUML und ArchGenXML innerhalb des Migrationsprojekts gemeinsam die Grundlage für die Entwicklung des Zielsystems dar.

[31] Zu finden unter http://www.webloupe.de.vu/.
[32] Zu finden unter http://argouml.tigris.org/.
[33] Die Produkt-Webseite befindet sich unter http://plone.org/products/archgenxml.

Innerhalb der Transformation wurde kein konventionelles Tool zur Übertragung der Legacy-Website in das Zielsystem angewandt. Stattdessen wurden die Transformationsroutinen vollständig neu programmiert. Da das Zielsystem Plone in *Python* programmiert ist und damit über diese Programmiersprache ein einfacher Zugang zu den internen Funktionen von Plone möglich ist, wurde beschlossen, die Transformationen in ebenfalls Python zu programmieren. Diese Festlegung auf Python zur Programmierung stellt somit ebenfalls eine Entscheidung für die Gestaltung der Testumgebung dar.

13 Abschließende Bewertung

Nach der Erläuterung der Grundlagen zum Referenz-Prozessmodells ReMiP und der technologischen Rahmenbedingungen in Abschnitt 1, sowie der konkreten Anwendung des ReMiP zur Migration der Website des GXL-Projekts in Abschnitt 2, bildet Abschnitt drei das Resümee. Zum Abschluss werden Erfahrungen bei der Anwendung des ReMiP aufgezeigt und der für die einzelnen Bereiche benötigte Aufwand beschrieben und dargestellt.

13.1 Erfahrungen bei der Anwendung des ReMiP

Das Referenz-Prozessmodell ReMiP erwies sich im Verlauf der Migration der GXL-Website als gute Orientierung für die strukturierte Durchführung der Migration. Durch die ausführliche Beschreibung und den anpassbaren Aufbau dieses Prozessmodells, konnten die für die Migration wichtigen Passagen durchlaufen und auf die aktuelle Migration abgestimmt werden.

Die im ReMiP aufgezeigten Abhängigkeiten zwischen den einzelnen Kernbereichen, wie zwischen Legacy-Analyse und Strategie-Auswahl, sind in der Beschreibung der einzelnen Abschnitte gut nachvollziehbar. Da das Prozessmodell diese Abhängigkeiten explizit darstellt, können diese bei der Projektplanung berücksichtigt werden und somit den Umfang der migrationsspezifischen Aktivitäten kontrollierbar machen.

Für den vergleichsweise geringen Umfang des Projekts erwiesen sich manche Elemente und geforderte Artefakte als sehr umfangreich. Allein schon die Darstellung des Prozesses in Form dieser Arbeit in einer weniger an den einzelnen Artefakten orientierten Form, erwies sich als aufwändig. Besonders bei kleineren Projekten muss durch den Anwender eine Skalierung des ReMiP erfolgen, um den Aufwand der Dokumentation dem der technischen Implementierung anzupassen.

Insgesamt konnte der ReMiP als Referenz-Prozessmodell für Software-Migrationen validiert werden. Die während der Migration der GXL-Website notwendigen Aktivitäten waren vollständig durch den ReMiP erfasst und konnten für die Anwendung angepasst werden. Die klare vorgegebene Struktur hat die Durchführung der Migration wesentlich unterstützt.

Kernbereich Legacy-Analyse /Aufbereitung
Gewisse Probleme ergaben sich bei der Belegung des Kernbereichs Legacy-Analyse / Sanierung. Der Bereich der Legacy-Analyse besitzt stark normativen Charakter, da sich aus ihm wesentliche Elemente des Ziel-Designs und der funktionalen Anforderungen ableiteten lassen. Allerdings existiert bereits bei der Aktivitätsgruppe „Legacy-System detailliert reverse-engineeren“ in der Aktivitätsbeschreibung ein Verweis auf den Einfluss der ausgewählten Migrationsstrategie [Ackermann (2005), S. 186].

Weniger klar wurde die Beziehung zwischen der Legacy-Aufbereitung und dem Ziel-Design dargestellt. Tatsächlich wirkt sich die Umgebung des Zielsystems ganz wesentlich auf die Wahl der Sanierungsmaßnahmen aus, wie z.B. Datenformate oder sinnvolle Einteilung des Legacy-Systems in Migrationspakete. Die Legacy-Aufbereitung ist also vollständig von Faktoren bestimmt, die außerhalb des Kernbereichs existieren.

Tatsächlich handelt es sich im Vergleich zur Analyse hier bereits um einen aktiven vorbereitenden Eingriff in das Alt-System für die Transformation. Zur besseren Beschreibung und Differenzierung der Erfordernisse dieses Aufgabenfeldes, wäre auch eine Ausgliederung der Legacy-Vorbereitung in einen eigenen Kernbereich denkbar. Aus der Definition von Ackermann geht zwar die Intention hervor [Ackermann (2005), S. 187], die Einordnung der

Legacy-Vorbereitung in einen logisch eher dem Ziel-Design nachgelagerten Bereich könnte zusätzliche Klarheit schaffen.

In Anlehnung an die Darstellung von [Ackermann (2005), S. 160] zur Verdeutlichung der Abhängigkeiten zwischen den Aktivitäten, stellt Abbildung 82 einen strukturellen Erweiterungsvorschlag des ReMiP dar. Die durchgezogenen Pfeile deuten auf den logischen, sequenziellen Datenfluss, wie er sich auch in der Reihenfolge der Benennung der Kernbereiche wieder findet. Die gestrichelten Pfeile entsprechen den migrationsspezifischen Abhängigkeiten, die zwischen den Kernbereichen bestehen.

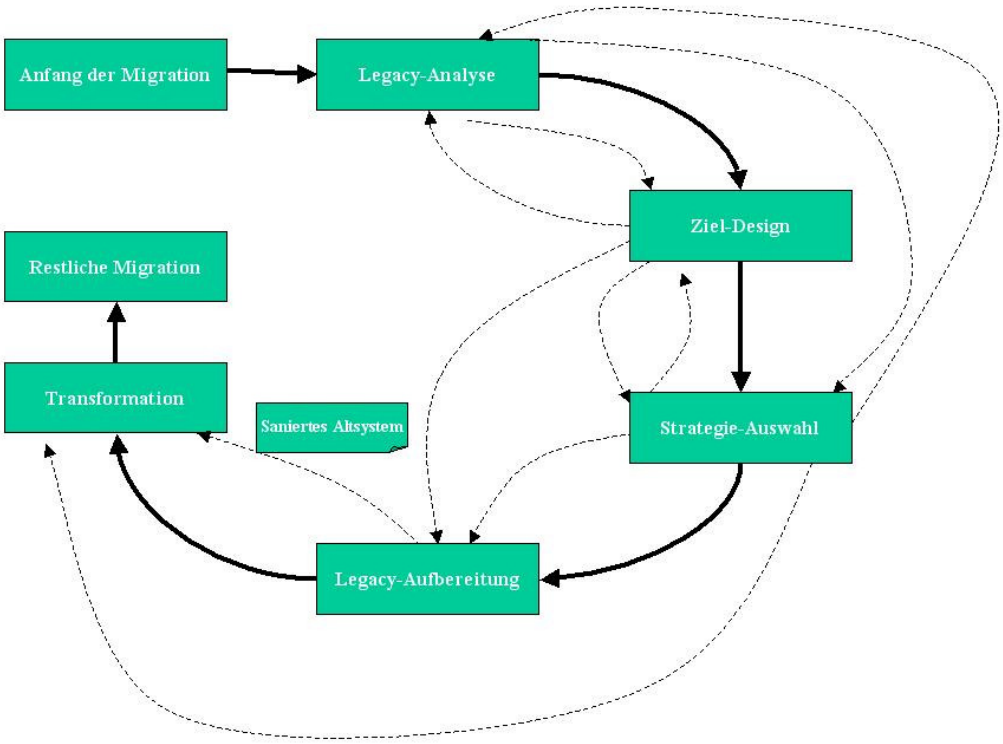

Abbildung 82: Angepasstes Modell zum ReMiP (Ausschnitt)

Der Bereich der Legacy-Aufbereitung wurde aus dem Kernbereich der Legacy-Analyse entfernt und bildet innerhalb des Modells einen eigenen Kernbereich. Dieser Kernbereich zeichnet sich dadurch aus, dass er vom Ziel-Design bezüglich der technischen Vorbedingungen der Sanierung und von der Strategie-Auswahl bezüglich der Migrationsstrategie bestimmt wird. Sein Artefakt zum nächsten Kernbereich der Transformation ist das sanierte Alt-System. Die Artefakte zwischen den anderen Kernbereichen wurden zur Wahrung der Übersichtlichkeit ausgelassen.

13.1.1 Nicht durchgeführte Aktivitäten

Im Rahmen der Migration der GXL-Website wurden auf Grund der konkreten Erfordernisse nicht alle Aktivitäten des ReMiP in der Durchführung der Migration zu berücksichtigt. Die davon betroffenen Aktivitäten umfassen die Thematiken der temporären Komponenten und der Programmierung. Die für die Migration notwendigen Aktivitäten erwiesen sich als weniger umfangreich, als durch das Prozessmodell abgedeckt. Der ReMiP musste, wie als Referenzprozess gedacht, angepasst werden. In komplexeren Migrationsprojekten mit

geänderten Bedingungen ist die Durchführung der hier ausgelassenen Aktivitäten sehr wahrscheinlich.

Temporäre Komponenten
Alle Aktivitäten, die mit der Spezifikation und der Implementierung temporärer Komponenten zusammenhängen, wurden während dieser Migration nicht durchgeführt. Durch die Wahl einer „Big Bang"-Umstellungsstrategie und die geringe Komplexität des Legacy-Systems wurde es nicht notwendig, diese Komponenten zu definieren. Das komplette System konnte innerhalb eines Transformationsschrittes in das Zielsystem überführt werden und bedurfte somit keiner Unterstützung durch Schnittstellen, die die Interoperabilität zwischen Alt- und Zielsystem herstellen.

