H. Neuroth, A. Oßwald, R. Scheffel, S. Strathmann, K. Huth (Hrsg.)

nestor Handbuch

Eine kleine Enzyklopädie der digitalen Langzeitarchivierung

Version 2.3

Kapitel 13

Tools

nestor

nestor Handbuch: Eine kleine Enzyklopädie der digitalen Langzeitarchivierung
hg. v. H. Neuroth, A. Oßwald, R. Scheffel, S. Strathmann, K. Huth
im Rahmen des Projektes: nestor – Kompetenznetzwerk Langzeitarchivierung und Langzeitverfügbarkeit digitaler Ressourcen für Deutschland
nestor – Network of Expertise in Long-Term Storage of Digital Resources
http://www.langzeitarchivierung.de/

Kontakt: editors@langzeitarchivierung.de
c/o Niedersächsische Staats- und Universitätsbibliothek Göttingen,
Dr. Heike Neuroth, Forschung und Entwicklung, Papendiek 14, 37073 Göttingen

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet unter http://www.d-nb.de/ abrufbar.

Neben der Online Version 2.3 ist eine Printversion 2.0 beim Verlag Werner Hülsbusch, Boizenburg erschienen.

URL für Kapitel 13 „Tools“ (Version 2.3): urn:nbn:de:0008-20100624128
http://nbn-resolving.de/urn/resolver.pl?urn:nbn:de:0008-20100624128

Gewidmet der Erinnerung an Hans Liegmann (†), der als Mitinitiator und früherer Herausgeber des Handbuchs ganz wesentlich an dessen Entstehung beteiligt war.

13 Tools

13.1 Einführung

Stefan Strathmann

Die Langzeitarchivierung digitaler Objekte ist eine überwältigend große Herausforderung.

Viele Gedächtnisinstitutionen verfügen über umfangreiche digitale Bestände, die sie auch künftig für Ihre Nutzer bereitstellen möchten. Es liegt auf der Hand, dass die vielen Arbeitsschritte, die durchgeführt werden müssen um eine sichere und langfristige Bereitstellung zu gewährleisten, möglichst nicht manuell erledigt werden sollten. Die digitale Langzeitarchivierung ist dringend auf automatisierte oder zumindest technik-gestützte Abläufe angewiesen.

Schon bei der Planung der digitalen LZA können computerbasierte Werkzeuge die Aufgaben erheblich erleichtern. Die dann später auf diese Planungen aufbauende Praxis der LZA ist ohne automatisierte Abläufe und entsprechende Werkzeuge kaum vorstellbar. Beispielsweise ist die dringend notwendige Erhe-

bung technischer Metadaten ein Prozess, der sich hervorragend zur Automatisierung eignet.

Mit dem Etablieren einer Praxis der digitalen LZA entstehen auch zunehmend mehr Werkzeuge, die genutzt werden können, um die anfallenden Aufgaben automatisiert zu bewältigen. Diese Werkzeuge sind häufig noch in den frühen Stufen der Entwicklung und speziell an die Bedürfnisse der entwickelnden Institution angepaßt. Sie werden aber zumeist zur Nutzung an die LZA-Community weitergegeben und entwickeln sich mit beeindruckender Geschwindigkeit weiter.

Das Kapitel 13 Tools stellt einige der vorhandenen Werkzeuge vor bzw. erläutert deren Benutzung. Insbesondere werden Werkzeuge zur Metadatenextraktion, zum Erstellen von Archivpaketen und zur Planung von LZA-Aktivitäten vorgestellt.

Die Herausgeber wünschen sich, dass dieses Kapitel in den folgenden Neuausgaben des nestor Handbuches deutlich erweitert werden kann.

13.2 Plato

Hannes Kulovits, Christoph Becker, Carmen Heister, Andreas Rauber

Die Planung digitaler Langzeitarchivierungsmaßnahmen und deren Dokumentation, wie im OAIS Referenzmodell vorgesehen, sowie von der Zertifizierungsinitiativen TRAC und nestor vorgeschrieben, stellen einen relativ komplexen und aufwändigen Prozess dar. Um diesen Ablauf schrittweise zu automatisieren, sowie um Unterstützung beim Durchlaufen der einzelnen Planungsschritte zu bieten, wurde Plato, das Planning Tool entwickelt, welches als Web-Applikation frei verfügbar ist. Plato führt den Anwender durch die einzelnen Schritte des Workflows zur Erstellung eines Langzeitarchivierungsplanes („Preservation Planning"), dokumentiert die Planungskriterien und Entscheidungen, und ermittelt teilautomatisiert die optimale Lösung für die jeweiligen spezifischen Anforderungen einer Institution. In diesem Kapitel wird ein detaillierter Überblick über Plato sowie seine Bedienung gegeben, und vor allem auch auf die bereits integrierten Services verwiesen, welche helfen, den Planungsablauf zu automatisieren.

Einführung

Plato[1] (Planning Tool) ist ein Planungstool, welches im Zuge des EU-Projekt PLANETS[2] entwickelt wurde. Das PLANETS Projekt arbeitet an einer verteilten, serviceorientierten Architektur mit anwendbaren Services und Tools für die digitale Langzeitarchivierung.[3] Das Planungstool implementiert den Planets Workflow zur Planung von Langzeitarchivierung.[4] Es können damit solide Entscheidungen für die Auswahl einer Planungsstrategie getroffen werden, die zu einer optimalen Planung von Langzeitarchivierung der betreffenden digitalen Objekte führt. Wie in Kapitel 12.4 ausführlich beschrieben besteht der PLANETS Preservation Planning Workflow im Kern aus drei Phasen: Die Definition des Planungskontextes (Archivierungsumgebung, Archivierungsgut) sowie der Anforderungen, die Auswahl und Evaluierung potentieller Maßnahmen („actions") anhand gewählter Beispielobjekte, sowie die Analyse der daraus resultierenden Ergebnisse. All diese Schritte werden mit Hilfe der Web-Applikation Plato unterstützt, um einzelne Prozess-Schritte zu automatisieren, sowie

1 http://www.ifs.tuwien.ac.at/dp/plato (12.02.2010)
Alle hier aufgeführten URLs wurden im Mai 2010 auf Erreichbarkeit geprüft.
2 http://www.planets-project.eu/ (12.02.2010)
3 Farquhar, 2007
4 Strodl 2007, Becker 2009

um eine automatische Dokumentation jeden Schrittes sicherzustellen.[5] In Plato ist es außerdem möglich einen Aktionsplan („Preservation Action Plan“) zu erstellen, der auf die in der dritten Phase erhaltenen empirischen Ergebnisse aufbaut und einen aufführbaren Workflow zur Durchführung der Langzeitarchivierungsschritte beinhaltet.

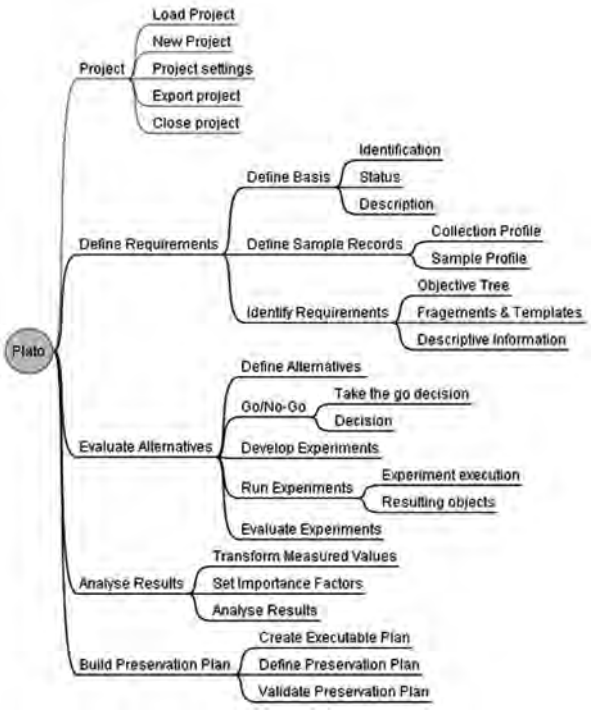

Abbildung 1: Aufbau von Plato

Das Ergebnis des Planungsdurchlaufs mit Plato ist ein Preservation Plan, der für eine konkrete Gruppe von digitalen Objekten die optimale Langzeitarchivierungsmaßnahme (samt Begründung für deren Auswahl) dokumentiert und entsprechende Anleitungen zur Durchführung der Maßnahme sowie deren erneute Evaluierung definiert. Dieser Plan wird in einer Registry abgespeichert, kann aber auch lokal als XML und PDF Dokument abgelegt und somit ebenfalls in ein Langzeitarchiv übernommen werden.

