AP6:

TextGrid-Tools I (Pflichtenhefte) (R 6.0.1)

Version 2010-08 (final)
Arbeitspaket 6
verantwortlicher Partner Uni Würzburg et al.

TextGrid

Vernetzte Forschungsumgebung in den eHumanities

Ein Projekt im D-GRID

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

Projekt: TextGrid - Vernetzte Forschungsumgebung in den eHumanities
BMBF Förderkennzeichen: 01UG0901A
Laufzeit: Juni 2009 bis Mai 2012
Berichtszeitraum: Juni 2009 bis Mai 2009

Dokumentstatus: Final
Verfügbarkeit: öffentlich
Autoren:

Thorsten Vitt, Uni Würzburg
Mirjam Blümm, SUB Göttingen
Thomas Selig, FH Worms
Jens Stegmann, IDS Mannheim
Christoph Ludwig, FH Worms
Wolfgang Pempe, SUB Göttingen
Andreas Aschenbrenner, SUB Göttingen
Simon Rettelbach, Uni Trier
Oliver Schonefeld, IDS Mannheim
Peter M. Fischer, IDS Mannheim
Tobias Schraut, MPDL
Martin Hellmann, MüZE
Julian Dabbert, Uni Detmold/Paderborn
Mayce Al-Azawi, TU Kaiserslautern
und weitere Mitglieder des Konsortiums

Inhaltsverzeichnis

0. Einleitung

Im Arbeitspaket 6 werden zur Förderung der fachwissenschaftlichen Nachhaltigkeit von TextGrid fachwissenschaftliche Tools (im Gegensatz zu allgemeinen Werkzeugen (AP2) zur Nutzung der Infrastruktur (AP1)) weiter- und neuentwickelt bzw. in TextGrid integriert.

Dieser Report gibt einen Überblick über die Entwicklungen und Planungen nach einem Jahr Projektlaufzeit. Zu diesem Zweck vereint es *Pflichtenhefte* und weniger formelle Spezifikationen für die im AP entwickelten und weiterentwickelten Tools. Bei den weiterentwickelten Tools wurden dabei z.T. die in der ersten Projektphase entstandenen Pflichtenhefte in überarbeiteter Form übernommen.