Zu diesen Aktivitäten gehören die folgenden:

- Kernbereich „Anforderungsanalyse" / Aktivitätsgruppe „Anforderungen ermitteln": Anforderungen an temporäre Komponenten definieren, Anforderungen an Wrapper definieren
- Ziel-Design / Übergangsarchitektur definieren
- Transformation / Komponenten implementieren: Gateways implementieren
- Übergabe / Übergabe durchführen: Gateways installieren
- Übergabe / Legacy-System ablösen: Gateways eliminieren

Das Auslassen der Durchführung dieser Aktivitäten hängt mit der Wahl der Migrationsstrategie zusammen. Wäre auf Grund der Struktur des Legacy-Systems die Umsetzung der Migration in einem inkrementellen Ansatz vorteilhaft gewesen, so wäre wahrscheinlich ein Großteil der oben genannten Aktivitäten durchgeführt worden.

Programmierung der Funktionalität
Bezüglich der Funktionalität des Legacy-Systems und des Zielsystems existierte ein Sonderfall. Einerseits verfügte das Legacy-System nur über einen sehr geringen Anteil an Programmierung, der zudem wegen der Wahl einer Datenkonversionsstrategie nicht durch die Migration zu transformieren war (vgl. Abschnitt 5.1). Damit entfielen die Aktivitäten zur Transformation dieses Programmcodes.

Zum anderen verfügte das Zielsystem bereits standardmäßig über die notwendige Funktionalität zur Verwaltung der Website. Somit war es auch nicht erforderlich, neue Komponenten zu implementieren. Die einzige Programmierung, die unter Berücksichtigung der Migrationsstrategie auszuführen war, bestand aus der Definition der Content-Typen. Diese wurde durch die Modellierung eines Klassendiagramms und die Transformation des Diagramms in Python-Code durch das Werkzeug ArchGenXML durchgeführt.

Durch die Strategie der datenorientierten Konversion und die umfangreiche Funktionsausstattung des Zielsystems Plone wurden folgende Aktivitäten reduziert oder entfielen sogar ganz:

- Ziel-Design / Programme entwerfen: diese Aktivitätsgruppe wurde erheblich reduziert und wurde durch das Datenmodell fast vollständig vorgegeben
- Transformation: sämtliche Arbeiten zur Transformation von Programmcode entfielen
- Transformation / Komponenten implementieren: diese Aktivitätsgruppe entfiel

Auch hier ist die untergeordnete Behandlung der Programmierung von Funktionalität während der Migration auf die besonderen Rahmenbedingungen der Migration zurückzuführen. Die Wahl einer anderen Migrationsstrategie hätte verstärkte Transformationsaktivitäten auf Programmcode-Ebene zur Folge gehabt. Ebenso wäre durch die Wahl eines weniger funktionsreichen Zielsystems auch die Neuentwicklung von Komponenten denkbar gewesen.

13.1.2 Quantifizierung des Aufwands bei Ausführung der Aktivitäten

Durch die praktische Anwendung des ReMiP war es möglich, Erfahrung mit der Intensität der Aktivitäten im Migrationsverlauf zu sammeln. In der Arbeit von Ackermann konnte eine solche Bestimmung nicht durchgeführt werden, da sie sich auf die Ausarbeitung des theoretischen Referenz-Prozessmodells konzentriert hat und keine praktischen Werte vorlegen konnte [Ackermann (2005), S. 163].

Die vorliegende Arbeit bietet nun aber die Möglichkeit, durch die konkrete Anwendung des ReMiP zur Migration der GXL-Website solche Werte zu liefern. Die Erfahrungen im Umgang mit dem Referenz-Prozessmodell in Bezug auf den benötigten Aufwand zur Durchführung der Aktivitäten werden in diesem Abschnitt beschrieben.

Zur Darstellung des Aufwands der Bearbeitung werden die migrationsspezifischen Kernbereiche und ihre unterstützenden Basisbereiche im Phasenverlauf des Migrationsprozesses abgebildet. Die Intensität wird durch die Verwendung von „Hügeln“ verdeutlicht, die je nach Ausprägung in dem jeweiligen Abschnitt für hohe oder niedrige Intensität der Ausführung stehen. Diese Darstellungsweise ist, wie auch Ackermann verdeutlicht, als so genannter Hump-Chart im Rational Unified Process (RUP) üblich [Ackermann (2005), S. 163].

Die Höhe der „Hügel“ zur Angabe der Intensität für jede Aktivität wurde nach der subjektiven Einschätzung gewählt. Dabei stellt sie keine willkürliche Festlegung dar, sondern wird im Anschluss begründet. Die Grafik muss aber dennoch eher als eine grobe Abschätzung verstanden werden. Abbildung 83 zeigt die Intensitätsverteilung bei der Anwendung des ReMiP für die Website-Migration.

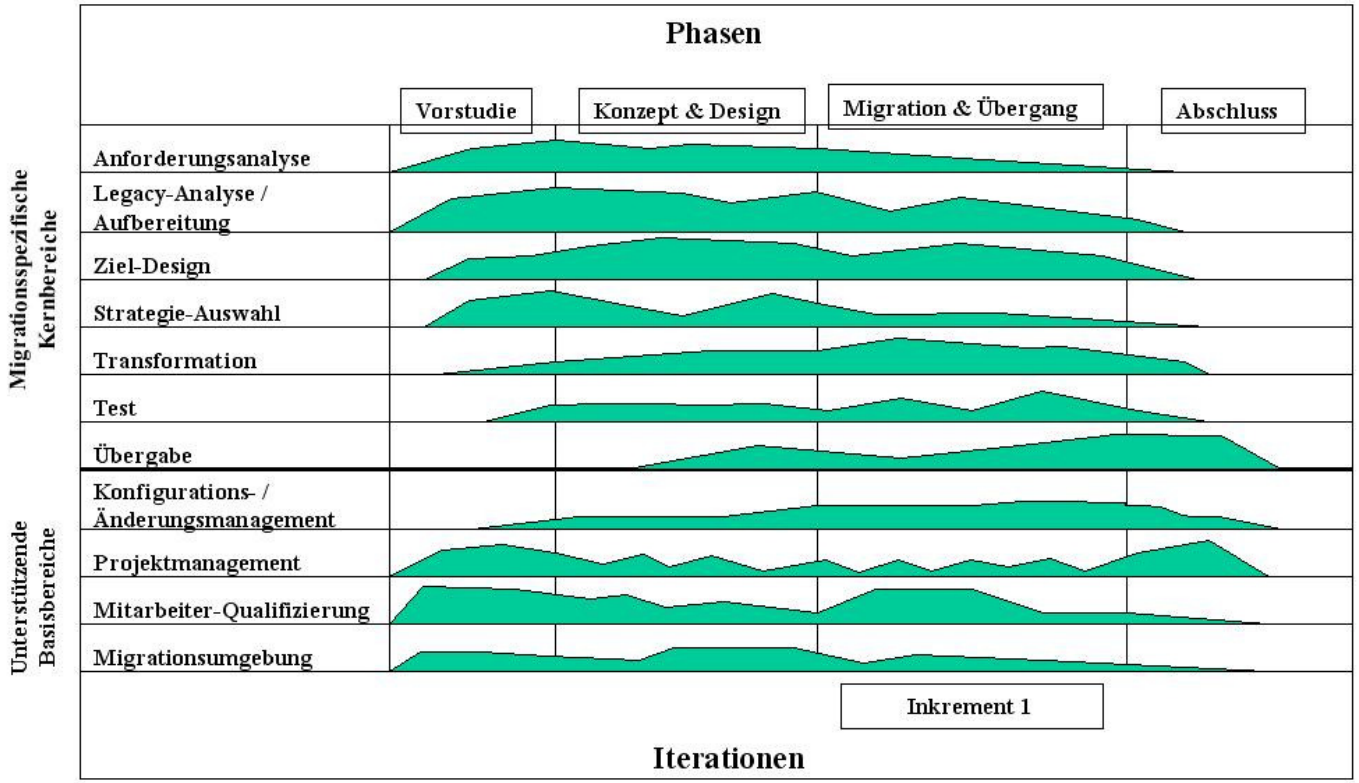

Abbildung 83: Hump-Chart zur Intensität der Aktivitäten im Migrationsverlauf

Anforderungsanalyse
Die Anforderungsanalyse wurde erwartungsgemäß besonders in den Phasen der Vorstudie und zu Beginn von Konzept & Design besonders intensiv ausgeführt. Zunächst standen Aktivitäten wie die Identifizierung der Stakeholder und die erste Ableitung nicht-funktionaler Anforderungen im Mittelpunkt. Durch die immer detailliertere Untersuchung des Legacy-Systems in der Legacy-Analyse ergaben sich auch zunehmend die funktionalen Anforderungen, die direkt aus der Struktur des Alt-Systems übernommen werden konnten. Deshalb reicht die hohe Intensität der Anforderungsanalyse noch weiter in die Phase Konzept & Design hinein.

Im Verlauf der Migration ergab sich neben der genauen Kenntnis des Legacy-Systems auch eine tiefere Auseinandersetzung mit der Leistungsfähigkeit des Zielsystems. Es stellte sich heraus, dass bestimmte Anforderungen, die anfangs definiert wurden, nicht unter Einhaltung des Projektrahmens durchgeführt werden konnten. Durch das Management der Anforderungen mussten während der Phase Konzept & Design Anpassungen vorgenommen werden. So ergibt sich innerhalb dieser Phase ein temporäres Ansteigen der Intensität der Aktivität der Anforderungsanalyse.

Dazu gehörte die Anforderung nach Übertragung des Designs und der Gestaltung der Website im Zielsystem. Da das umfassende Design der Ziel-Website mit der Programmierung vieler Templates verbunden gewesen wäre, wurde auf die Funktionalität von Archetypes zur automatischen Erstellung von Ansichtsseiten zurückgegriffen. Diese Darstellung genügt nicht sehr hohen Anforderungen an das Design, dennoch stellt die den Inhalt der Webseite annehmbar und verständlich dar.

Legacy-Analyse / Aufbereitung
Die Auseinandersetzung mit dem Aufbau des Zielsystems durch den Kernbereich der Legacy-Analyse stellt von Anfang bis Ende eine wichtige Tätigkeit dar. Zu Beginn musste in der Vorstudie bereits für die globale Bewertung des Alt-Systems eine recht intensive Analyse durchgeführt werden.