Plato ist über den link http://www.ifs.tuwien.ac.at/dp/plato als Web-Applikation frei verfügbar. Auf der Startseite wird eingangs informiert, was Plato ist

5 Becker, 2008a

und welche Neuerungen in der Entwicklung von Plato hinzugekommen sind. Ein Register weißt zudem auf weiterführende Literatur *(„Documentation")*, Fallstudien *(„Case Studies")* und Veranstaltungen *(„Events")* hin, auf denen Plato vorgestellt und präsentiert wird und wurde. Auf der „Documentation"-Webseite wird eine Liste einführender Literatur zu Plato und dem Planungsworkflow angeboten. Außerdem werden alle wissenschaftlichen Publikationen, die zu Plato veröffentlicht wurden, sowie die Projektberichte zur Verfügung gestellt. Auf der „Case Studies"-Webseite kann Einblick in fertig gestellte Beispielpläne genommen werden. Unter anderem sind hier Case Studies zur Erhaltung von Video Spielen, Interaktive Multimediale Kunst und elektronische Diplomarbeiten und Dissertationen zu finden. Diese können als hilfreiche Vorlage für einen eigenen Preservation Plan dienen. Bei der Entwicklung von Plato wurde besonders auf eine benutzerfreundliche Bedienung im Web-Interface geachtet, die auf allen gängigen Browsern immer wieder ausführlich getestet wird.

Die Schritte in Plato

Um einen eigenen Preservation Plan in Plato zu erstellen, muss sich der Anwender als erstes ein Konto *(„Account")* anlegen. Nach erfolgreicher Anmeldung öffnet sich eine Seite, die vorab die Möglichkeit bietet, einen existierenden Plan aus einer angebotenen Liste zu öffnen, einen neuen Plan zu kreieren, einen „Demo-Plan" zu erstellen oder aber einen schon existierenden Plan in Plato zu importieren. Der Demo-Plan dient zum Testen von Plato. Es kann hierbei durch einen fertig gestellten Plan beliebig durchgeklickt und auch verändert werden. Abbildung 1 bietet einen Überblick über die gesamte Menüstruktur von Plato und die einzelnen Phasen des Planungsprozesses, die in den folgenden Abschnitten detailliert erläutert werden.

Um einen neuen Plan zu erstellen, muss als erstes der Bestand, für den er erstellt werden soll, definiert werden. Üblicherweise handelt es sich dabei um eine mehr oder weniger konsistente Sammlung von digitalen Objekten, die mit Hilfe einer bestimmten Langzeitarchivierungsmaßnahme (z.B. einem bestimmten Migrationstool) behandelt werden sollen, da sie konsistente technische (z.B. Dateiformat, Struktur, Metadaten) und oft auch konzeptionelle Eigenschaften (Verwendungszweck, Zielgruppe) aufweisen. Zudem sollten die Risiken für die Langzeitarchivierung im Vorhinein bekannt sein, welchen mit Hilfe des Preservation Plans begegnet werden soll. Die aufklappbare Navigationsleiste im oberen Bereich des Bildschirms gibt im ersten Menüpunkt die Möglichkeit das Planungsvorhaben zu verwalten. Die weiteren Menüpunkte stehen für die einzelnen Phasen der Planungsworkflows. Der Übersichtlichkeit halber wurden die

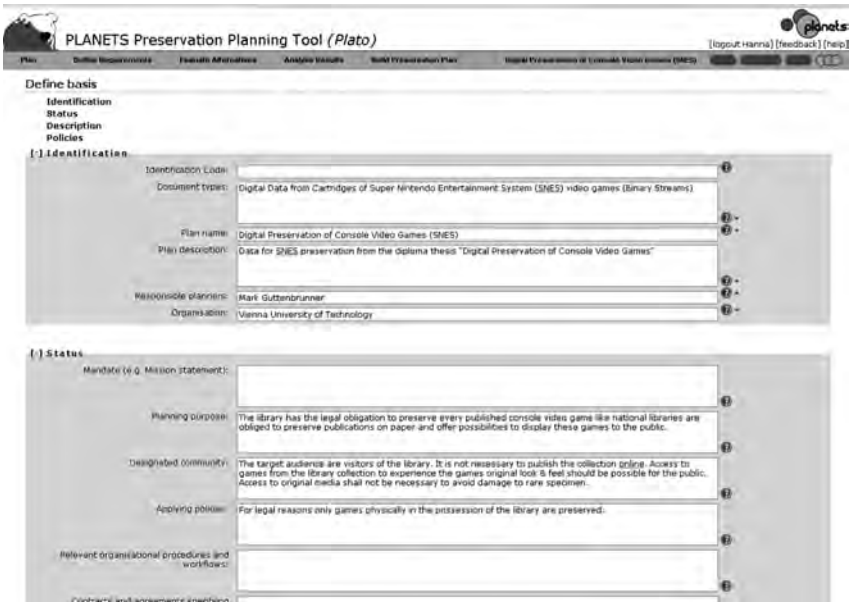

Abbildung 2: Phase 1/ Schritt 1 in Plato

Begrifflichkeiten des Planungsworkflow in der Navigationsleiste übernommen. Auf der rechten Seite der Navigationsleiste zeigt ein Verlaufsanzeiger in Form von gefüllten Kreisen den Status des Planes. Wurde mit den Planungsphasen angefangen, kann leicht durch die einzelnen fertiggestellten Schritte navigiert werden. Es sollte jedoch bei Änderungen in vorhergehenden Phasen darauf geachtet werden, dass diese gespeichert werden. Wird eine Änderung in einer vorhergehenden Phase oder in einem vorhergehenden Schritt einer Phase durchgeführt und diese dann auch gespeichert, wird der Status des Planungsprozesses bis zu dieser Änderung zurückgesetzt, da sich die Voraussetzungen bis zu diesem Status verändert haben und so die nachfolgenden Schritte dementsprechend angepasst werden müssen. Dies bedeutet aber nicht, dass alle nachfolgenden Informationen automatisch gelöscht werden. Damit Änderungen am Preservation Plan jederzeit nachvollzogen werden können, protokolliert Plato intern die letzte Änderung mit. Diese wird mit Datum und dem Benutzernamen der Person, die die Änderung vorgenommen hat, gespeichert und kann im Analyseschritt eingesehen werden.

Phase 1: Definition der Anforderungen

Im zweiten Menüpunkt der Navigationsleiste wird die erste Phase des Planungsworkflows „Festlegen der Anforderungen" („Define requirements") in einzelne Untermenüpunkte, die den einzelnen Schritten innerhalb der Phasen entsprechen, aufgeschlüsselt. In „Define requirements" sollen im ersten Schritt („Define basis") Informationen und Daten zum Planungsvorhaben sowie zum Planungskontext dokumentiert werden (Abbildung 2). Dies beinhaltet zum einen die Dokumentation über den Plan selber („Identifikation"), z.B. wer der Planungsbeauftragte ist und um welche Dokumentenarten es sich handelt. Zum anderen sollen der Status, angewendete Rahmenbedingungen („Policies"), die Zielgruppe und das Mandat (z.B. gesetzliche Verpflichtungen) erfasst werden. Außerdem werden hier die Auslöser („Trigger"), deretwegen dieser Plan erstellt wird, vermerkt. Hierzu ist eine Reihe von Auslösern vordefiniert, wie z.B. die Behandlung eines neuen Bestandes, ein geändertes Langzeitarchivierungsrisiko für ein bestehendes Dateiformat, neue Anforderungen von Seiten der Anwender, etc.

Abbildung 3: Anforderungsbaum (a) in Plato und (b) als Mindmap

Im zweiten Schritt werden repräsentative Beispielobjekte vom Anwender ausgewählt und in Plato hochgeladen und gespeichert. Hier werden konkret einzelne Objekte aus dem Bestand (oder aus einer Sammlung von Referenzobjekten) ausgewählt, anhand derer die jeweiligen Tools zur Langzeitarchivierung getestet werden sollen. Bei der Auswahl sollte darauf geachtet werden, dass man das Spektrum der technischen und intellektuellen Eigenschaften der Objekte innerhalb des Bestandes erfasst, also z.B. sowohl ein sehr kleines als auch ein sehr großes Objekt auswählt, eines mit Makros, Bildern, mit bestimmten Formatierungen, etc. Anschließend muss beschrieben werden, um welche Objekte es sich dabei handelt – sowohl intellektuell als auch technisch. Für die Beschreibung

der technischen Eigenschaften bietet Plato automatische Unterstützung. Die Formate der Beispielobjekte werden durch den in Plato integrierten Identifizierungsservice DROID[6] automatisch identifiziert und mit Informationen zum PUID (Pronom Persistent Unique Identifier)[7], zum Namen des Formats, der Version sowie des MIME-Type (Multipurpose Internet Mail Extensions-Type)[8] im Plan gespeichert. Dazu werden die Dateien via Webservice an DROID geschickt, welches entsprechende technische Metadaten erhebt und zurückliefert, die daraufhin in den Preservation Plan in Plato übernommen werden. (An der Integration weiterer Analysewerkzeuge für detailliertere Beschreibungen wird derzeit gearbeitet.) Darüber hinaus soll die ursprüngliche technische Umgebung möglichst genau beschrieben werden (verwendete Software, Betriebssystem sowie Art der Verwendung).