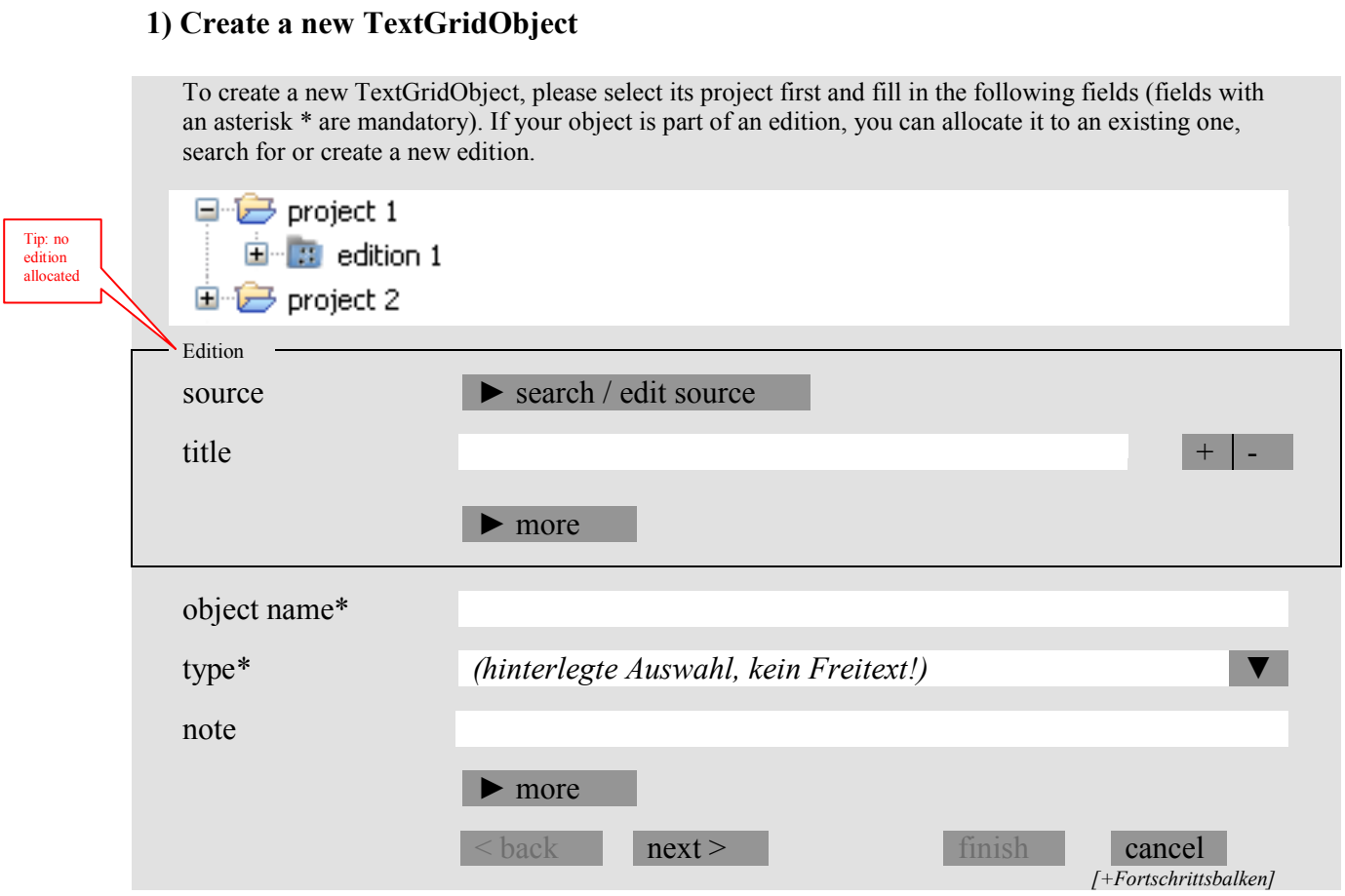
1. AP 6.1

1.1. Metadaten-Editor / -Annotation

Aufgrund von Erfahrungen und Feedback zum TextGrid-Metadatenschema aus der ersten Projektphase wurde dieses Schema grundlegend überarbeitet. Insbesondere wurde es stärker an etablierte Metadatenstandards angelehnt, und es wurde eine konzeptuelle Teilung in *Item*, *Ausgabe* (mit ggf. einer *Quelle* aus der physischen Welt) und *Werk* vorgenommen.

Um die komplexer gewordenen Metadaten bequem und konsistent eingeben zu können, ist deshalb eine besonders benutzerorientierte Gestaltung der entsprechenden Eingabemasken nötig. Im folgenden Abschnitt ist ein Entwurf für die Eingabemasken dargestellt, die im Kontext verschiedener Situationen im Lab (neues Objekt erstellen, Objekt importieren, Metadaten eines Objekts bearbeiten …) verwendet werden sollen.