Der Aufwand für die Legacy-Analyse verstärkte sich zunehmend im Übergang zur Phase Konzept & Design. Jetzt wurde die detaillierte Untersuchung des Alt-Systems zur Ableitung der Funktionalität für das Zielsystem notwendig. Die Untersuchung der Website in der in Abschnitt 3.3 geschilderten Intensität führte zu Beginn der Phase zu einem sehr hohen Arbeitsaufwand.

Im Verlauf der Migration wurden innerhalb des Ziel-Designs durch den Perspektivenwechsel hin zum Zielsystem die Notwendigkeit für die Untersuchung weiterer Elemente festgestellt. Insbesondere innerhalb des Entwurfs der Transformationsregeln wurde erkannt, dass das Navigationsmodell der Website für die Programmierung weiter zu detaillieren ist. In Rückgriff darauf wurden die Webseiten in Pakete eingeteilt, wie in Abschnitt 3.4 nachzuvollziehen. Deshalb ergibt sich zum Ende der Konzept & Design-Phase ein temporärer Anstieg der Intensität.

Die Durchführung der Sanierungsaktivitäten für den HTML-Quelltext der Webseiten führte zu einem Anstieg der Intensität in der Phase Migration & Übergang. Dabei wurde die Sanierung ausnahmsweise sehr spät durchgeführt, nämlich nur kurz vor der Übergabe des Zielsystems. Die späte Durchführung dieser Aktivität ist allerdings auf die Besonderheit des in Plone integrierten HTML-Editors zurückzuführen, der die frühere Ausführung nicht unterstützt (vgl. Abschnitt 3.4).

Innerhalb der Phase Abschluss sind bereits alle Legacy-Analyse-Tätigkeiten ausgeführt und es entsteht nur noch durch die Archivierung des Alt-Systems ein gewisser Aufwand.

Ziel-Design
Das Ziel-Design kommt bereits zu einem frühen Zeitpunkt zu tragen. Durch die globale Bewertung des Legacy-Systems und die Notwendigkeit der Erarbeitung einer Migrationsstrategie, müssen bereits im Verlauf der Vorstudie die Architektur und die technischen Möglichkeiten des Zielsystems bekannt sein. Somit weist das Ziel-Design in dieser Phase bereits eine feststellbare Intensität auf.

Die konkrete Ausbildung vom Design des Zielsystems ergibt sich hingegen schwerpunktmäßig in der Phase Konzept & Design. In zunehmendem Maße übernimmt das Ziel-Design frei gewordene Kapazitäten der Legacy-Analyse und bildet das Legacy-System in die neue Zielumgebung ab. Hierbei entwickelt sich vor allem im mittleren Abschnitt dieser Phase ein besonders hohes Arbeitsaufkommen, das durch die Entwicklung von Designalternativen und die Erprobung der neuen Zielumgebung erklären lässt.

Nach der Ausarbeitung des Entwurfs für das Zielsystem übernimmt zum Ende der Phase „Konzept & Design“ die Ausarbeitung der Transformationsregeln den Hauptanteil des Aufwands. Im Verlauf der Migration wurde ein Großteil der Arbeit für die Entwicklung und Anpassung dieser Regeln aufgewandt. Die hohe Intensität trägt sich auch weiter in die Phase Migration & Übergang, da bei der Ausführung der Transformationen immer wieder Verbesserungsmöglichkeiten bei der Formulierung der Transformationsregeln entdeckt wurden.

Erst mit Abschluss der eigentlichen Migration nimmt die Intensität des Bereichs Ziel-Design in Hinblick auf die Weiterentwicklung der Transformationsregeln ab. Zwischen Übergang und Abschluss ergibt sich durch die Erstellung der Dokumentation oftmals der Bedarf das Ziel-Design an bestimmten Stellen zu verfeinern.

Strategie-Auswahl
Die Aktivität zur Auswahl der passenden Strategie für die Migration besitzt ihre größte Intensität zum Ende der Phase Vorstudie. Nachdem das Legacy-System global bewertet und die technischen Begebenheiten potenzieller Zielsysteme erfasst wurden, bestimmt die Migrationsstrategie die Grundlagen für das spätere Vorgehen.

Während zu Beginn der Phase Konzept & Design vor allem Aktivitäten zur Detaillierung des Legacy-Systems durchgeführt werden, gibt die Anpassung der Migrationsstrategie anschließend die Vorgaben für das Design des Zielsystems und die sinnvolle Einteilung der Transformationen. Somit erhöht sich im Übergang von Legacy-Analyse zum Ziel-Design die Intensität für die Strategie-Auswahl.

Im Vorfeld der Phase Migration & Übergang erfolgt die genauere Spezifizierung der Transformation-, Umstellungs- und Übergangsstrategien. Zur Abstimmung dieser Tätigkeiten ergibt sich ein erhöhter Aufwand zur Anpassung der Migrationsstrategie zu Beginn der Transformation und der Übergabe. Danach bis zum Abschluss muss die Migrationsstrategie nicht mehr wesentlich weiter bearbeitet werden und diese Aktivität nimmt merkbar an Intensität ab.

Transformation
Die Transformation bestimmter Einheiten des Alt-Systems begann bereits innerhalb der Phase Konzept & Design. Hierbei wurde die Durchführbarkeit getestet und die Transformation auf die Anforderungen des Ziel-Designs abgestimmt. Der Aufwand dieser Aktivität innerhalb der Phase war vor allem deshalb so hoch, weil bei der Transformation nicht auf ein Werkzeug zurückgegriffen werden konnte. So mussten die Transformationsroutinen je nach Notwendigkeit weiterentwickelt und getestet werden.

Durch die beständige Bearbeitung und Verbesserung der Transformationsmechanismen in der vorhergehenden Phase, nimmt der Aufwand für die Transformation während der Phase „Migration & Übergang“ nur moderat zu. Bei der tatsächlichen Transformation des Alt-Systems kann auf die getesteten Routinen zurückgegriffen werden und es sind nur noch bestimmte manuelle Eingriffe im Zielsystem notwendig. Dazu gehören die in Abschnitt 6.2 genannten Aktivitäten.

Test
Das Testen des Zielsystems und der Transformationen beginnt bereits sehr früh, noch in der Phase der Vorstudie. Der Grund lag darin, dass man die technischen Möglichkeiten des Zielsystems möglichst früh bewerten wollte.

Auch in der nächsten Phase „Konzept & Design“ musste immer wieder überprüft werden, ob das Zielsystem tatsächlich den Anforderungen entspricht und wie es sich verhält. Da in dieser Phase die Transformationsregeln formalisiert und testweise Transformationen durchgeführt wurden, ergibt sich hier eine kontinuierliche und feststellbare Intensität.

Besonders erhöhte sich der Testaufwand naturgemäß zum Abschluss der Migration vor der Übergabe. Dabei unterschied sich der notwendige Arbeitsaufwand erheblich von dem zuvor und die Intensität stieg sprunghaft an. Mit der Phase „Abschluss“ enden die Testaktivitäten.

Übergabe
Die Aktivitäten für die Übergabe setzten bereits während der Phase „Konzept & Design“ ein. Innerhalb dieser Phase wurden vereinzelt Akzeptanztests bezüglich des Zielsystems durchgeführt und die Ergebnisse in das Design übernommen.

Verstärkt hat sich der Aufwand zum Ende der Phase „Migration & Übergang“ und „Abschluss“. Die Vorbereitung der Zielumgebung, das Integrieren des Zielsystems und die Fertigstellung der Dokumentation verursachten einen beständigen Anstieg der Intensität. Durch die Archivierung des Legacy-Systems reicht der Kernbereich Übergabe noch weit in die Phase Abschluss hinein.

Konfigurations- / Änderungsmanagement
Von Seiten der Legacy-Website ergaben sich im Verlauf der Migration keine Änderungen an der Struktur. Damit mussten diese nicht durch Konfigurations- und Änderungsmanagement nachvollzogen werden.

Der gesamte Aufwand zur Wahrung der Integrität der erstellten Migrationsartefakte beschränkte sich somit auf die Entwicklung des Zielsystems. Das Ziel-Design, das zu Beginn erstellt wurde, erwies sich als relativ stabil. Dennoch mussten zu verschiedenen Zeitpunkten der Migration nach Absprache mit den andern Projektbeteiligten Änderungen vorgenommen werden. Zu Beginn konnten die Änderungen noch mit geringem Aufwand durchgeführt werden, da das Design lediglich „auf Papier“ vorhanden war.

Während begonnen wurde, das Zielsystem in Plone zu implementieren, kam mit dem Klassendiagramm für die automatische Generierung von Content-Typen ein weiteres Dokument hinzu, dass aktualisiert werden musste. Sobald Änderungen notwendig wurden, musste das Datenmodell in der Dokumentation und für die Implementierung in Plone überarbeitet werden.

Besonders schwerwiegend wurden einzelne Änderungen des Designs, nachdem die Transformationsregeln formuliert waren. Da die Transformationsregeln auf dem Datenmodell für die Erzeugung von Content-Typen in Plone aufbauten, mussten auch diese angepasst werden. Der Aufwand für die Aktualisierung stieg nochmals merklich. Bis zum Abschluss der Migration blieb somit der Aufwand für das Konfigurations- und Änderungsmanagement hoch.

Bemerkenswert ist, dass durch die Wahl einer „Big Bang“-Umstellungsstrategie das Aktualisierungsproblem nicht noch um eine weitere Dimension eskalierte. Wären die Migrationspakete bereits komplett transformiert gewesen, wenn Unzulänglichkeiten des Modells erkannt worden sind, hätte neben der Dokumentation, dem Klassendiagramm und den Transformationsregeln auch das bereits überführte Migrationspaket überarbeitet oder aufgegeben werden müssen.

Projektmanagement
Im Rahmen der Durchführung von Projektmanagement wurden zu Beginn der Migration innerhalb der Vorstudie diverse Aktivitäten ausgeführt. Insbesondere umfassten sie die Abschätzung des Projektsumfangs mit Erstellung eines Zeitplans und die Planung des Projekts in Abstimmung mit der gewählten Migrationsstrategie. Somit wurden Aktivitäten innerhalb dieses Kernbereichs während der Vorstudie mit einer erhöhten Intensität ausgeführt.