Im nächsten Schritt („Identify Requirements“) lässt das Tool den Anwender die Anforderungen zur Planung der Langzeitarchivierung definieren. Dies ist einer der aufwändigsten Schritte in der Erstellung des Plans. Es sollte gewährleistet sein, dass möglichst die Sicht aller Stakeholder (Anwender, Techniker und Archivexperten) in diesem Schritt berücksichtigt wird. Deshalb bietet sich an, die Liste der Anforderungen in einem Workshop zu erstellen. Meist wird in diesen Workshops mit Post-it Notes oder mit Mind-Mapping Software gearbeitet und die Liste in Form eines Baumes strukturiert, um die einzelnen Anforderungen nach inhaltlichen Gesichtspunkten zu strukturieren (Abbildung 3 (b)). Mindmaps, die in der frei verfügbaren Software FreeMind[9] oder in Mindmeister[10] erstellt wurden, können in Plato importiert angezeigt werden. Ferner können die Kriterienbäume natürlich auch innerhalb von Plato mit Hilfe des Web Interface editiert werden (Abbildung 3 (a)). Plato bietet außerdem eine Bibliothek mit Vorlagen in „*Show the template library*“, die unterteilt ist in „Öffentliche Vorlagen“ *(„Public Templates*“), „Eigene Vorlagen“ *(„My Templates*“), „Öffentliche Fragmente“ *(„Public Fragements“)* und „Eigene Fragmente“ *(„My Fragements“)*. Diese enthalten vordefinierte Zweige von Kriterien, die in verschiedenen Standardszenarien immer wieder auftauchen und daher nicht jedes Mal von neuem manuell definiert werden müssen, sondern einfach aus der Bibliothek übernommen werden können. Plato beinhaltet öffentlich verfügbare Templates beispielsweise für die Langzeitarchivierung von Diplomarbeiten und Dis-

6 http://droid.sourceforge.net

7 http://www.nationalarchives.gov.uk/aboutapps/pronom/puid.htm

8 http://www.iana.org/assignments/media-types/

9 http://freemind.sourceforge.net/wiki/index.php/Main_Page

10 http://www.mindmeister.com/ Webversion eines Mindmapping Tools, welches Mindmaps als FreeMind File importiert und exportiert.

sertationen und die Langzeitarchivierung von Internetseiten. Erarbeitet wurden diese Templates aus verschiedenen umfangreichen Case Studies und beinhalten detaillierte Anforderungen an die Langzeitarchivierung der jeweiligen Sammlung. Die Vorlage für die Langzeitarchivierung von Internetseiten enthält genaue Anforderungen an das Aussehen, den Inhalt, die Struktur und das Verhalten von Webseiten. Wird ein Template als Anforderungsbaum übernommen kann dieser problemlos angepasst werden – Teilbäume, die nicht zutreffen, gelöscht und weitere Anforderungen eingefügt werden. Es existiert ebenfalls eine allgemeine Vorlage, die der vorgeschlagenen Grundstruktur eines Anforderungsbaumes entspricht: „Objekteigenschaften", „Technische Eigenschaften", „Infrastruktureigenschaften", „Prozesseigenschaften" und „Kontext". Diese kann herangezogen werden, wenn der Anforderungsbaum von Grund auf neu erstellt werden soll. Es gibt außerdem die Möglichkeit, die Bibliothek mit eigenen Fragmenten oder Vorlagen zu erweitern um sie so an wiederkehrende Anforderungen in der eigenen Institution anzupassen.

In letzter Konsequenz soll mit Hilfe von Plato objektiv ermittelt werden, wie gut einzelne Tools diese Kriterien erfüllen. Zu diesem Zweck muss jedem einzelnen Kriterium nach Möglichkeit ein objektiver Messwert zugewiesen werden. So kann zum Beispiel der Durchsatz bei Migrationstools in MB pro Sekunde gemessen werden; die Bewahrung der eingebetteten Metadaten in einem Objekt mit „Ja / Nein"; die Verfügbarkeit einer Dateiformatdefinition als „freier Standard", „Industriestandard", „proprietäres Format". Das Tool bietet eine Vielzahl von Messskalen („Boolean", „Ordinal", „Yes", „Acceptable", „No", „Integer", „Number" etc.) an, die unabhängig ausgewählt und genutzt werden können.

Am Ende der ersten Phase sind somit die Anforderungen an die optimale Lösung für den gesuchten Preservation Plan definiert, sowie Beispielobjekte ausgewählt, anhand derer einzelne Tools getestet werden sollen.

Phase 2: Evaluierung der Alternativen

In der zweiten Phase „Evaluierung der Anforderungen" *(„Evaluate alternatives")* kann der Anwender Langzeiterhaltungsmaßnahmen definieren, welche er überprüfen beziehungsweise testen will. Alternativen sind hierbei Tools *(„preservation action services")*, die den gewünschten Endzustand des Beispielobjektes erzeugen sollen. Dazu können Tools aus den verschiedensten digitalen Erhaltungsstrategien („Migration", „Emulation", „Beibehaltung des Status quo") (Kapitel 8) verglichen werden. Bei Textdateien kann beispielsweise Formatmigration oder die Beibehaltung des Status quo evaluiert werden. In anderen Fällen, beispiels-

weise bei Videospielen wird eher in Richtung Emulation der Systemumgebung evaluiert. Es können auch sämtliche Erhaltungsstrategien in einem Plan evaluiert werden.

Die Auffindung passender Alternativen kann sich je nach Bestand unterschiedlich aufwändig gestalten. Häufig müssen intensive Recherchephasen eingeplant werden, um herauszufinden, welche Tools überhaupt für die gewünschte Erhaltungsstrategie und den betreffenden Objekttyp derzeit verfügbar sind. Bei der Recherche nach geeigneten Tools können Service Registries helfen, die Tools für die Langzeitarchivierung zu listen. In Plato wurden deshalb *„Service Registries"* wie „CRIB"[11], „Planets Service Registry" oder „Planets Preservation Action Tool Registry" implementiert, welche die Suche nach geeigneten Tools automatisch durchführen und diese dem Anwender vorschlagen. Dazu wird in den Registries nach Tools gesucht, die auf den vorliegenden Beispieldateitypen operieren können. Je nach Art der Registry werden dabei auch komplexere Lösungen wie z.B.: Migrationspfade in mehreren Schritten (von TeX (Typesetting System) über DVI (Device Independent File Format) zu PDF (Portable File Document)) ermittelt. Der Anwender kann dann entscheiden, welche Vorschläge er übernehmen will. Will der Anwender Tools testen, die nicht von den Service Registries vorgeschlagen wurden, können diese manuell angegeben werden. Der Nachteil ist hierbei, dass diese Tools lokal installiert und gemessen werden müssen.

Im nächsten Schritt *„Go/No-Go"* gibt Plato erneut die Möglichkeit zu überlegen, welche der aufgelisteten Alternativen im Planungsprozess evaluiert werden sollen. In diesem Schritt sind Alternativen abwählbar, die beispielsweise interessant wären, aber in der Anschaffung zu kostspielig sind. Andererseits kann auch die Evaluierung eines bestimmten Tools vorerst aufgeschoben werden *(„Deferred go"),* samt Definition, wann bzw. unter welchen Bedingungen die Evaluierung nachgeholt werden soll, sofern z.B. ein bestimmtes Tool erst in naher Zukunft verfügbar sein wird. Die Gründe, die für oder gegen eine Alternative sprechen, können in Plato dokumentiert werden.

Bevor die Experimente zu den einzelnen Alternativen durchgeführt werden, muss der Anwender für jede einzelne Alternative die Konfiguration der Tools definieren und dokumentieren. Sind die Alternativen aus den Service Registries entnommen, erfolgt die Beschreibung automatisch. Sind die Szenarien für die einzelnen Experimente der einzelnen Alternativen und deren Rahmen (Personal, Tools etc.) vollständig und dokumentiert, können die Experimente im Schritt „Run Experiment" durchgeführt werden. Die Ausführung erfolgt

11 http://crib.dsi.uminho.pt/

wieder automatisch, sofern der Anwender die Services von den in Plato angebotenen Service Registries genutzt hat. Hierbei werden die einzelnen Beispielobjekte an die Webservices geschickt, die diese je nach Erhaltungsstrategie migrieren oder in einem Emulator wie z.B. GRATE[12] (andere Emulatoren können manuell aufgerufen werden) emulieren. Teilweise werden Messungen (Zeit, etc.) automatisch erhoben und Logfiles wie auch Fehlermeldungen übernommen. Die entstandenen Ergebnis-Dateien im Falle eines Migrationsprozesses können heruntergeladen werden. Die Ergebnisdateien werden in Plato gespeichert und bilden – gemeinsam mit den ursprünglichen Beispielobjekten – Teil des Plans und der Dokumentation der Experimente, die es erlauben, die Evaluierung jederzeit zu einem späteren Zeitpunkt zu wiederholen bzw. alte Ergebnisse mit jenen neuer Tools zu vergleichen. Bei manuellen Experimenten muss der Anwender die Tools mit den Beispielobjekten selbst aufrufen und die Experimente selbst durchführen sowie die Ergebnisse hochladen, so dass auch diese in Plato gespeichert sind.