1) Create a new TextGridObject



NAVIGATOR

EDITION

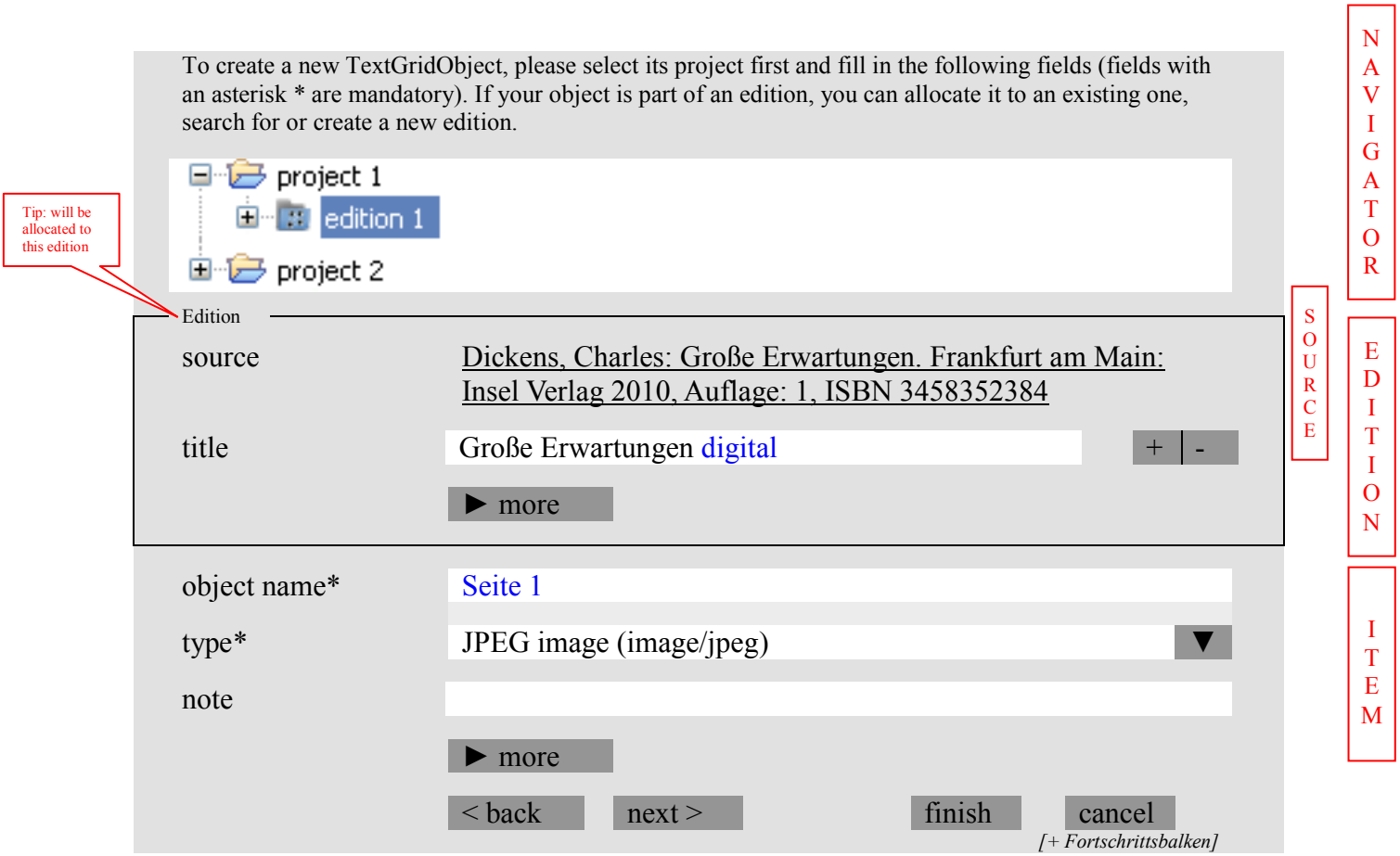
ITEM

Fall 1: Die Ausgabe ist bereits in TextGrid gespeichert

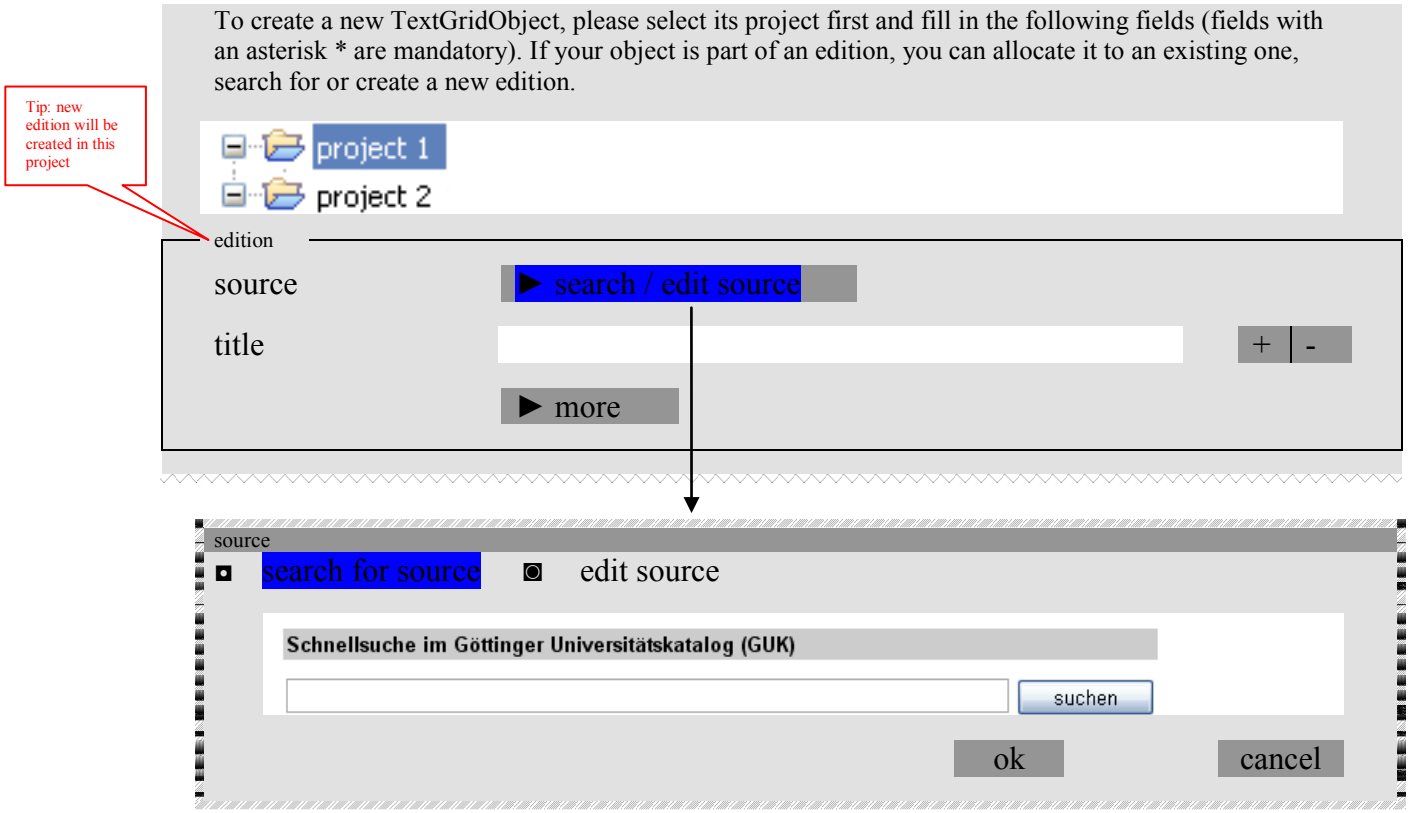
Ein Benutzer wählt im *project 1* die *edition 1* aus. Die Seite wird mit den entsprechenden Informationen zur Ausgabe nachgeladen.