Im Verlauf der Migration wurde der Fortschritt des Projekts kontrolliert. Dies geschah in regelmäßigen Treffen zur Besprechung der bisher bearbeiteten Teilbereiche. Somit ergab sich vor jedem Treffen ein erhöhter Aufwand für das Projektmanagement, durch die Erstellung von Berichten für das Treffen als auch durch das Treffen selbst. Daraus resultieren die in Abständen erhöhten Aufwände für das Projektmanagement bei den Phasen „Konzept & Design“ und „Migration & Übergang“.

Bei der Übergabe des Zielsystems und dem Abschluss des Migrationsprojekts erhöhte sich der Aufwand des Projektmanagements nochmals. Dies umfasste abschließende Treffen der Projektbeteiligten und das endgültige Verfassen dieser Arbeit zur Bildung des Projektabschlusses.

Mitarbeiterqualifizierung
Der Bedarf der Mitarbeiterqualifizierung war zu Beginn der Migration vor allem auf Seiten des Programmierers hoch. Zur Durchführung der Legacy-Analyse, für das Erlernen des Umgangs mit dem Zielsystem und mit den für die Migration notwendigen Werkzeuge musste zunächst dieser Basisbereich ausreichend vorbereitet werden. Daher verursachte die Mitarbeiterqualifizierung zu Beginn einen relativ hohen Arbeitsaufwand.

In der Phase „Konzept & Design“ ging die Intensität der Mitarbeiterqualifizierung zwar tendenziell zurück, aber es ergab sich immer wieder weiterer Qualifizierungsbedarf. So gibt es an einzelnen Stellen einen Anstieg des Aufwands. Besonderes gravierend war dies im Übergang zur Phase „Migration & Übergang“, da die Programmierkenntnisse für die Transformationsregeln zum großen Teil neu erlernt werden mussten.

Um zu gewährleisten, dass das Wartungsteam nach Durchführung der Migration weiterhin Anpassungen am Zielsystem implementieren kann, mussten Weiterbildungsmaßnahmen auf dem Zielsystem durchgeführt werden.

Migrationsumgebung
Die Auswahl und Einrichtung der Migrationsumgebung wurde bereits im Verlauf der Vorstudie eingeleitet. So fand z.B. für die Durchführung der notwendigen Analysen für das Legacy-System das Werkzeug ArgoUML zur Erstellung von Klassendiagrammen Verwendung. Dennoch war der Einsatz von migrationsspezifischen Werkzeugen an dieser Stelle eher gering.

Die Nutzung von Migrationswerkzeugen verstärkte sich zur Mitte der Phase „Konzept & Design“. An dieser Stelle musste das für die Programmierung in Plone verwendete Code-Generierungswerkzeug ArchGenXML eingerichtet werden, wobei Probleme bei der Modellierung von Klassendiagrammen für ArchGenXML und mit dem Tool selbst auftraten. Dabei erhöhte sich der Aufwand signifikant.

Im Verlauf der Migration gab es punktuell Erweiterungen bei der Auswahl der Migrationswerkzeuge, wie z.B. die Auswahl eines Werkzeugs zur automatischen Erzeugung von Screenshots von Webseiten. Der damit verbundene Aufwand war allerdings eher gering.

14 Ausblick

Die vorliegende Arbeit beschäftigte sich mit der Anwendung des Referenz-Prozessmodells ReMiP für die Migration der Website des GXL-Projekts nach Plone. Damit konnte die Funktionsweise dieses Modells nachvollzogen und verifiziert werden. Das Ziel zur Überprüfung der Machbarkeit des ReMiP wurde somit erreicht.

Das damit eng verbundene Ziel der Migration der GXL-Website konnte ebenso erfolgreich abgeschlossen werden. Die neue GXL-Website wurde installiert und kann an Hand der mitegelieferten CD nachvollzogen werden.

Als Resultat der Untersuchung wurde auf Grund der Erkenntnisse im Migrationsverlauf eine Erweiterung des Referenz-Prozessmodells vorgeschlagen. Diese Erweiterung stellt das Prozessmodell nicht in Frage, sondern schlägt eine logische Restrukturierung vor, die dem Ablauf der Migration gerecht werden kann.

Zudem konnten durch die konkrete Durchführung einer Migration Erfahrungen mit der Intensität der einzelnen Migrationsbereiche gemacht werden. Die dabei gewonnenen Werte sind zwar nur als Anhaltspunkte zu verstehen, zeichnen aber den Arbeitsverlauf während der Migration gut nach.

Die abgeschlossene Migration beschreibt noch nicht alle Aspekte des ReMiP. Durch Besonderheiten des Migrationsprojekts wurden einzelne Aktivitäten des ReMiP nur angedeutet oder sogar gänzlich ausgelassen. Die Durchführung weiterer Projekte mit vollständigeren Charakteristiken kann zusätzlich wertvolle Daten liefern.

Das dargestellte Migrationsprojekt war bezüglich seines Umfangs beschränkt. Durch seine Abdeckung der zu berücksichtigenden Artefakte erwies der ReMiP sich für ein kleineres Migrationsprojekt als sehr umfangreich. Gerade im Umfeld von größeren Projekten sollte er seine Vorteile ausschöpfen können. Insofern ist auch dessen Anwendung für solche Projekte ein wichtiges Untersuchungsgebiet.

Literaturverzeichnis

Die referenzierten Internet-Ressourcen wurden am 22.12.2006 auf Gültigkeit überprüft.

Abts / Mülder (2004)
Abts, D. / Mülder, W., ”Grundkurs Wirtschaftsinformatik. Eine kompakte und praxisorientierte Einführung“, Vieweg, 2004

Ackermann (2005)
Ackermann, E., ”Ein Referenz-Prozessmodell zur Software-Migration“, Diplomarbeit, Universität Koblenz-Landau, 2005

AGAS (2003)
AGAS, ”AGAS Webpräsenz-Projektpraktikum“, Dokumentation, Sommersemester 2003, Oktober 2003
URL: http://www.uni-koblenz.de/FB4/Dokumentation/AGASDokumentation.pdf

Aune (2005)
Aune, Nate, ”Building an Artist Community Website“, Plone Conference Vienna Austria, September 21, 2005
URL: http://www.jazkarta.com/presentations/artist-community.pdf

Baumann (2003)
Baumann, S., ”Ganzheitliche Unterstützung der Wertschöpfungskette in Medien-Unternehmen mit Content Management Lösungen“, in "Content Management Handbuch - Strategien, Theorien und Systeme für erfolgreiches Content Management" von Stahl, F. / Maass, W. [Hrsg.], S. 77-88, 2003

Boiko (2002)
Boiko, B., ”Content management bible“, Hungry Minds, 2002

Gersdorf (2003)
Gersdorf, R., ”Content Management für die flexible Informationswiederverwendung“, in "Content Management Handbuch - Strategien, Theorien und Systeme für erfolgreiches Content Management" von Stahl, F. / Maass, W. [Hrsg.], S. 59-75, 2003

Gipp (2006)
Gipp, T., ”Functional Web Site Specification“, Logos, 2006

Herold (2003)
Herold, H., “make – Das Profi-Tool zur automatischen Generierung von Programmen“, Addison-Wesley, 2003

Kappel et al. (2004)
Kappel, G., Pröll, B., Reich, S., Retschitzegger, W. [Hrsg.], ”Web Engineering“, dpunkt.verlag, 2004

Limi (2005)
Limi, A., ”Archetypes Developers Guide“, 2005
URL:
http://plone.org/products/archetypes/documentation/old/ArchetypesDeveloperGuide/index_html#schema

Masak (2005)
Masak, D., “Legacysoftware. Das lange Leben der Alt-Systeme“, Springer, 2005

McKay / da Silva (2006)
McKay, A. / Silva, Sidnei da, ”The Definitive Guide to Plone“, First Edition, enfoldsystems, 12th December, 2006
URL: http://plone.org/documentation/manual/definitive-guide/definitive_guide_to_plone.pdf

Kommentar: Dieses Buch bezieht sich auf Plone 2.0 und wird von den Verfassern als veraltet beschrieben. In Einschränkung und in den Fällen, wofür es in dieser Arbeit herangezogen wird, behält es dennoch seine Gültigkeit.

Meinel / Sack (2004)
Meinel, C. / Sack, H., ”WWW - Kommunikation, Internetworking, Web-Technologien“, Springer Verlag, 2004

Ritz (2005)
Ritz, R., ”Programming Plone - The MySite Tutorial“, (c) ITB, Humboldt-University Berlin, 2004, 2005 (preview release as of May 2, 2005), 2005
URL: http://docs.neuroinf.de/programming-plone/mysite.pdf

Sneed (1999)
Sneed, Harry M., ”Objektorientierte Softwaremigration“, Addison-Wesley, 1999

Thome / Böhn (2006)
Thome, R. / Böhn, M., ”Enterprise Content Management“, Wisu, 4/06, S. 541-550, 2006

Wöhr (2004)
Wöhr, H., ”Web-Technologien - Konzepte, Programmiermodelle, Architekturen“, dpunkt.verlag, 2004

Zschau (2003)
Zschau, O., ”Web Content Managment Systeme - Eine Einführung“, in "Content Management Handbuch - Strategien, Theorien und Systeme für erfolgreiches Content Management" von Stahl, F. / Maass, W. [Hrsg.], S. 51 - 57, 2003

Anhang

Erläuterungen zur mitgelieferten CD

Zur Ergänzung des textuellen Teils der Abschlussarbeit wird eine CD mitgeliefert. Sie enthält folgende Verzeichnisse:

- *GXL-Website_Legacy_offline*: innerhalb dieses Ordners befindet sich eine Spiegelung der Legacy-Website von GXL. Am besten kann sie durch die Verknüpfung *GXL-Website_Legacy* angesteuert werden.
- *GXL-Website_migriert_offline*: dort ist eine Spiegelung der migrierten GXL-Website gespeichert. Da zum Zeitpunkt der Abgabe der Arbeit die endgültige Internet-Adresse noch nicht festgelegt war, wurde die Testinstallation gespiegelt. Die Homepage lässt sich am besten durch die Verknüpfung *GXL-Website_migriert* aufrufen.
- *Produkt-Code GXL_Website*: in diesem Ordner ist das Produkt-Verzeichnis von GXL_Website abgelegt. Hier kann die Programmierung der Content-Typen und vor allem die der Transformationen eingesehen werden. Die Transformationsskripte befinden sich im Ordner *scripts*.
- *Screenshots*: hier befinden sich die Screenshots, die für die Testgruppe zur Lauffähigkeit der GXL-Website auf Browsern gemacht wurden.
- *Sonstiges*: hier befinden sich die für diese Arbeit verwendeten, frei verfügbaren Programme sowie die ZARGO-Datei *GXL_Website*, die zur Erstellung des gleichnamigen Produkts für Plone mit ArgoUML und ArchGenXML verwendet wurde.