Im fünften Schritt („Evaluate Experiments“) der zweiten Phase werden die Experimente auf Basis der Kriterien der Anforderungsliste bzw. des Anforderungsbaum evaluiert. Es werden hierbei für jedes einzelne Ergebnis (also z.B.: für jedes Migrationsergebnis eines jeden Beispielobjekts mit jedem einzelnen Tool) alle Kriterien des Anforderungsbaumes auf Blattebene evaluiert, um die Ergebnisse der einzelnen Experimente empirisch für jede Alternative zu erheben. Auch hier kann mit Hilfe von automatischen Tools („Preservation characterization tools“) ein Teil der Arbeit automatisiert werden. Diese „Characterization Tools“ analysieren den Inhalt der Dateien und erstellen eine abstrakte Beschreibung, die es erlaubt, in vielen Bereichen die Unterschiede vor und nach der Migration zu erheben. Beispiele für solche Beschreibungssprachen sind JHOVE[13] oder XCDL[14]. (An Tools, die einen automatischen Vergleich von Emulationsergebnissen erlauben, wird derzeit gearbeitet). Werte, die nicht automatisch erhoben werden können (wie z.B. eine subjektive Beurteilung des Qualitätsverlustes bei Kompressionsverfahren in der Videomigration), müssen manuell ins System eingegeben werden.

Am Ende der zweiten Phase ist somit für jedes einzelne Beispielobjekt bekannt, wie gut jedes einzelne der Preservation Action Tools die im Kriterienbaum definierten Anforderungen erfüllt.

---

12 http://www.planets-project.eu/docs/reports/Planets_PA5-D7_GRATE.pdf

13 http://hul.harvard.edu/jhove/

14 Becker, 2008

Phase 3: Analyse der Ergebnisse

Die dritte Phase „Consider results“ zielt nun darauf ab, die Ergebnisse aus den Experimenten zu aggregieren, um die optimale Lösung auszuwählen. Um dies zu tun, muss der Erfüllungsgrad der einzelnen Anforderungen durch die verschiedenen Tools erfasst und verglichen werden. Nachdem die Maßzahlen allerdings in den unterschiedlichsten Einheiten erhoben wurden, müssen diese zuerst in eine einheitliche Skala, den sogenannten Nutzwert („Utility value“), transformiert werden. Dazu werden Transformationsskalen festgelegt, welche die aufgetretenen Messwerte jeweils auf einen einheitlichen Wertebereich (z.B. angelehnt an das Schulnotensystem zwischen null und fünf) festlegen. Der Wert „null“ steht für ein unakzeptables Ergebnis, welches, kommt er in einer Anforderung zu einer Alternative vor, dazu führt, dass diese Alternative ausgeschlossen wird. Andererseits bedeutet „fünf“ die bestmöglichste Erfüllung der Anforderung. Beispielsweise kann eine Bearbeitungszeit von 0-3 Millisekunden pro Objekt mit dem Wert „fünf“ belegt werden, 3-10ms mit „vier“, 10-50ms mit „drei“, 50-100ms mit „zwei“, 100-250ms mit „eins“, und jeder Wert über 250ms als unakzeptabler Wert mit „null“ definiert werden. „Ja/nein“ Messwerte können entweder auf „fünf/eins“ oder „fünf/null“ abgebildet werden, je nachdem, ob die Nichterfüllung einen Ausschließungsgrund darstellen würde oder nicht.

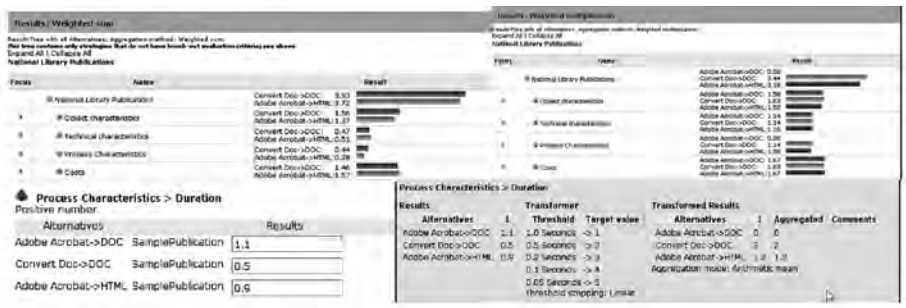

Abbildung 4: Evaluierungsergebnisse elektronischer Dokumente

Nachdem nun alle Messwerte in einheitliche Nutzwerte transformiert worden sind, kommt der optionale Schritt der Gewichtung. In der Grundeinstellung werden alle Kriterien innerhalb einer Ebene des Kriterienbaumes als gleich wichtig betrachtet. Sollte es nun der Fall sein, dass für das Planungskonsortium manche Kriterien von essentieller Bedeutung sind, während andere nur eine untergeordnete Rolle spielen, kann in diesem Schritt jedem Kriterium ein ei-

genes Gewicht relativ zu den anderen Kriterien gegeben werden. So kann z.B. der Erhalt des visuellen Erscheinungsbildes eine viel höhere Bedeutung haben als z.B. der Durchsatz, wenn die Anzahl der zu bearbeitenden Dateien nicht immens groß ist. In diesem Fall können die Prozessmesswerte geringer gewichtet werden, während die entsprechenden Kriterien die das Aussehen und den Erhalt der Funktionalität messen, höher gewichtet werden. In Plato kann dies mit Hilfe von Schiebereglern erfolgen, wo für jede Ebene des Baums die relative Gewichtung jedes einzelnen Knotens nach Wunsch angepasst und fixiert werden kann. Nicht veränderte Gewichte werden danach automatisch entsprechend angepasst.

Sind diese Schritte durchgeführt, kann Plato einen kumulierten Nutzwert für jede Alternative ausrechnen, d.h. wie gut jede einzelne Alternative die Gesamtheit aller Kriterien erfüllt. In der Folge können die Alternativen nach Gewichtung gereiht werden. Dazu stehen eine Reihe von Aggregierungsfunktionen zur Verfügung, von denen üblicherweise zwei im Kern relevant sind, nämlich die additive und die multiplikative Zusammenführung. Letztere zeichnet sich dadurch aus, dass ein Tool, das in einem einzigen Kriterium eine nicht akzeptable Leistung aufweist (also einmal den Nutzwert „null" zugewiesen bekam), auch im Gesamtranking mit „null" gewichtet wird und somit aus der Evaluierung ausscheidet. Hier kann so noch einmal gesondert die Beurteilung der einzelnen Messungen hinterfragt und angepasst werden.

Plato bietet dem Anwender dazu auch eine graphische Darstellung der Ergebnisse, damit die spezifischen Stärken und Schwächen jeder einzelnen potentiellen Maßnahme *„Preservation action"* vom Anwender auf Anhieb gesehen werden können. Abbildung 4 zeigt die Darstellung des Endergebnisses in Plato aus einem Planungsprozess zur Langzeitarchivierung wissenschaftlicher Arbeiten, die ursprünglich in PDF vorliegen, ähnlich dem Beispiel in (Becker 2007b) (Siehe dazu auch Abb. 4 im Kapitel 12.4, wo Ergebnisse einer ähnlichen Studie in tabellarischer Form zusammengefasst wurden.) Als Alternativen wird eine Reihe von Migrationstools evaluiert. Ferner wird zusätzlich die Null-Hypothese evaluiert, d.h. das Resultat unter der Annahme, dass man keine Langzeitarchivierungsmaßname setzt. In der Abbildung 4 sind die Nutzwerte unter Verwendung der beiden Aggregationsmethoden „Gewichtete Summe" und „Gewichtete Multiplikation", wie in Kapitel 12.4 beschrieben, dargestellt. Innerhalb von Plato kann man zwischen den beiden Aggregationsmethoden wechseln und der Ursache für die unterschiedlichen Rankings auf den Grund gehen. Indem der Baum expandiert wird, kann der Anwender erkennen, in welchen Kriterien die Leistung der einzelnen Tools mit „nicht akzeptabel" bewertet wurde.

Es wird außerdem erkennbar, dass die Alternativen „Migration in RTF (Rich Text Format) mit Adobe Acrobat“ und „Migration in TXT (Text File) mit Adobe Acrobat“ bei den Kriterien *„Apearance“* – „*Structure“* – „*Structure Tables“* und *„Conten“*„ – „*Figure Content*“ jeweils mit „Null“ bewertet wurden. Die „ConvertDoc Migration in RTF“ scheidet wiederum z.B. im Kriterium „*Technical Characteristics*“ – „*Tool“* – „*Makrosupport*“ aus. Die Null-Hypothese PDF *(„unchanged“)* scheidet bei der Aggregationsmethode Multiplikation aus, da das essentielle Kriterium der Verhinderung von eingebetteten Skripts „*Behaviour“* – *„Script blocking*“ nicht erfüllt wird. Durch Klicken auf das jeweilige Kriterium kann unmittelbar zu den entsprechenden Messwerten gesprungen werden. Hier können dann die Gründe für die unterschiedlichen Bewertungen nachvollzogen werden, sowie zu jedem späteren Zeitpunkt die entsprechend migrierten Dokumente geöffnet und deren Bewertung verglichen werden.

Am Ende der dritten Phase liegt nun eine Reihung der einzelnen alternativen „Preservation Action Tools“ vor, die es erlaubt, das am besten geeignete Tool auszuwählen sowie zu begründen, warum dieses Tool besser ist als die anderen. Darüber hinaus kann evaluiert werden, in welchen Bereichen es Schwächen aufweist, und so eine eventuelle Kombinationsstrategie empfohlen werden, d.h. es können unter Umständen zwei Tools kombiniert werden, von denen eines eher das Aussehen, das andere die interne Struktur und den Inhalt bewahrt, oder beispielsweise Elemente (z.B. Metadaten), die bei einer anderweitig hervorragenden Migration verloren gehen, durch separate Extraktion und Speicherung gerettet werden.