Die Angaben zur *source* sind verlinkt auf die *source*-Maske. Über diese können die Metadaten bearbeitet werden.
Das *title*-Feld wird automatisch vorbelegt mit dem Titel der *source*. Es kann jedoch vom Benutzer bearbeitet werden. Über die Schaltfläche „+“ werden weitere Titelfelder hinzugefügt, bzw. über „-“ wieder entfernt.
Über den Knopf *more* erfolgt eine Anzeige der gesamten Metadaten zur Ausgabe.

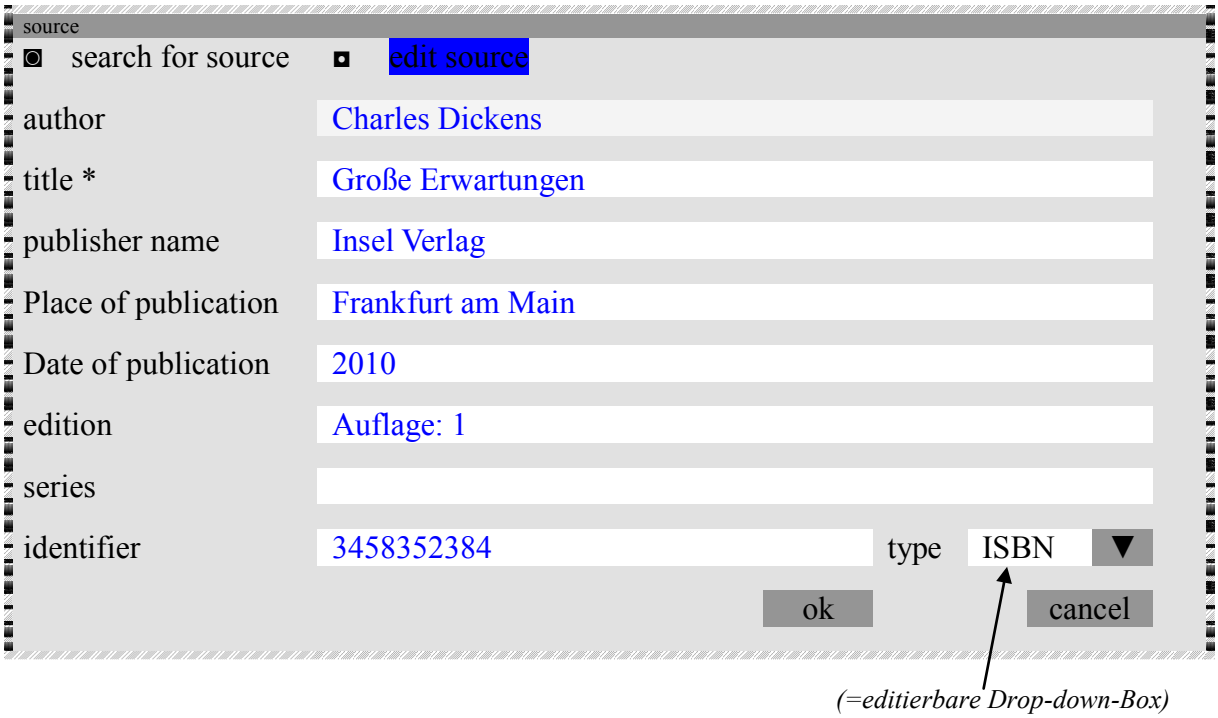
Anschließend tippt der Benutzer noch den Namen des Objekts ein und wählt über eine Dropdown-Liste den Mime-Type.