Webseiten zur Bearbeitung von Content der Ziel-Website

Das Rahmenwerk Archetypes für Content-Typen in Plone bietet neben der im Abschnitt 4.3.2 beschriebenen Standardansicht auf Objekte auch eine Ansicht zu deren Bearbeitung und Verwaltung. Diese Ansicht wird von Archetypes automatisch bei jedem Aufruf des *edit*-Templates für ein Objekt erstellt. Die Bearbeitung und Verwaltung von Content sind die Kernaufgaben des Redakteurs und des für die Veröffentlichung von Inhalten zuständigen Programmierers. In diesem Abschnitt wird auf die von Plone und Archetypes zur Verfügung gestellte Funktionalität eingegangen.

Verwaltung von Content
Im Content Management System Plone wird Inhalt in Form von Instanzen von Content-Typen eingefügt. Dabei kann es sich bspw. um eine Webseite des Content-Typs *Document* handeln. Diese Objekte werden innerhalb von anderen Objekten eingefügt, den *Containern*. Container sind ordnerartig. Wird eine Webseite auf der obersten Ebene einer Website, ihrer Wurzel, eingefügt, so fungiert diese als Container für den eingefügten Inhalt. So ergibt sich in der Website ein hierarchischer Aufbau von Objekten und Containern.

Wie in Abbildung 55 zu erkennen ist, stellt das Objekt *Plone-Website* die Wurzel der Zielwebsite dar. Auf der nächsten hierarchischen Ebene befinden sich die Container-Objekte *Conferences*, *Organisations*, *Persons* und *GXL* mit den eigentlichen Inhalten der GXL-Website.

In den Objekten *Conferences*, *Organisations* und *Persons* werden zentral die Daten der Konferenzen, Organisationen und ihrer Organisationseinheiten und der Personen gespeichert. Sollen Instanzen der Content-Typen *Conference*, *Organisation* / *OrganisaitonalUnit* oder *Person* gelöscht oder hinzugefügt werden, geschieht dies in diesen Containern. Dorthin wird von anderen Stellen der Webseite auf die Inhalte der Sammel-Content-Typen plone-intern

verlinkt. Der Container selbst soll dem Betrachter der Website nicht zugänglich sein, er dient lediglich zur Verwaltung der dort gespeicherten Instanzen.

Für die Verwaltung der Inhalte muss sich der Redakteur, bzw. der für die Veröffentlichung zuständige Programmierer, auf der Website als *Manager* einloggen. Nachdem er den gewünschten Container über die Website-Navigation erreicht hat, kann er verschiedene Aktionen durchführen (vgl. auch Abschnitt 1.5.4). Eine der wichtigsten Aktionen ist diejenige für das Hinzufügen von Content-Typ-Instanzen in den Container. Durch das Datenmodell wurde festgelegt, dass in die Container nur Inhalte von bestimmten Content-Typen eingefügt werden dürfen (vgl. Abschnitt 4.2.2).

Hinzufügen von Content
Für die Veranschaulichung der Verwaltung von Content wird exemplarisch der Container *Persons* ausgewählt. Durch die Datenmodellierung ist die Menge der hinzufügbaren Content-Typen zu *Persons* auf Instanzen des Typs *Person* beschränkt. Eine Instanz von Person kann durch das Anklicken der Aktion *add person* durchgeführt werden. Diese Aktion ist in Abbildung 84 schwarz eingerahmt und mit einem Pfeil gekennzeichnet.

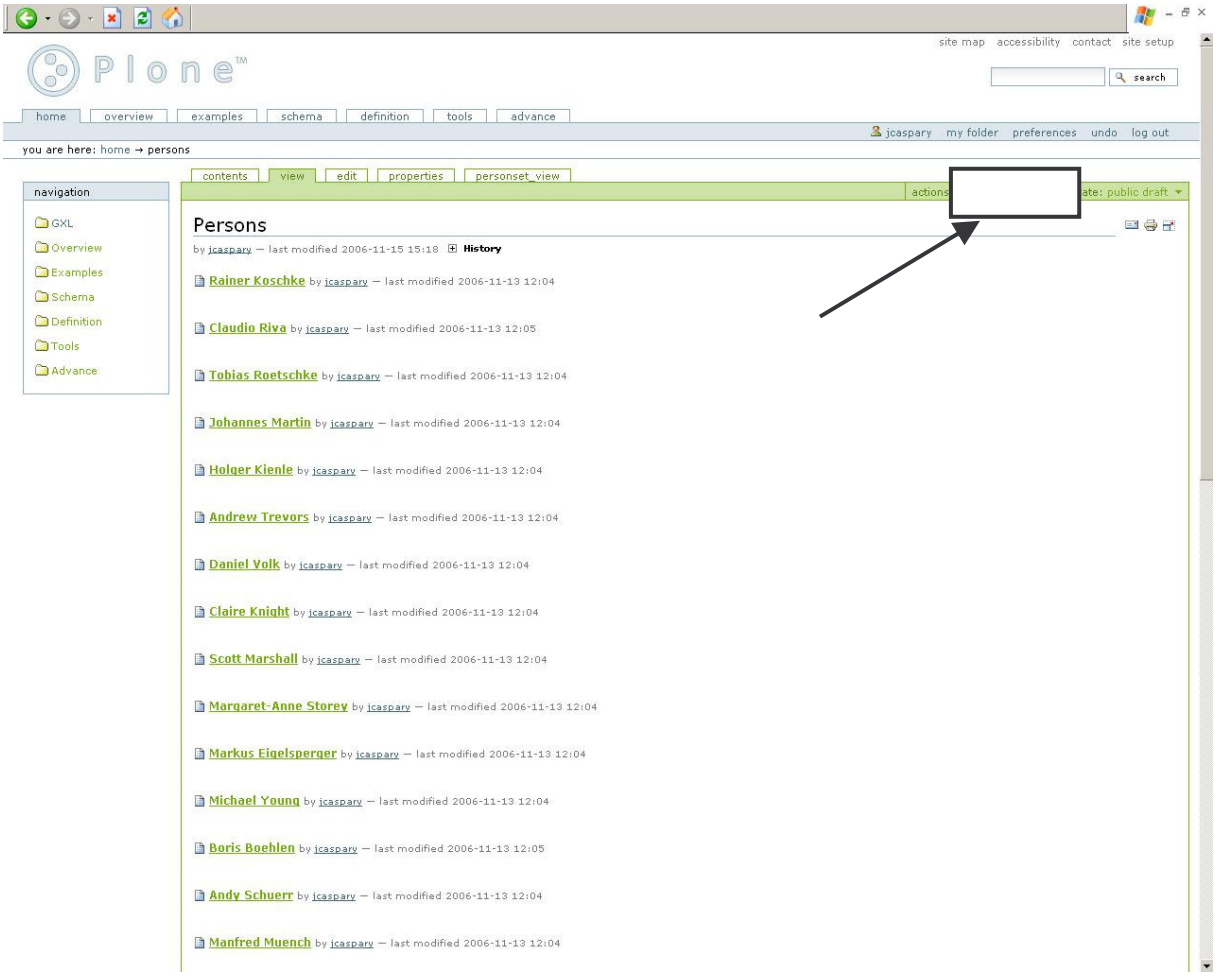

Abbildung 84: Ansicht des Containers *Person*

Nach Anklicken der Aktion *add person* fügt Plone eine Instanz von *Person* in den Container *Persons* hinzu und öffnet die Ansicht zur Bearbeitung der *Person*-Instanz. Bei dieser Bearbeitungsansicht handelt es sich um ein von Archetypes dynamisch generiertes Formular, das die Befüllung der Felder der Instanz des Content-Typs über die Web-Oberfläche ermöglicht. Abbildung 85 zeigt die Standardansicht zur Bearbeitung von *Person* (siehe auch Umrandung und Pfeil). Der Vorgang der Erzeugung einer neuen Instanz von Person kann sehr

gut in der Breadcrumbs-Leiste abgelesen werden (home → persons → portal_factory → person → […]). *Portal_factory* steht für die Konstruktor-Methode, in diesem Fall für den Konstruktor von *Person*.