Phase 4: Aufbau des Durchführungsplans

Nachdem sich der Anwender am Ende der dritten Phase auf Basis der Analyseergebnisse für eine Alternative entschieden hat, erfolgt die Erstellung des Preservation Plans in der vierten Phase („Build Preservation Plan“). Diese umfasst nicht mehr den eigentlichen Planungsprozess, sondern die Vorbereitung der operativen Umsetzung eines Plans nach dessen Genehmigung. Sie wird hier daher nur verkürzt beschrieben. In dieser vierten Phase werden die notwendigen organisatorischen Maßnahmen definiert, die zur Integration der Erhaltungsmaßnahmen in die Organisation notwendig sind, dazu gehören ein detaillierter Arbeitsplan mit definierten Verantwortungen und Ressourcenzuteilungen zur Installation von notwendiger Hardware und Software. Zusätzlich werden Kosten und Überwachungskriterien für die Erhaltungsmaßnahmen definiert bzw. berechnet.

Im ersten Schritt der vierten Phase erstellt der Anwender einen Arbeitsplan inklusive der technischen Einstellungen wie den Speicherort der Daten, auf die die Maßnahme angewendet werden soll sowie die dafür notwendigen Parametereinstellungen für das Tool. Für die Qualitätssicherung werden Mechanismen geplant, welche die Qualität des Ergebnisses der Maßnahme überprüfen. Der zweite Schritt der Planerstellung beschäftigt sich mit den Kosten der getroffenen Erhaltungsmaßnahmen und der Überwachung des Planes. Die Kosten können entweder nach dem LIFE Kostenmodell[15] oder dem TCO Modell[16] (**T**otal **C**ost of **O**wnership Modell) aufgeschlüsselt werden. Um die laufende Aktualität des Planes sicherzustellen, werden Überwachungskriterien *(„Trigger conditions")* definiert, die festlegen, wann der Plan neu überprüft werden muss. Beispielsweise kann eine geänderte Objektsammlung eine Überprüfung erfordern um eventuell neu zutreffende Alternativen berücksichtigen zu können. Der letzte Schritt zeigt dann den vollständigen Preservation Plan mit empfohlenen Maßnahmen zur Erhaltung einer Sammlung von digitalen Objekten. Nachdem der Plan einer letzten Prüfung unterzogen wurde, wird er von einer berechtigten Person in Plato bewilligt und damit von diesem Zeitpunkt an als gültig festgelegt.

Der Preservation Plan

Im letzten Schritt der vierten Phase gibt Plato den gesamten Preservation Plan aus, welcher dann zur Archivierung als PDF exportiert werden kann. Der Preservation Plan enthält alle Informationen, die der Anwender eingegeben hat, sowie die Ergebnisse der Evaluierung der einzelnen Alternativen als Balkendiagramme. Die Evaluierungsergebnisse der Alternativen werden ebenfalls in einer Baumstruktur dargestellt, wodurch diese zu allen Anforderungen auf allen Ebenen angezeigt werden können. Der Preservation Plan ist wie folgt aufgebaut:

- Identifikation des Planes
- Beschreibung der organisatorischen Einrichtung
- Auflistung aller Anforderungen
- Beschreibung der Alternativen
- Aufbau der Experimente
- Evaluierung der Experimente
- Transformationstabellen
- Resultate (Summe und Multiplikation)

15 Shenton, 2007

16 http://amt.gartner.com/TCO/index.htm

```
<plan>
  <basis>
  </basis>
  <sampleRecords>
    <record shortName="Publikation"
        fullname="publ.pdf"
        contentType="application/pdf">
      <data>JVBERi0xLjQNJeLjz9MNCjc2IDAgb2
            IDw8L0xpbmVhcml6ZWQgMS9MIDQ4Nzg
            yNC9PIDc5L0Ug...
      </data>
      <formatInfo puid="fmt/18"
          name="Portable Document Format"
          version="1.4"
          mimeType="application/pdf"
          defaultExtension="pdf">
    </record>
    <record>
      ...
    </record>
  </sampleRecords>
  <alternatives>
    <alternative discarded="false"
        name="Adobe Acrobat to DOC">
      <description>...</description>
      <experiment>
        <description>...</description>
        <runDescription></runDescription>
        <uploads>
          <upload fullname="publ.doc"
            contentType="application/msword">
            <data>elxydGYxXGFuc2lcYW5zaWNwZzEy
                  NTJcdWMxXGRlZmYIHtcZm9udHRib
                  HtcZjBcZnN3aXNzXGZj...
            </data>
          </upload>
        </uploads>
      </experiment>
    </alternative>
    <alternative>...</alternative>
  </alternatives>
  <decision>...</decision>
  <tree>
    ...
    <node>
      <leaf name="Encoding" weight="0.4">
        <ordinalScale unit=""/>
        <ordinalTransformer>
          <mappings>
            <mapping ordinal="Original"
              target="5.0"/>
            <mapping ordinal="Changed"
              target="3.0"/>
            <mapping ordinal="None"
              target="1.0"/>
        </ordinalTransformer>
        <evaluation>
          <alternative>...</alternative>
          <alternative>...</alternative>
        </evaluation>
    ...
    </node>
```

⇒ Auflistung der Beispielobjekte auf Basis derer die Experimente durchgeführt wurden. Dieser Block enthält die gesamte Datei inklusive Metainformation und Inhalt (uuencoded).

⇒

Auflistung der gewählten Alternativen und den dazugehörigen Experimenten.

Die Experimente enthalten neben der genauen Beschreibung auch die Ergebnisdateien (Resultate) als upload-Block. Ebenfalls genau beschrieben mit Metadaten und Inhalt (uuencoded).

⇒

Der <tree> Block stellt den umfangreichsten im XML Dokument dar. Er enthält:

- Die einzelnen Knoten und Blätter (Anforderungen)
- Die gewählte Skala der Anforderung
- Die Gewichtungen der einzelnen Knoten und Blätter
- Die Transformationstabellen
- Die Evaluierung pro Alternative

- Entscheidung für eine Strategie
- Kosten
- Überwachung
- Bewilligung

Plato unterstützt auch den Export des Preservation Plans als XML Datei, welche einem definierten, öffentlich verfügbaren Schema entspricht. Diese Datei enthält alle Daten um den Preservation Plan auf einem anderen System reproduzieren zu können. Neben der Basisinformation, die der Benutzer während der Planerstellung eingegeben hat, sind auch alle Beispielobjekte, auf Basis derer die Evaluierung erfolgte, in die XML Datei eingebettet. Ebenfalls enthalten sind Metadaten über diese Beispielobjekte (z.B. Pronom Unique Identifier), die detaillierten Transformationstabellen, Evaluierungsergebnisse der einzelnen Experimente und die Ergebnisse. Die XML Datei ist wie folgt aufgebaut:

Zusammenfassung

Um einen Preservation Plan in Plato zu erstellen bedarf es viel Erfahrung. In diesem Kapitel wurde das Planungstool Plato vorgestellt, das Institutionen bei der Erstellung von Langzeitarchivierungsplänen unterstützt, die optimal auf ihre Bedürfnisse zugeschnitten sind. Plato implementiert den Planungsprozess, wie er in Kapitel 12.4 vorgestellt wird. Neben der automatischen Dokumentation aller Planungsschritte sowie der durchgeführten Experimente unterstützt es den Prozess vor allem durch die Integration von Services, welche Schritte wie die Beschreibung der ausgewählten Objekte, das Auffinden geeigneter Tools oder die Durchführung und Analyse der Ergebnisse von Langzeitarchivierungsmaßnahmen automatisieren. Durch den Zwang zu exakten Definitionen zu den zu bewahrenden Eigenschaften *(„Significant properties")* (und damit auch automatisch jener Aspekte, die vernachlässigt werden können bzw. verloren gehen dürfen) sowie der Anforderungen an den Langzeitarchivierungsprozess selbst bietet die Erstellung des Kriterienbaumes *(„Objective tree")* einen enormen Verständnisgewinn. Hierbei wird häufig erstmals bewusst und offensichtlich, was digitale Langzeitarchivierung insgesamt bedeutet. Der Anwender muss (und wird dadurch) ein Verständnis für die spezifischen Eigenschaften des zu archivierenden Bestandes entwickeln, um richtige Anforderungen und Entscheidungen treffen zu können.

Plato ist ein Planungstool, welches laufend weiterentwickelt wird. Erweiterungen betreffen vor allem die Einbindung zusätzlicher Services, die einzelne Schritte innerhalb des Planungsworkflows weiter automatisieren. Darüber

hinaus werden die Library Templates und Fragements laufend durch die Zusammenarbeit mit Bibliotheken, Archiven, Museen und anderen Dokumentationseinrichtungen erweitert. Bisher werden diese nur eingeschränkt zur Verfügung gestellt, da diese sonst zu „selbsterfüllenden Prophezeiungen" führen könnten, weil diese ohne Überarbeitung und kritische Prüfung übernommen werden würden. Zum jetzigen Zeitpunkt wird in laufenden Case Studies überprüft, ob Institutionen gleicher Größe, mit ähnlichen Anforderungen ähnliche Bäume erstellen, die im positiven Falle als Templates verfügbar gemacht werden können.