Fall 2: Die Ausgabe ist nicht in TextGrid gespeichert

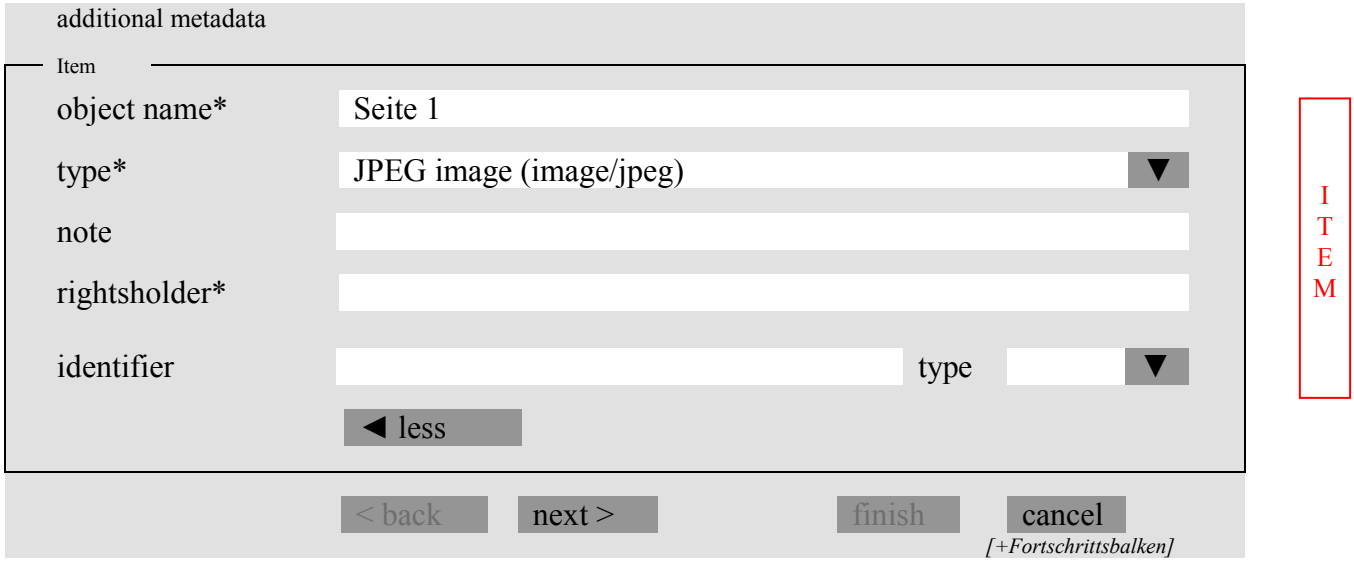


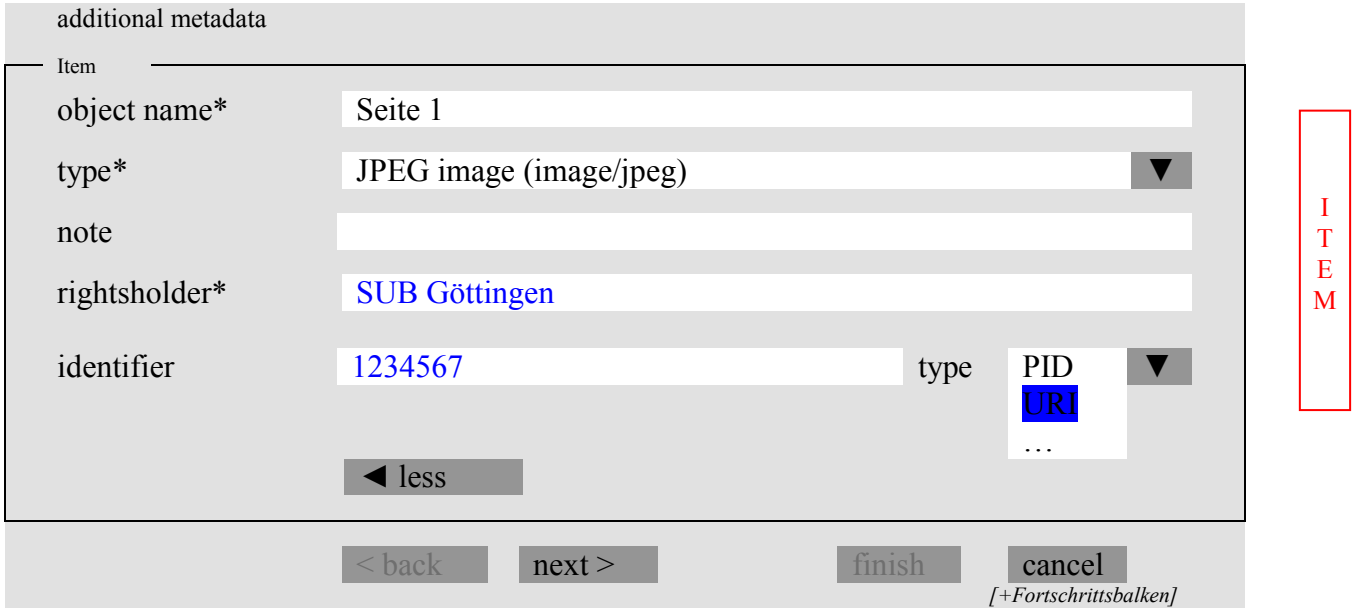
Wird die *source* über den Bibliothekskatalog gefunden, wird sie importiert und wie im Fall 1 als Zitation angezeigt, andernfalls muss der Benutzer die entsprechenden Daten einfüllen, die dann nach der Bestätigung durch den ok-Knopf ebenfalls als Zitation angezeigt werden.



Bei XML-Dokumenten folgen die Seiten *Select Schema* und *Select Root Element*.