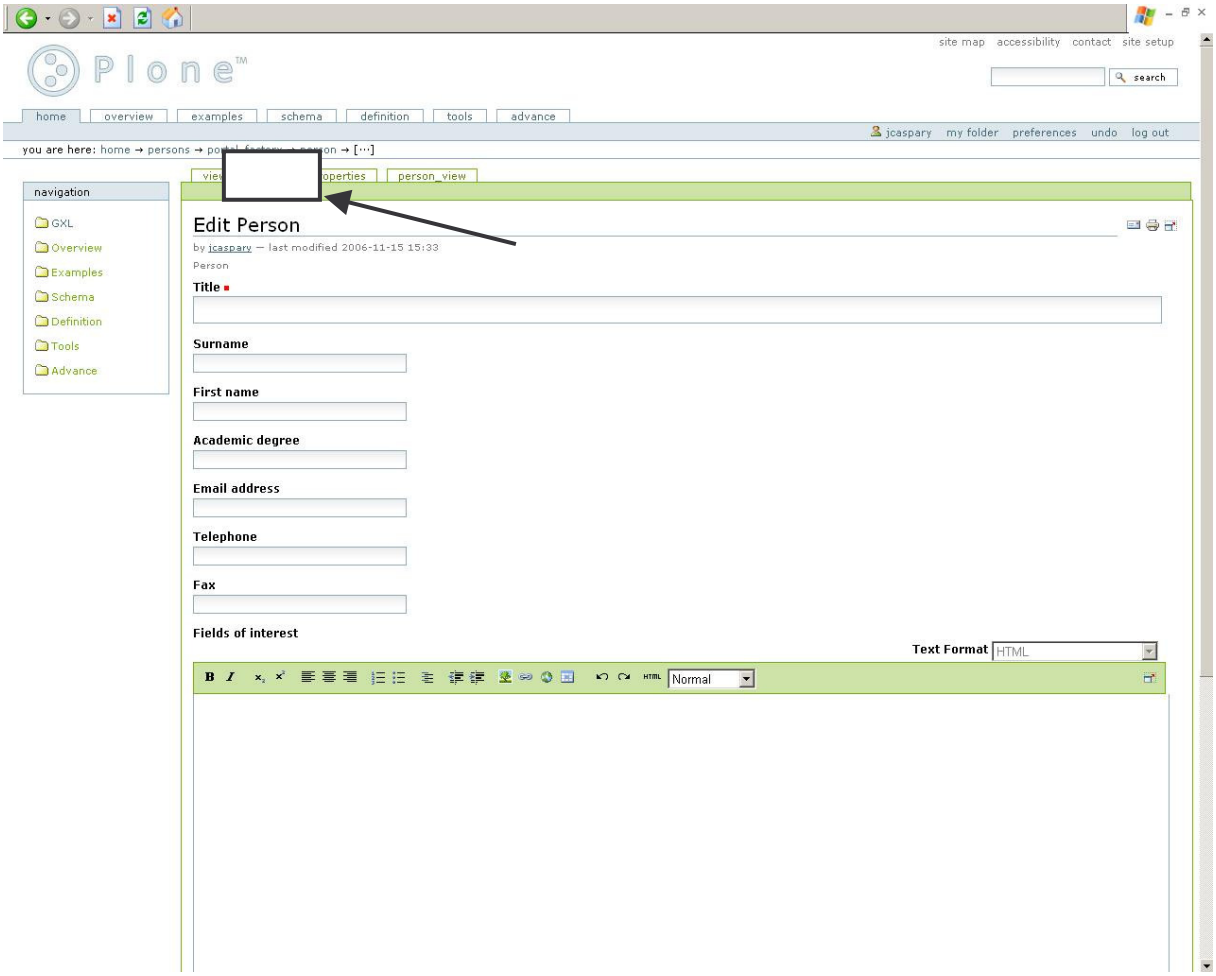

Abbildung 85: Bearbeitungsansicht des Content-Typs *Person*

Über dieses Formular werden nun alle Daten zur Person eingetragen. Felder mit einer roten Markierung (wie in Abbildung 85 bei *Title*) sind obligatorisch. Wird ein obligatorisches Feld nicht ausgefüllt oder erfüllt es nicht die Anforderungen des Validators (vgl. Abschnitt 1.5.5), so wird das Formular durch Plone erneut aufgerufen und die Fehlerstelle markiert. Zur Veranschaulichung dieses Mechanismus Im Abbildung 86 wurde das *Title*-Feld leer gelassen.

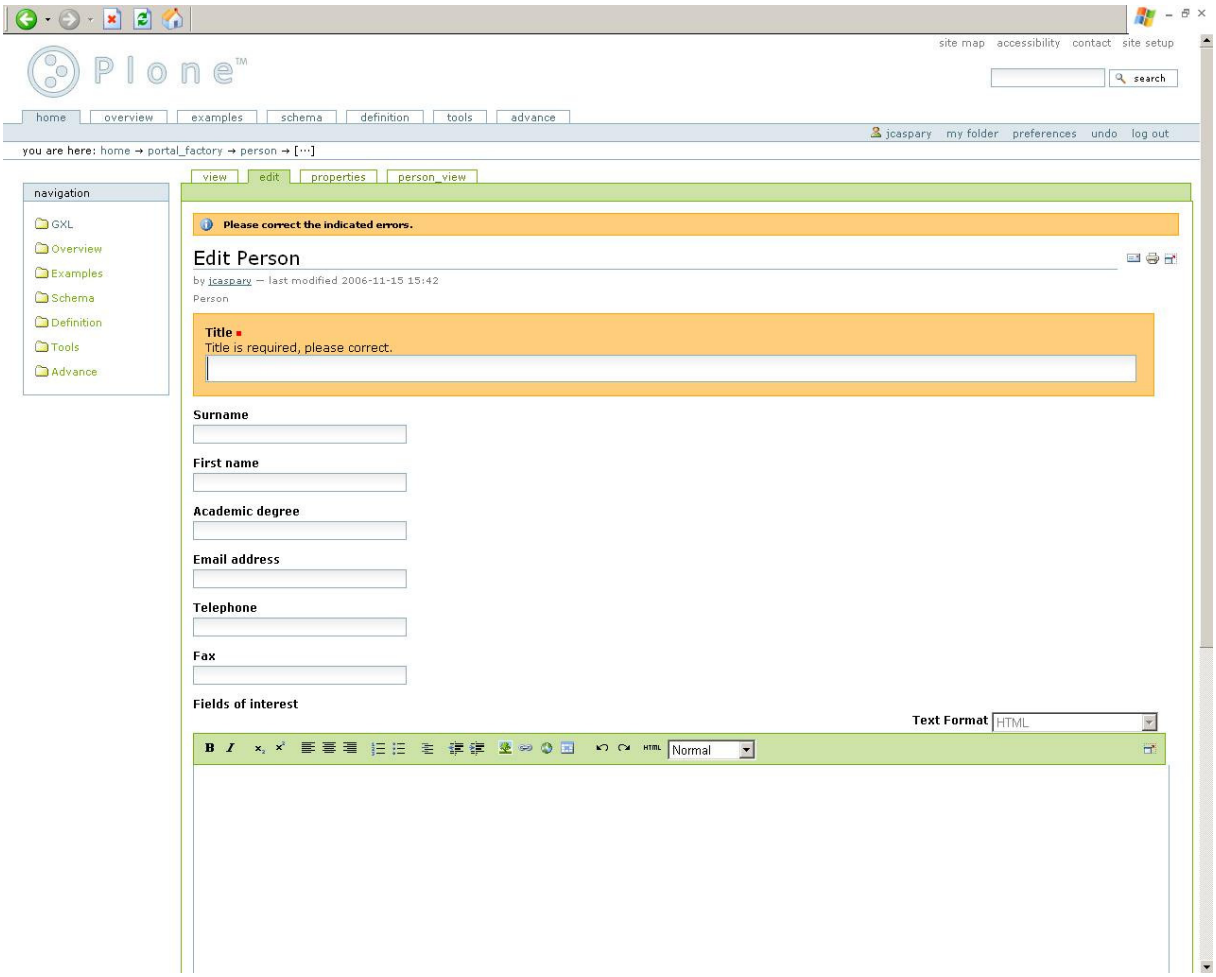

Abbildung 86: Bearbeitungsansicht mit Fehlermeldung

Wurden schließlich alle notwendigen Felder korrekt eingetragen, kann die Instanz des Content-Typs durch Anklicken des dafür vorgesehenen Buttons gespeichert werden. Zur Bestätigung der korrekten Verarbeitung des Contents, zeigt Plone die Standardansicht der Instanz an mit der orange hervorgehobenen Meldung, dass die Änderungen gespeichert wurden. Abbildung 87 zeigt beispielhaft das Ergebnis der Hinzufügung einer Person.