Um mit Plato selbstständig arbeiten zu können, wurden neben den wissenschaftlichen Veröffentlichungen eine Reihe frei verfügbarer Tutorials, Case Studies[17] und ein Handbuch erstellt, welche unter http://www.ifs.tuwien.ac.at/dp/plato/intro_documentation.html abgerufen werden können. Außerdem bestehen derzeit Überlegungen, bei Bedarf das derzeit nur in Englisch verfügbare Webinterface auch in andere Sprachen zu übersetzen.

17 z.B. Becker, 2007a; Becker 2007b; Kulovits 2009

Literaturverzeichnis

Becker, Christoph / Kolar Günther / Küng Josef / Andreas Rauber. (2007a) *Preserving Interactive Multimedia Art: A Case Study in Preservation Planning*. In: Goh, Dion Hoe-Lian et al.: Asian Digital Libraries. Looking Back 10 Years and Forging New Frontiers. Proceedings of the Tenth Conference on Asian Digital Libraries (ICADL'07). Berlin / Heidelberg: Springer. S. 257-266.

Becker, Christoph / Strodl Stephan / Neumayer Robert / Rauber Andreas / Nicchiarelli Bettelli, Eleonora / Kaiser, Max. (2007b) *Long-Term Preservation of Electronic Theses and Dissertations:* A Case Study in Preservation Planning. In: Proceedings of the Ninth Russian National Research Conference on Digital Libraries: Advanced Methods and Technologies, Digital Collections. 2007.

Becker, Christoph / Ferreira Miguel / Kraxner Michael / Rauber, Andreas / Baptista, Ana Alice / Ramalho, José Carlos. (2008a) *Distributed Preservation Services: Integrating Planning and Actions*. In: Christensen-Dalsgaard, Birte et al.: Research and Advanced Technology for Digital Libraries. Proceedings of the 12th European Conference on Digital Libraries (ECDL'08). Berlin, Heidelberg: Springer-Verlag. S. 25-36.

Becker, Christoph / Rauber, Andreas / Heydegger, Volker / Schnasse, Jan / Thaller, Manfred. (2008b) *A Generic XML Language for Characterising Objects to Support Digital Preservation*. In: Proceedings of the 2008 ACM symposium on Applied computing. 2008. S. 402-406.

Becker, Christoph / Kulovits, Hannes / Guttenbrunner Mark / Strodl Stephan / Rauber An-dreas / Hofman, Hans (2009) *Systematic planning for digital preservation: Evaluating potential strategies and building preservation plans*. In: International Journal on Digital Libraries (IJDL)

Farquhar, Adam. / Hockx-Yu, Helen (2007) *Planets: Integrated services for digital preservation*. In: International Journal of Digital Curation, 2. (2007). S. 88-99.

Kulovits Hannes / Rauber, Andreas / Kugler Anna / Brantl Markus / Beinert Tobias / Schoger, Astrid (2009) *From TIFF to JPEG 2000? Preservation Planning at the Bavarian State Library Using a Collection of Digitized 16th Century Printings*. In: D-Lib Magazine, November/Dezember 2009, Volume 15 Number 11/12, ISSN 1082-9873

nestor-Arbeitsgruppe Vertrauenswürdige Archive – Zertifizierung (Hrsg.) (2006): *Kriterienkatalog vertrauenswürdige digitale Langzeiarchive. Version 2*. (nestor-Materialien 8). Frankfurt am Main: nestor. www.langzeitarchivierung.de/downloads/mat/nestor_mat_08.pdf

Davies, Richard / Ayris, Paul / McLeod, Rory / Shento, Helen, Wheatley, Paul(2007): *How much does it cost? The LIFE Project - Costing Models for Digital Curation and Preservation*. In: LIBER Quarterly. The Journal of European Research Libraries. 17. 2007. http://liber.library.uu.nl/publish/issues/2007-3_4/index.html?000210

Strodl, Stephan / Becker, Christoph / Neumayer, Robert / Rauber, Andreas.. (2007) *How to Choose a Digital Preservation Strategy: Evaluating a Preservation Planning Procedure*. In: Proceedings of the ACM IEEE Joint Conference on Digital Libraries. 2007. S. 29 - 38.

13.3 Das JSTOR/Harvard Object Validation Environment[18] (JHOVE)

Stefan E. Funk

Einführung

Wie in den vorangehenden Kapiteln bereits besprochen wurde, ist es für eine langfristige Erhaltung von digitalen Objekten dringend erforderlich, zu wissen und zu dokumentieren, in welchem Dateiformat ein solches digitales Objekt vorliegt. Zu diesem Zweck sind auch Informationen von Nutzen, die über das Wissen über den Typ eines Objekts hinausgehen, vor allem detaillierte technische Informationen. Zu wissen, dass es sich bei einem digitalen Bild um ein TIFF-Dokument in Version 6.0 handelt, reicht evtl. nicht aus für eine sinnvolle Langzeiterhaltung. Hilfreich können später Daten sein wie: Welche Auslösung und Farbtiefe hat das Bild? Ist es komprimiert? Und wenn ja, mit welchem Algorithmus? Solche Informationen – technische Metadaten – können aus den Daten des Objekts selbst (bis zu einem gewissen Grad, welcher vom Format der Datei abhängt) automatisiert extrahiert werden.

Anwendung

Mit JHOVE wird im Folgenden ein Werkzeug beschrieben, das außer einer Charakterisierung einer Datei (Welches Format liegt vor?) und einer Validierung (Handelt es sich um eine valide Datei im Sinne der Format-Spezifikation?) zu guter Letzt auch noch technische Metadaten extrahiert. JHOVE kann entweder mit einem grafischen Frontend genutzt werden – wobei eine Validierung oder Extraktion technischer Metadaten von vielen Dateien nicht möglich ist, oder als Kommandozeilen-Tool. Ebenso kann JHOVE auch direkt als Java-Anwendung in eigene Programme eingebunden werden, was für eine automatisierte Nutzung sinnvoll ist. Letzteres ist jedoch dem erfahrenen Java-Programmierer vorbehalten. Als Einführung wird hier das grafische Frontend kurz erklärt sowie eine Nutzung auf der Kommandozeile beschrieben.

---

18 JHOVE – JSTOR/Harvard Object Validation Environment: http://hul.harvard.edu/jhove/

Anforderungen

Für die Nutzung von JHOVE wird eine Java Virtual Machine benötigt, auf der JHOVE Projektseite bei Sourgeforge.net[19] wird Java in Version 1.6.0_12 empfohlen.

Das grafische Frontend JhoveView

Download

Nach dem Herunterladen des .zip oder .tar.gz Paketes von der Sourceforge Projektseite – beschrieben wird hier die Version 1.2 vom 10. Februar 2009 – wird das Paket in ein beliebiges Verzeichnis entpackt. Zum Starten des grafischen Frontends starten Sie bitte das Programm JhoveView.jar im Verzeichnis ./bin/ – entweder durch Doppelklick oder von der Kommandozeile per java -jar bin/JhoveView.jar (nach dem Wechsel in das Verzeichnis, indem sich JHOVE befindet).

Menü-Optionen

Die beiden vorhandenen Menü-Optionen "File" und "Edit" sind schnell erklärt:

- Unter "File" kann eine Datei aus dem Internet oder vom Dateisystem geöffnet werden, das sogleich von JHOVE untersucht wird.
- Unter "Edit" kann gezielt ein JHOVE-Modul gewählt werden, mit dem eine Datein untersucht werden soll. Nicht die Einstellung "(Any)" zu benutzen – für eine automatische Erkennung des Formats – kann zum Beispiel dann Sinn machen, wenn eine TIFF-Datei nicht automatisch als solche erkannt wird, weil sie vielleicht nicht valide ist. Dann kann JHOVE dazu bewegt werden, dieses Bild mit dem TIFF-Modul zu untersuchen, um so eine entsprechende – und weiter helfende – Fehlermeldung zu bekommen. Weiterhin kann hier die Konfigurationsdatei editiert werden (um neue Module einzubinden).

19 http://sourceforge.net/projects/jhove/

Dateien untersuchen

Wählt man nun eine Datei aus, für erste Tests sollten die vorhandenen Module berücksichtigt werden, wird diese Datei von JHOVE untersucht. Im Folgenden wird ein Fenster angezeigt, in dem alle von JHOVE extrahierten Informationen angezeigt werden. Hier kann nach Belieben durch den Baum geklickt werden. An erster Stelle wird das Modul und dessen Versionsnummer angezeigt, mit dem die Datei untersucht wurde. Wird hier als Modulname "BYTESTREAM" angezeigt, heißt das, dass JHOVE kein passendes Modul gefunden hat, das Bytestream-Modul wird dann als Fallback genutzt. Hier hilft es unter Umständen – wie oben erwähnt – das Modul per Hand einzustellen.