2) additional metadata





SOURCE

EDITION

Edition

source Dickens, Charles: Große Erwartungen. Frankfurt am Main: Insel Verlag 2010, Auflage: 1, ISBN 3458352384

title * Große Erwartungen digital + -

◀ less

contributor + -

identifier type ▼

language German (ger) ▼ script Latin (Latn) ▼

licence * ▼

note

released in * ▶ search / edit edition

work ▶ search / edit work

! Changes in the metadata will affect all objects associated with this edition revert

< back next > finish cancel

[+Fortschrittsbalken]

EDITION

Edition

source Dickens, Charles: Große Erwartungen. Frankfurt am Main: Insel Verlag 2010, Auflage: 1, ISBN 3458352384

title * Große Erwartungen digital + -

further title Eine digitalisierte Ausgabe + -

◀ less

contributor + -

identifier type ▼

language German (ger) ▼ script Latin (Latn) ▼

licence * cc by-sa ▼

note

released in * ▶ search / edit edition

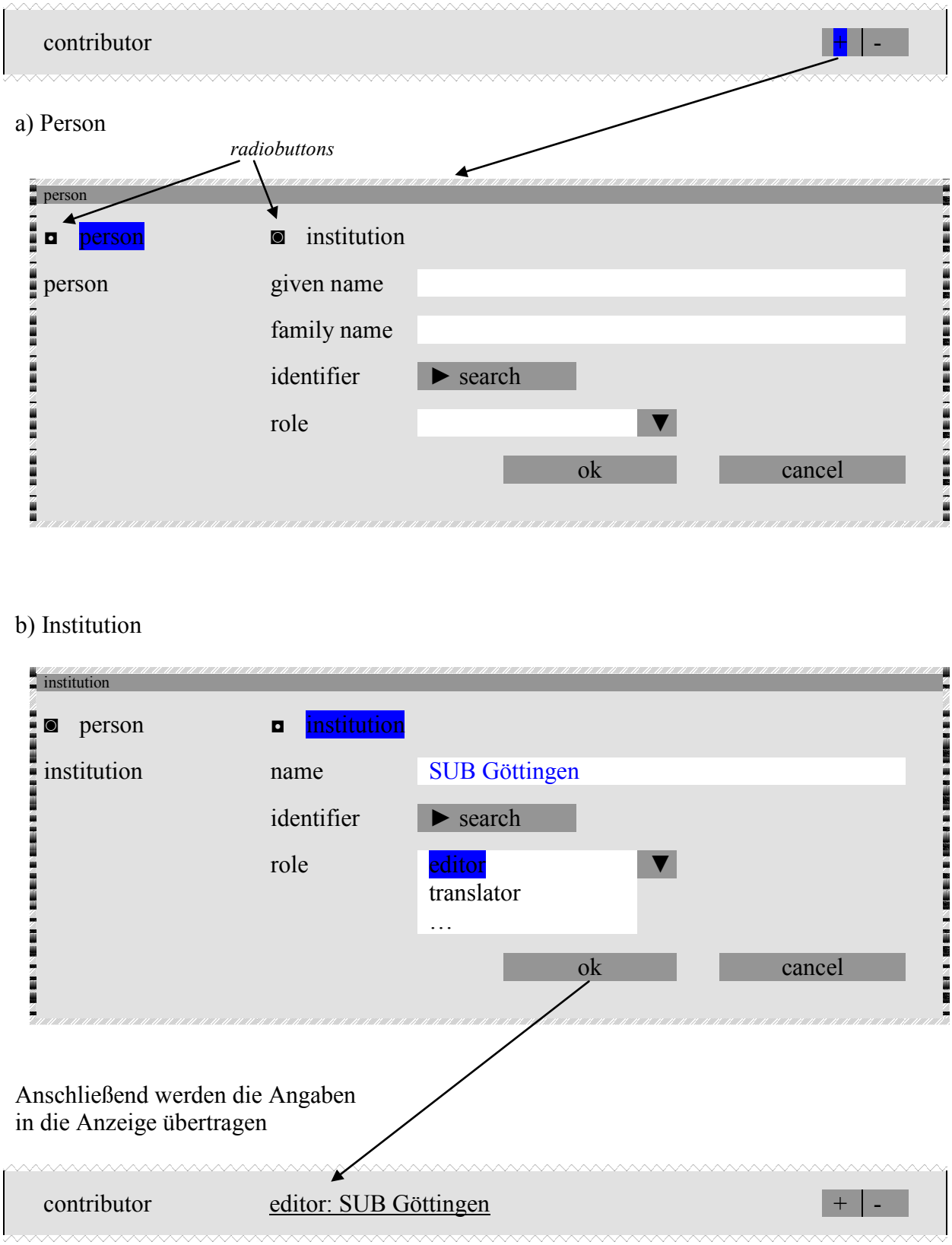
work ▶ search / edit work

! Changes in the metadata will affect all objects associated with this edition revert

< back next > finish cancel

[+Fortschrittsbalken]