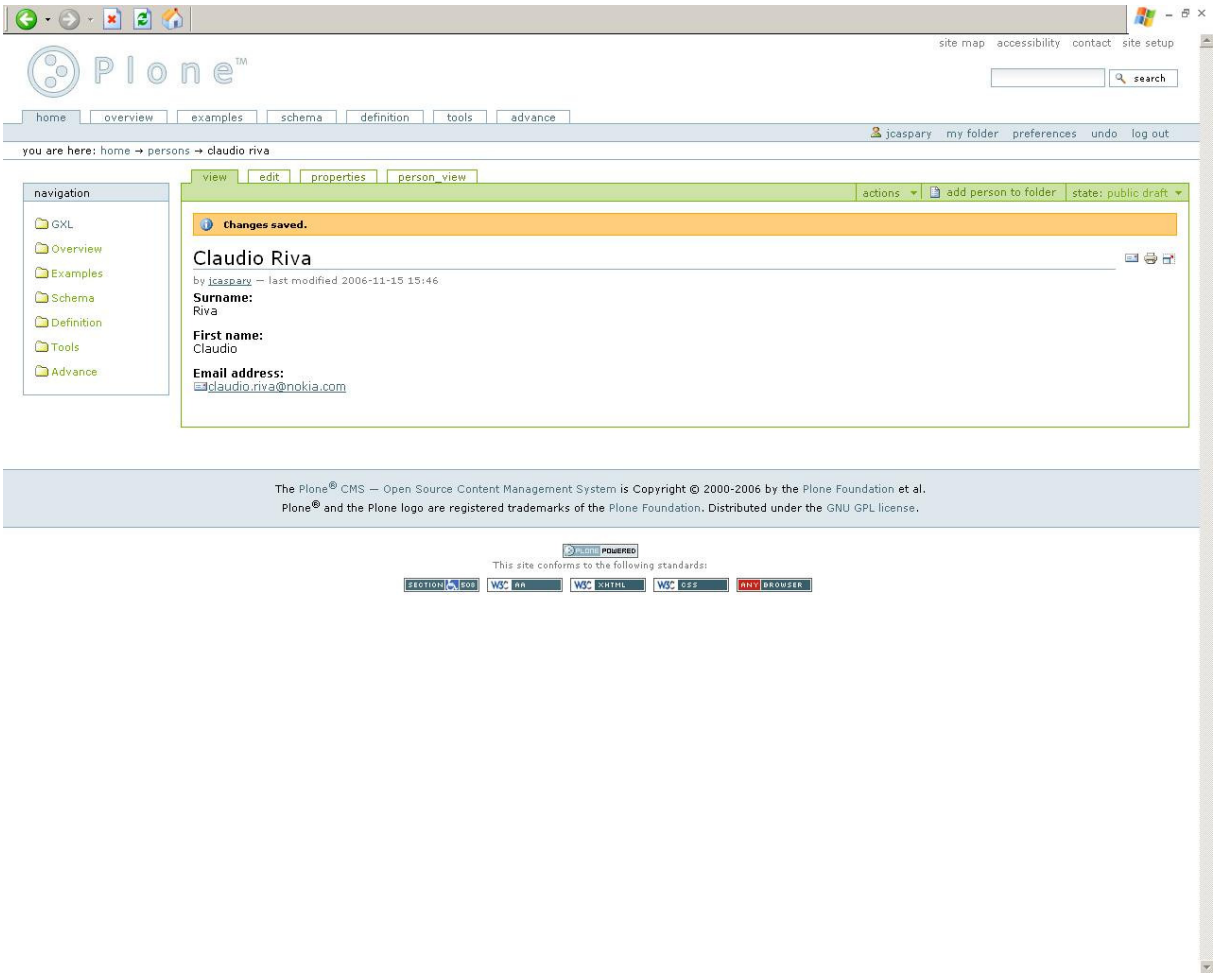

Abbildung 87: Rückmeldung durch Plone nach erfolgreicher Eintragung von Content

Analog zu dem beschriebenen Verfahren für die Container-Content-Typen von *Conference*, *Organisation* und *Person* kann auch beim Container *GXL* verfahren werden. Der einzige Unterschied bei *GXL* besteht darin, dass die Menge der hinzufügbaren Content-Objekte nicht so eingeschränkt ist.

Löschen von Content

Neben dem Hinzufügen von Content gehört auch die Möglichkeit, Content zu löschen, zu den wichtigen Operationen bei der Website-Verwaltung. In Plone gibt es hierfür zwei Varianten. Bei einer wird das Objekt, ausgehend vom Container, gelöscht und bei der anderen direkt über eine Aktion. Zunächst wird die Alternative mit dem Container beschrieben.

Das Beispiel mit dem Container *Persons* aus der Beschreibung des Vorgehens beim Hinzufügen von Content wird hier übernommen. Auch bei dieser Operation muss der Anwender als Manager eingeloggt sein. Neben der Standardansicht zur Anzeige eines ordnerartigen Objekts, existiert eine Ansicht zur Bearbeitung seiner Inhalte. Abbildung 88 zeigt diese und hebt die dabei ausgewählte Aktion hervor.

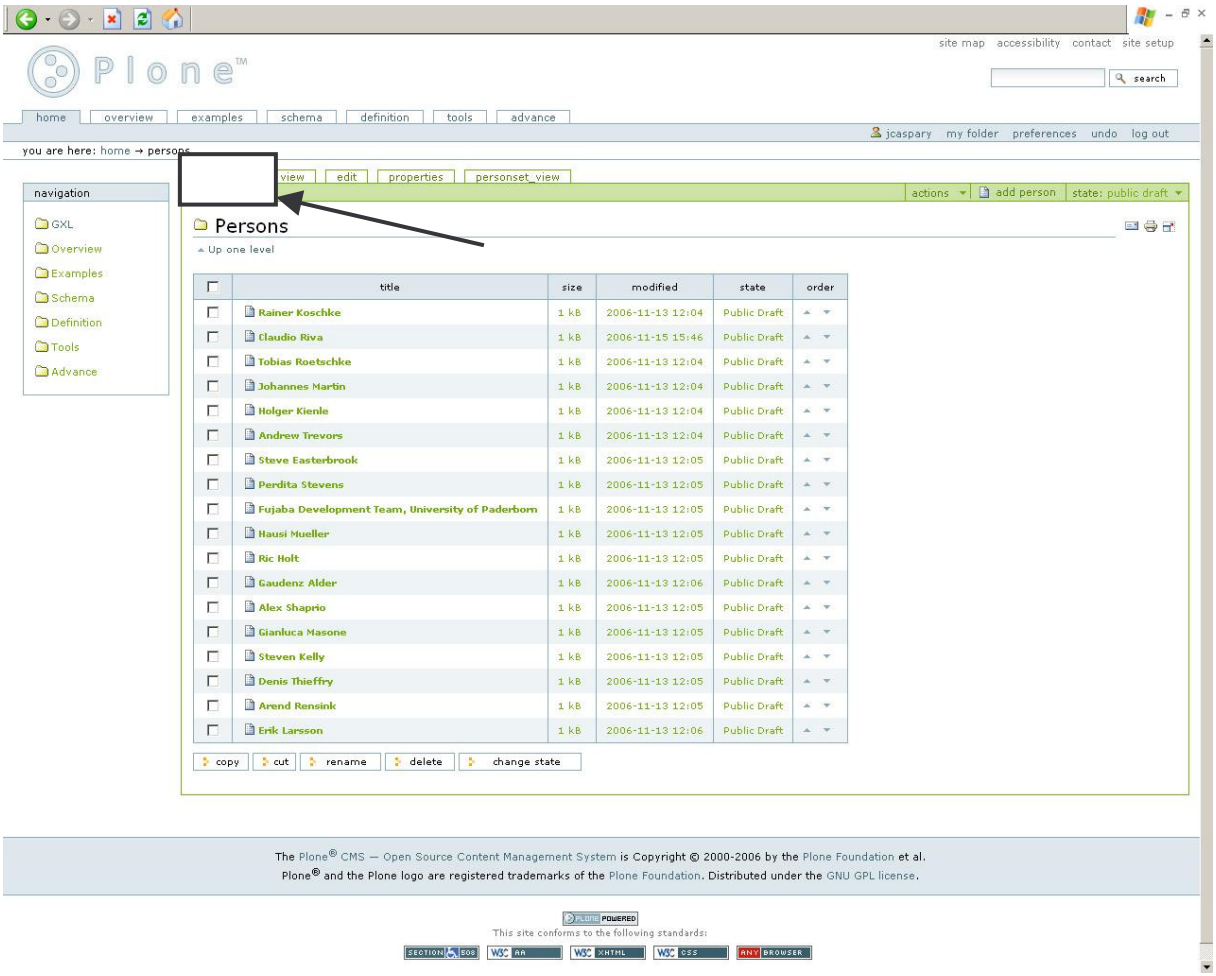

Abbildung 88: Inhaltsansicht für den Container *Persons*

Die in der Liste in Abbildung 88 befindlichen Instanzen des Content-Typs Person können alle durch Anklicken der Check-Boxen links ausgewählt werden. Anschließend kann für die ausgewählten Objekte eine der am Tabellenende befindlichen Aktionen ausgewählt werden. Von den Buttons *copy*, *cut*, *rename*, *delete* und *change state* muss zur Löschung des Contents *delete* ausgewählt werden.

Das beschriebene Vorgehen ist insbesondere für Fälle geeignet, bei denen mehrere Instanzen eines Content-Typs gelöscht werden sollen. Alternativ zur Löschung über die Ordnerliste kann jedes Objekt über eine Liste standardmäßiger Aktionen entfernt werden. Hierfür muss der Anwender als Manager eingeloggt sein und das betroffene Objekt angezeigt haben. Abbildung 89 zeigt am Beispiel einer Instanz des Content-Typs Person, die Liste von Aktionen, die an jedem Objekt in Plone ausgeführt werden können. Die Liste ist umrahmt, während die Aktion zum Löschen mit einem Pfeil markiert ist.