JHOVE Ausgaben anzeigen und speichern

Die Speicheroption, die nun zur Verfügung steht, kann genutzt werden, um die Ergebnisse wahlweise als Text oder als XML zu speichern und in einem anderen Programm zu nutzen/anzusehen. So können die Informationen beispielsweise in einem XML- oder Texteditor bearbeitet oder anderweitig genutzt werden. Im Folgenden ein Beispiel einer Untersuchung einer Textdatei im Zeichensatz UTF-8:

```
JhoveView (Rel. 1.1, 2008-02-21)
       Date: 2009-03-03 10:33:31 CET
       RepresentationInformation:
       /Users/fugu/Desktop/nestor-hand
       buch-kapitel-13_2009-03-03/test.txt
       ReportingModule: UTF8-hul, Rel. 1.3 (2007-08-30)
       LastModified: 2009-03-03 10:33:12 CET
       Size: 64
       Format: UTF-8
       Status: Well-Formed and valid
MIMEtype: text/plain; charset=UTF-8
UTF8Metadata:
       Characters: 60
       UnicodeBlocks: Basic Latin, CJK Unified Ideographs
```

Als XML-Repräsentation sieht das Ergebnis aus wie folgt und kann somit maschinell sehr viel genauer interpretiert werden.

```
<?xml version="1.0" encoding="utf-8"?>
<jhove xmlns:xsi="http://www.w3.org/2001/XMLS-
chema-instance" xmlns="http://hul.harvard.edu/
ois/xml/ns/jhove" xsi:schemaLocation="http://
```

```
      hul.harvard.edu/ois/xml/ns/jhove http://hul.
      harvard.edu/ois/xml/xsd/jhove/1.5/jhove.xsd"
      name="JhoveView" release="1.1" date="2008-02-21">
      <date>2009-03-03T10:40:00+01:00</date>
      <repInfo
      uri="/Users/fugu/Desktop/nestor-hand-
buch-kapitel-13_2009-03-03/test.txt">
      <reportingModule release="1.3"
date="2007-08-30">UTF8-hul</reportingModule>
      <lastModified>2009-03-03T10:33:12+01:00</lastModified>
      <size>64</size>
      <format>UTF-8</format>
      <status>Well-Formed and valid</status>
      <mimeType>text/plain; charset=UTF-8</mimeType>
      <properties>
            <property>
                  <name>UTF8Metadata</name>
                  <values arity="List" type="Property">
                        <property>
                              <name>Characters</name>
                              <val-
ues arity="Scalar" type="Long">
                                    <value>60</value>
                              </values>
                        </property>
                        <property>
                              <name>UnicodeBlocks</name>
                              <val-
ues arity="List" type="String">

<value>Basic Latin</value>

<value>CJK Unified Ideographs</value>
                              </values>
                        </property>
                  </values>
            </property>
      </properties>
      <note>Additional representation in-
formation includes the line endings:
            CR, LF, or CRLF</note>
      </repInfo>
      </jhove>
```

Eine genauere Dokumentation des grafischen Frontends, des Kommandozeilentools, sowie zu JOHVE allgemein findet sich auf der JHOVE-Homepage (auf Englisch) unter "Tutorial", aktuelle Informationen zur Distribution und die neueste Version derselben auf der JHOVE SourceForge-Projektseite.

JHOVE auf der Kommandozeile

Die Möglichkeit, ganze Verzeichnisse zu untersuchen und kurz mal zu schauen, wieviele valide Dateien darin enthalten sind, ist – neben allen Möglichkeiten des grafischen Frontends – ein großer Vorteil des Kommandozeilentools, das JHOVE zur Verfügung stellt.

Konfiguration

Um das Kommandozeilentool nutzen zu können, ändern Sie bitte zunächst den Namen der Datei jhove.tmpl in jhove (Linux/Unix) oder jhove_bat.tmpl in jhove.bat (Windows). Ändern Sie bitte noch – den Anweisungen in diesen Dateien zufolge – den Pfad zu Ihrem JHOVE-Verzeichnis in diesen Skripten. Haben Sie beispielsweise das JHOVE-Paket in /home/ kopiert, lautet der Pfad /home/jhove (Linux/Unix), arbeiten Sie auf einem Windows-System, tragen Sie für das Verzeichnis C:\Programme\ bitte C:\Programme\jhove ein. Sollte der Pfad zu Ihrer Java-Installation nicht stimmen, passen Sie bitte auch diesen noch an. Wenn Sie alles richtig konfiguriert haben, bekommen Sie durch Tippen von ./jhove bzw. jhove.bat detaillierte Informationen zu Ihrer JHOVE-Installation.

Verzeichnisse rekursiv untersuchen

Wenn Sie nun beispielsweise alle XML-Dateien untersuchen möchten, die sich im Beispiel-Verzeichnis der JHOVE-Installation befinden, rufen Sie JHOVE folgendermaßen auf:

```
./jhove -h audit examples/xml/
```

Die Ausgabe enthält folgendes und beschreibt in Kürze, welche Dateien untersucht wurden, ob und wie viele davon valide sind:

```
<?xml version="1.0" encoding="UTF-8"?>
<jhove xmlns:xsi="http://www.w3.org/2001/XMLSche-
ma-instance" xmlns="http://hul.harvard.edu/
ois/xml/ns/jhove" xsi:schemaLocation="http://
hul.harvard.edu/ois/xml/ns/jhove http://hul.
harvard.edu/ois/xml/xsd/jhove/1.5/jhove.xsd"
name="Jhove" release="1.1" date="2008-02-21">
```

```
<date>2009-03-03T11:27:27+01:00</date>
<audit home="/Users/Fugu/Desktop/jhove">
<file mime="text/xml" status="well-formed">
examples/xml/build.xml</file>
<file mime="text/plain; charset=US-ASCII" status="valid">
examples/xml/external-parsed-entity.ent</file>
<file mime="text/plain; charset=US-ASCII" status="valid">
examples/xml/external-unparsed-entity.ent</file>
<file mime="text/xml" status="well-formed">
examples/xml/jhoveconf.xml</file>
<file mime="text/plain; charset=US-ASCII" status="valid">
examples/xml/valid-external.dtd</file>
</audit>
</jhove>
<!-- Summary by MIME type:
text/plain; charset=US-ASCII: 3 (3,0)
text/xml: 2 (0,2)
Total: 5 (3,2)
-->
<!-- Summary by directory:
/Users/Fugu/Desktop/jhove/examples/xml: 5 (3,2) + 0,0
Total: 5 (3,2) + 0,0
-->
<!-- Elapsed time: 0:00:02 >
```

Weitere Parameter

Als weitere Parameter können unter anderem Handler und Module genauer spezifiziert werden sowie Ausgabe-Dateien und Encoding konfiguriert werden. Hier darf nach Belieben probiert, getestet und gespielt werden, um zu probieren, technische Metadaten zu extrahieren und Dateien zu validieren.
Im Folgenden noch eine kurze Beschreibung des Nutzung des Kommandozeilentools...

```
jhove  [-c config] [-m module [-p param]]
       [-h handler [-P param]]
       [-e encoding][-H handler] [-o output] [-x saxclass]
       [-t tempdir] [-b bufsize] [[-krs] dir-file-or-uri […]]
```

...und die Bedeutung der wichtigsten:

`-c config`	–	Pfad zur JHOVE-Konfigurationsdatei.
`-m module`	–	Name des Moduls, möglich sind hier: AIFF-hul, ASCII-hul, BYTESTREAM, GIF-hul, HTML-hul, JPEG-hul, JPEG2000-hul, PDF-hul, TIFF-hul, UTF8-hul, WAVE-hul und XML-hul.
`-p param`	–	Modul-spezifische Parameter.
`-h handler`	–	Name des Output-Handlers (Grundeinstellung: TEXT).
`-P param`	–	Handler-spezifische Parameter.
`-o output`	–	Name der Ausgabe-Datei (Grundeinstellung: stdout).
`-x saxclass`	–	SAX-Parser-Klasse (Grundeinstellung: J2SE 1.4 default).
`-t tempdir`	–	Temporäres Verzeichnis, in dem temporäre Dateien erzeugt werden.
`-b bufsize`	–	Puffergröße für gepufferte I/O Operationen (Grundeinstellung: J2SE 1.4 default).
`-k`	–	Berechnet CRC32, MD5, und SHA-1 Checksummen.
`-r`	–	Zeigt rohe Data Flags an, nicht die textlichen Äquivalente.
`-s`	–	Format-Identifikation basiert nur auf internen Signaturen.

`dir-file-or-uri`	–	Verzeichnis, Pfadname oder URI der zu untersuchenden Dateien.

13.4 Die kopal Library for Retrieval and Ingest (koLibRI)

Stefan E. Funk

Einführung

Die *kopal Library for Retrival and Ingest* ist ein Framework zur Integration eines Langzeitarchivs wie dem IBM Digital Information Archiving System[20] (DIAS) in die Infrastruktur einer Institution. Insbesondere organisiert koLibRI das Erstellen und Einspielen von Archivpaketen in DIAS und stellt Funktionen zur Verfügung, um diese abzurufen und zu verwalten. koLibRI stellt eine Bibliothek von Java-Tools dar, die im Projekt kopal entwickelt wurden. Sie wurde bewusst so angelegt, dass sie als Ganzes oder in Teilen auch in anderen Zusammenhängen nachnutzbar ist.