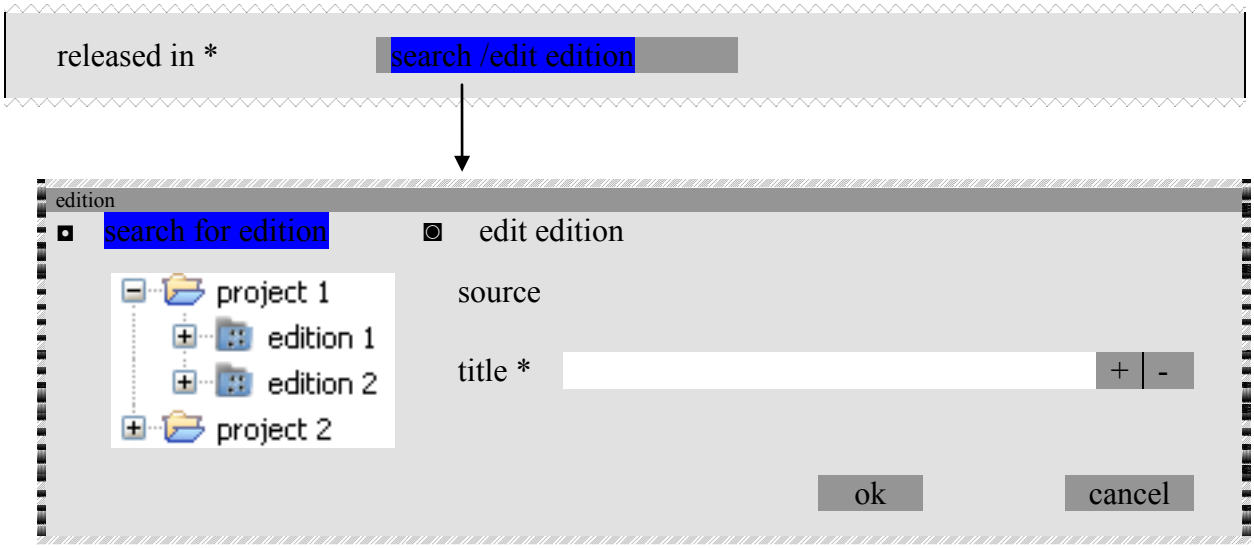
Wird eine Person oder Institution angegeben, öffnet sich ein Popup-Fenster:



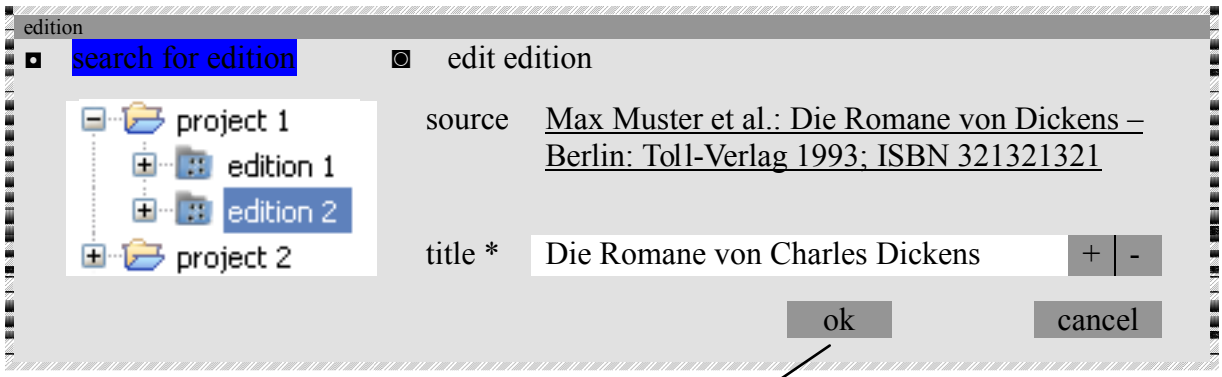
AGENT

Die Ausgabe ist Teil einer anderen Ausgabe (z.B. in einem Sammelband)

Fall 1: Die Ausgabe ist in TextGrid gespeichert



Ein Benutzer wählt im *project 1* die *edition 2* aus. Das Fenster wird mit den entsprechenden Informationen zur Ausgabe nachgeladen.