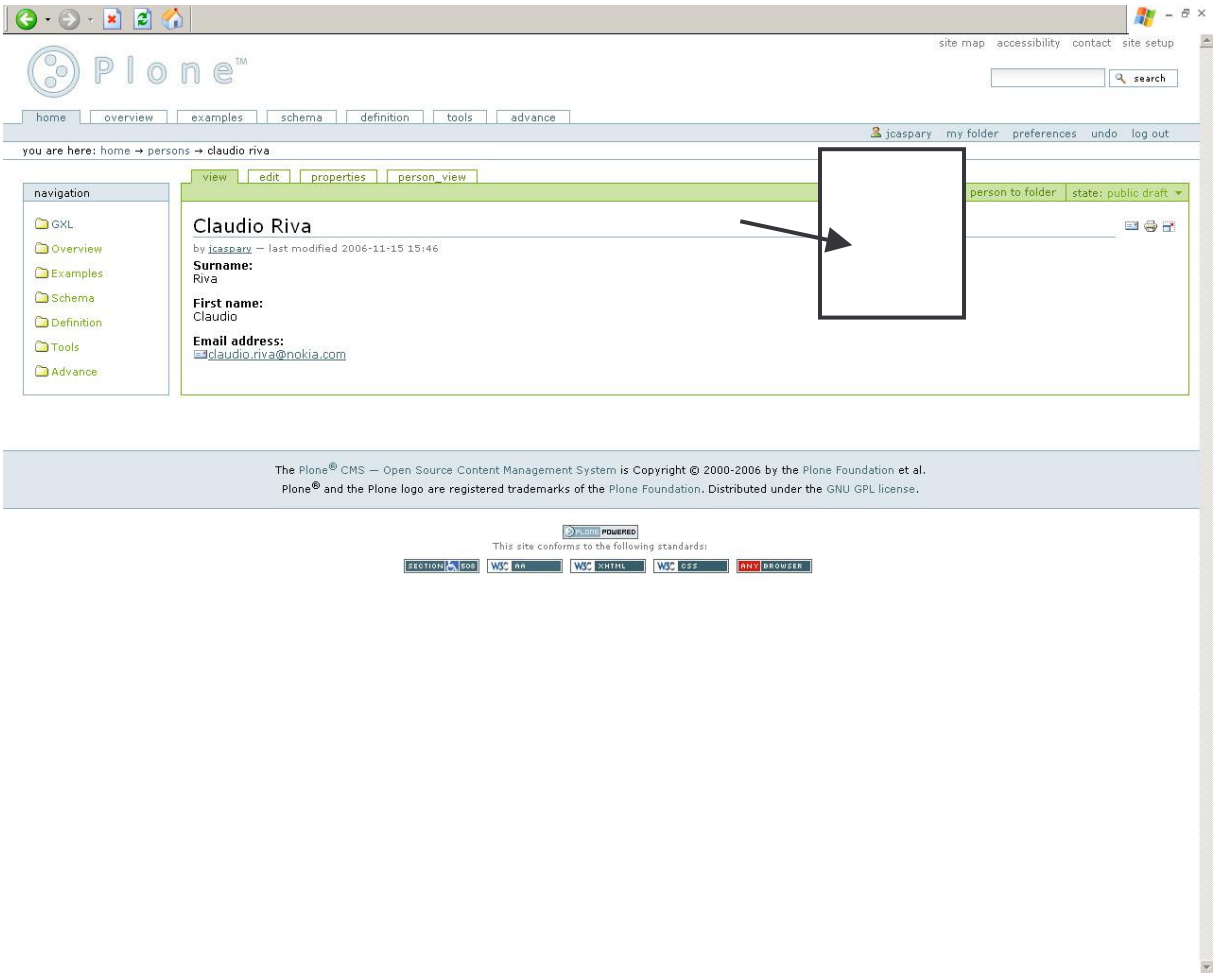

Abbildung 89: Aktion zum Löschen eines Objekts

Bearbeiten von Content

In der redaktionellen Tätigkeit wird Content regelmäßig bearbeitet und modifiziert. In Plone erfolgt die Bearbeitung der Felder der Content-Typen über das Bearbeitungsformular, wie es im Unterpunkt „Hinzufügen von Content“ beschrieben wurde. Je nach Auswahl des Widgets für das Content-Feld erscheint eine andere Eingabemaske für das Objekt. Besonders zu erwähnen ist der visuelle HTML-Editor Kupu zur Bearbeitung von Richtext-Feldern (siehe auch Abschnitt 3.4). Auf das Bearbeitungsformular wird an dieser Stelle nicht weiter eingegangen.

Neben dem Bearbeitungsformular existiert aber noch eine andere Ansicht zur Bearbeitung von Content. Wie in Abschnitt 1.5.3 dargestellt, verfügt Plone über eine Fülle von Metadaten, die den Content beschreiben. Diese Metadaten sind für Suchmaschinen und die Verwaltung des Contents innerhalb von Plone nützlich. Abbildung 90 zeigt die Ansicht zur Bearbeitung der Metadaten und umrahmt die dafür ausgewählte Aktion.

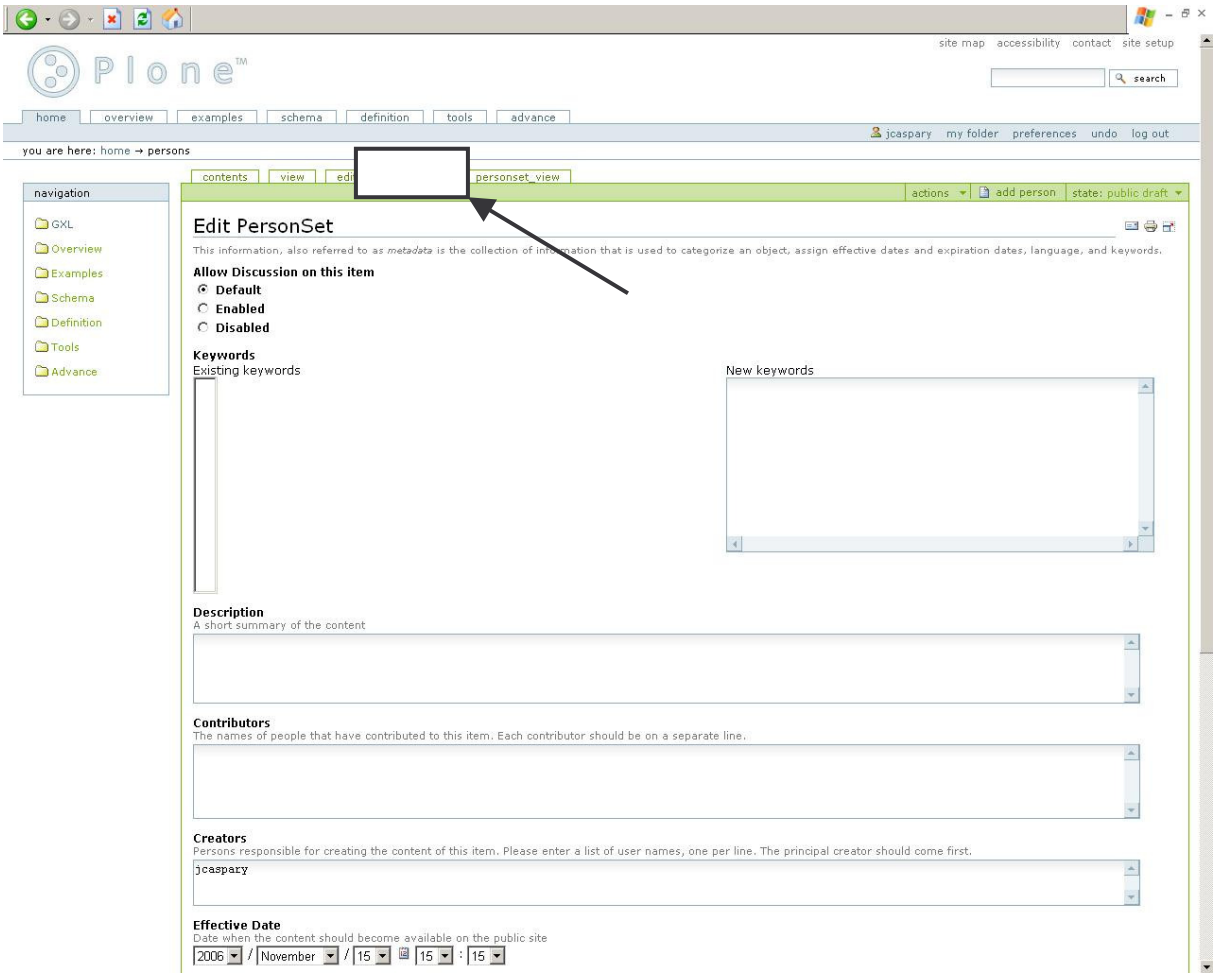

Abbildung 90: Ansicht zur Bearbeitung der Metadaten von Content in Plone

Das Formular zur Bearbeitung der Metadaten ist bei allen Content-Typen generell gleich. Es gibt Möglichkeiten auf Definitionsebene der Content-Typen, Metadatenfelder auf die Bearbeitungsansicht der Content-Typ-Felder (*edit*) zu übertragen. In diesen Fällen fehlt das Widget in dem Metadatenformular (*properties*).

Glossar

Anforderung: Vorgabe für Funktionalität und qualitative Eigenschaften des Zielsystems.

Client: Programm, das zu einem Server Kontakt aufnimmt und mit ihm kommuniziert.

CMF: Content Management Framework, auf Zope aufbauendes Rahmenwerk, das Content Management-Funktionalität bereitstellt.

Content Management System: Software zur Durchführung von Content Management.

Content Management: Verwaltung von Content, die u.a. die einfache Bearbeitung von Content und eine Benutzerverwaltung zur Verfügung stellt.

Content: Einheiten von Inhalt, die mit Metadaten versehen sind.

CSS: Cascading Style Sheets, Vorlagen für die Formatierung der Anzeige von HTML-Code.

GXL-Website: siehe Legacy-Website

HTML: Hypertext Markup Language, Dokumentenbeschreibungssprache für Webseiten.

HTTP: Hypertext Transfer Protocol, Internet-Protokoll für den Transport von Hypertext-Dokumenten wie Webseiten.

LitDB: Datenbank zur Verwaltung von Literaturdaten von Veröffentlichungen der Universität Koblenz-Landau und weiterführender Literatur. Zu finden unter *https://www.uni-koblenz.de/~litdb/secure/index.php*.

Literaturdatenbank: siehe LitDB

Literaturdatenbank des Fachbereichs 4 der Universität Koblenz-Landau: siehe LitDB

Legacy-Website: allgemein beschreibt es eine veraltete Webseite; hier ist es die GXL-Website, die nach Plone zu migrieren ist. Zum Zeitpunkt der Erstellung dieser Arbeit ist sie unter der Adresse *http://www.gupro.de/GXL/* zu finden.

Metadaten: Daten, die Content näher beschreiben, ohne dabei selbst Content zu sein.

Navigationsbereich: Bereich innerhalb der Navigationsleiste der Legacy-Website, der eine thematische Einordnung der Navigationspunkte ermöglicht.

Navigationspunkt: Innerhalb eines Navigationsbereich eingebundener Link auf eine Webseite der Legacy-Website.

Server: Programm, das von einem Client aufgerufen wird und mit diesem kommuniziert.

Web-Applikation: Internet-Anwendung, die von mehreren Benutzern gleichzeitig unter Nutzung eines Browsers bedient werden kann.

Web-Applikationsserver: Server mit Web-Applikationenfunktionalität

Webseite: eine einzelne Seite, die in HTML geschrieben und über das Internet erreichbar ist.

Website: der Bereich, der mehrere Webseiten logisch zusammenfasst.

Website-Betrachter: der Nutzer einer Website

Website-Betreiber: der Verwalter der zu migrierenden GXL-Website

Website-Designer: der Designer für die Website

Website-Programmierer: der Programmierer für die Funktionalität der Website

Website-Verantwortlicher: hier der für die Website der Universität Koblenz-Landau Hauptverantwortliche

XMI: XML Metadata Interchange, XML-Format.

XML: Extensible Markup Language, Dokumentenbeschreibungssprache, die die eigene Definition von Dokumentenstrukturen erlaubt.

Zielsystem: System, in das die Legacy-Website migriert wird. Als Zielsystem wurde hier Plone ausgewählt.

ZMI: Zope Management Interface, Web-basierte Oberfläche zur Bearbeitung des Web-Applikationsservers Zope.

Zope: Z Object Publishing Environment, in Python geschriebener, objektorientierter Web-Applikationsserver

Zu migrierende Website: siehe Legacy-Website