An dieser Stelle soll der Teil koLibRIs beschrieben und beispielhaft dargestellt werden, der für das Erstellen von Archivpaketen verantwortlich ist; eine ausführliche Beschreibung der gesamten Funktionalität der kopal Library for Retrieval and Ingest sowie weitere technische Details ist in deren Dokumentation[21] zu finden und auch auf der Internetseite des Projekts kopal[22].

Funktionsweise

Im einfachen Fall generiert koLibRI aus den mit dem zu archivierenden Objekt gelieferten Metadaten sowie den von JHOVE[23] maschinell extrahierten Metadaten eine XML-Datei nach METS Schema, verpackt diese zusammen mit dem Objekt in einer Archivdatei (.zip oder .tar) und liefert diese Datei als Submission Information Package (SIP) an das DIAS.

So gesehen ist koLibRI für eine vollständige Langzeitarchivierungslösung mit DIAS entwickelt worden. Jedoch kann koLibRI auch als eigenständige Software zur Generierung der METS Dateien oder kompletten SIPs nach dem Universellen Objektformat[24] vollkommen ohne DIAS eingesetzt werden. Die auf diese Weise generierten XML-Metadatendateien oder die kompletten SIPs kön-

20 http://www-5.ibm.com/nl/dias/
21 http://kopal.langzeitarchivierung.de/kolibri/koLibRI_v1_0_dokumentation.pdf
22 http://kopal.langzeitarchivierung.de/
23 JSTOR/Harvard Object Validation Environment (JHOVE): http://hul.harvard.edu/jhove/
24 http://kopal.langzeitarchivierung.de/downloads/kopal_Universelles_Objektformat.pdf

nen für den Datenaustausch zwischen verschiedenen Institutionen verwendet werden; ein Aspekt, der bei der Entwicklung des UOF besonders im Vordergrund stand. Alternativ kann durch den modularen Aufbau der koLibRI auch mit vertretbarem Aufwand an ein anderes Archivsystem oder ein anderes Metadatenformat angepasst werden, da die Schnittstellen ausreichend spezifiziert sind.
Mit koLibRI kann ein für das Erstellen der Archivpakete benötigter Workflow abgebildet werden. Dieser Workflow kann den eigenen Bedürfnissen angepasst und erweitert werden. Die koLibRI-Infrastruktur nutzt prinzipiell vier Konstrukte, um Workflows abzubilden und zu verarbeiten:

- Zunächst sammelt der sogenannte *ProcessStarter* die einzuspielenden Dateien/Daten ein, die als kleinste Einheit definiert wurden – in unserem Beispiel wird dies ein Verzeichnis mit beliebigen Dateien sein.
- Jede Einheit wird vom ProcessStarter an die *ProcessQueue* angehängt, die dann der Reihe nach (oder auch nebenläufig) abgearbeitet werden.
- In den *ActionModules* werden einzelne Aufgaben implementiert, die für ein jedes Objekt in der ProcessQueue durchgeführt werden sollen. Für den Beispiel-Workflow werden hier folgende Module genutzt: FileCopyBase, MetadataExtractorDmd, MetadataGenerator, MetsBuilder und Zip. Weitere Module werden mit koLibRI geliefert und können integriert werden.
- Die Reihenfolge, in der die ActionModules für jedes dieser Objekte in der ProcessQueue verarbeitet werden, wird als *Policy* konfiguriert.

Installation und Konfiguration

Download

Zunächst wird das koLibRI-Paket von der kopal-Homepage benötigt, bitte laden Sie diese von der Internetseite des Projekts kopal. Benötigt wird das gepackte Programmpaket „kopal Library for Retrieval and Ingest“ (http://kopal.langzeitarchivierung.de/kolibri/koLibRI_v1_0.zip), das Sie bitte in ein beliebiges Verzeichnis entpacken.

Anforderungen

Da die kopal Library for Retrieval and Ingest komplett in Java implementiert wurde, sollte die Software prinzipiell auf jeder Plattform laufen, die eine Java Virtual Machine in der Version 1.5 zur Verfügung stellt. Alle weiteren

erforderlichen Java Software-Bibliotheken sind in dem Paket enthalten und vorkonfiguriert.

Konfiguration des Workflowtool Skriptes

Zunächst müssen die folgenden Werte in den beiden Startskripten `workflowtool` (Linux/Unix) oder `workflowtool.bat` (Windows) an die lokalen Verhältnisse angepasst werden:

- `KOLIBRI_HOME`

Hier tragen Sie bitte den Pfad zu Ihrer koLibRI-Installation ein, zum Beispiel `/home/funk/kolibri_v1_0` (Linux/Unix) bzw. `C:\Programme\kolibri_v1_0` (Windows).

- `JAVA_HOME`

Sollte hier der Pfad zu Ihrer Java-Installation nicht stimmen, passen Sie diesen bitte ebenfalls noch an.

Konfiguration der Policies-Datei

Die Datei policies.xml im Verzeichnis config/ wird um die folgenden Zeilen ergänzt; vor dem letzten schließenden Tag – </policies> – fügen Sie bitte die folgenden Zeilen ein:

```
<policy name="example_lza_handbuch">
  <step class="FileCopyBase">
    <step class="XorFileChecksums">
      <step class="MetadataExtractorDmd">
        <step class="MetadataGenerator">
          <step class="MetsBuilder">
            <step class="Zip">
              <step class="CleanPathToContentFiles"/>
            </step>
          </step>
        </step>
      </step>
    </step>
  </step>
</policy>
```

Konfiguration der Konfigurations-Datei

In der Datei config.xml im Verzeichnis config/ werden folgende Werte gesetzt:

- Der Wert der Eigenschaft des Feldes `<field>defaultPolicyName</field>` wird in den Wert `<value>example_lza_handbuch</value>` geändert, so wird unsere vorher hinzugefügte Policy genutzt.
- Die Werte der Felder `logfileDir`, `destinationDir`, `workDir` und `tempDir` werden jeweils mit dem Pfad zu den jeweiligen Verzeichnissen ersetzt. Bitte legen Sie diese vorher an, am besten direkt in Ihren kolibri-Verzeichnis (Beispielsweise als „log“, „dest“, „work“ und „temp“): `<value>./log/</value>`, `<value>./dest/</value>`, `<value>./work/</value>` und `<value>./temp/</value>`.
- Schließlich wird noch ein Verzeichnis als Hotfolder benötigt, aus dem die zu behandelnden Dateien hineinkopiert werden. Bitte legen Sie ein weiteres Verzeichnis „./hotfolder“ an, dessen Wert sollte bereits in der Konfigurations-Datei eingetragen sein.

Starten von koLibRI

Zum Starten des Workflowtools wechseln Sie bitte in das Verzeichnis der koLibRI-Installation – oder bleiben gleich dort, sollten Sie schon da sein – und tippen

```
./workflowtool -c config/config.xml (Linux/Unix)
```

bzw.

```
workflowtool /c config\config.xml (Windows)
```

Sie bekommen nun – wenn nun alles richtig konfiguriert ist – eine Ausgabe auf der Konsole, die mit den folgenden Zeilen endet:

```
[INFO]       Checking hotfolder /Users/fugu/Desk-
top/koLibRI_v1_0/./hotfolder for new content

[INFO]

All current files scheduled, waiting for more
```

Nun können Sie testweise ein beliebiges Verzeichnis in dieses Hotfolder kopieren (bitte zu Anfang mit nicht allzuviel Inhalt!), und koLibRI fängt an zu arbeiten.

Ergebnis

Als Ergebnis erhalten Sie im Verzeichnis `dest` eine .zip-Datei, in der sich zum einen Ihre im Hotfolder befindlichen Dateien befinden und außerdem eine METS-Datei im Universellen Objektformat mit dem Namen `mets.xml`. Diese enthält neben den in der Template-XML-Datei `config/uof_template.xml` enthaltenen Daten – die für die METS-Datei als Vorlage genommen wird – technische Metadaten für jede einzelne Datei (extrahiert von JHOVE) im LMERfile-Format, sowie Metadaten zu dem gesamten Objekt im LMERobject-Format[25].

Weitere Konfigurationsmögichkeiten der kopal Library for Retrieval and Ingest – von denen es noch viele gibt – sowie eine ausführliche Beschreibung der Nutzung auch mit dem DIAS, und weitere Nutzungsszenarien und Erweiterungsmöglichkeiten der koLibRI sind in der ausführlichen Dokumentation nachzulesen. Weiterhin gibt es die Möglichkeit, über die koLibRI-Internetseite den Entwicklern Rückmeldungen zu Erfahrungen mit koLibRI mitzuteilen.

Literatur

Funk, Stefan; Kadir Karaca Koçer, Sabine Liess, Jens Ludwig, Matthias Neubauer: *kopal Library for Retrieval and Ingest – Dokumentation –. 2007*. http://kopal.langzeitarchivierung.de/kolibri/koLibRI_v1_0_dokumentation.pdf

25 http://www.d-nb.de/standards/lmer/lmer.htm