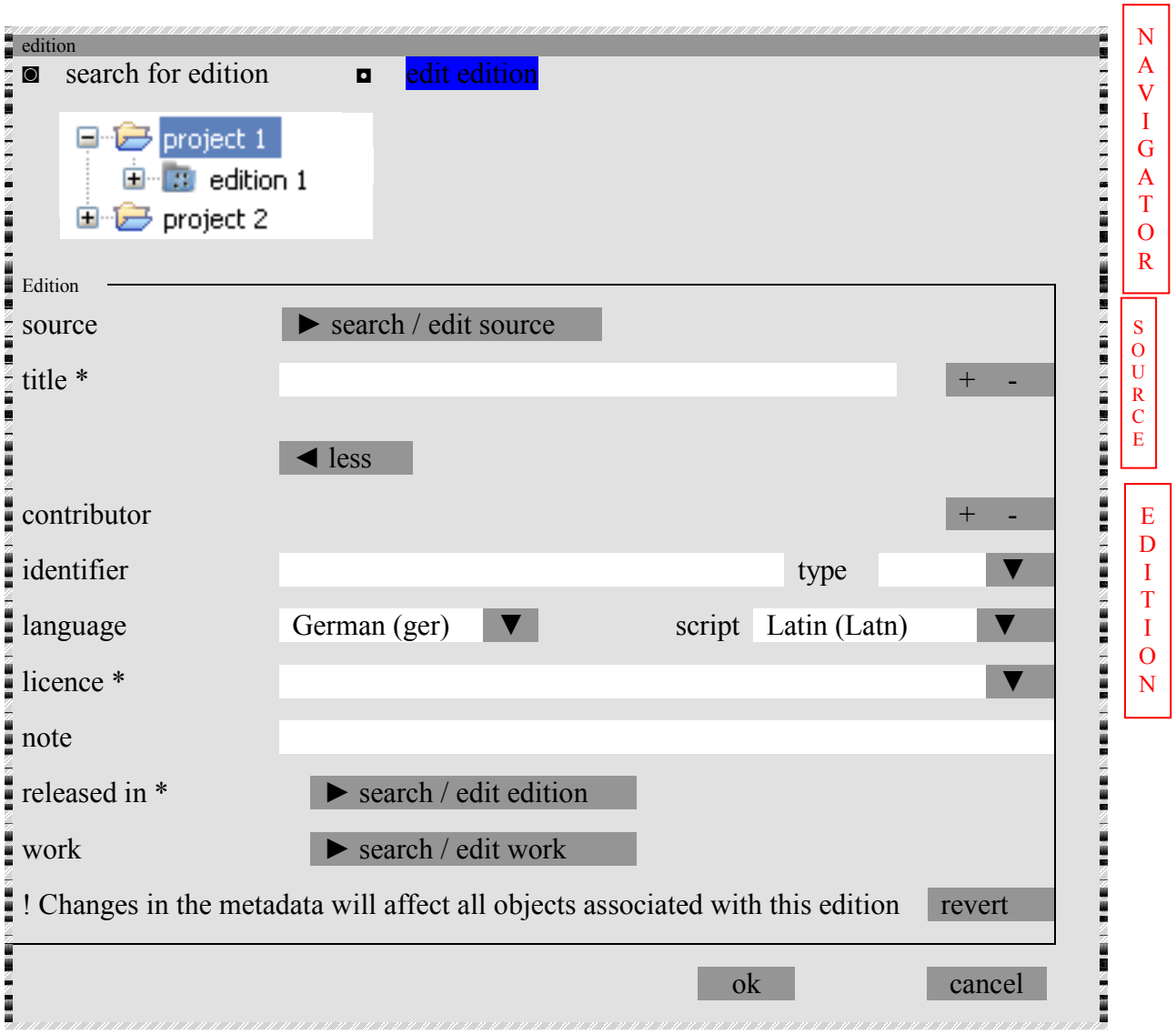


Die Angaben werden in die *edition*-Maske übertragen

EDITION

EDITION

Fall 2: Die Ausgabe wird neu angelegt

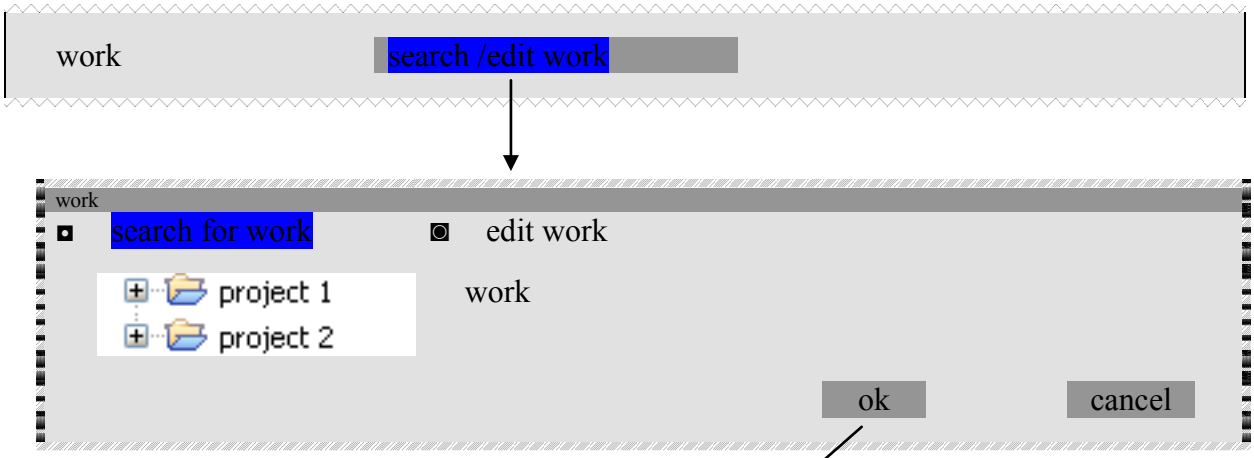


Anschließend werden die Daten wie im Fall 1 angezeigt.

Dieser Prozess ist rekursiv und kann beliebig tief geschachtelte *editions* abbilden.

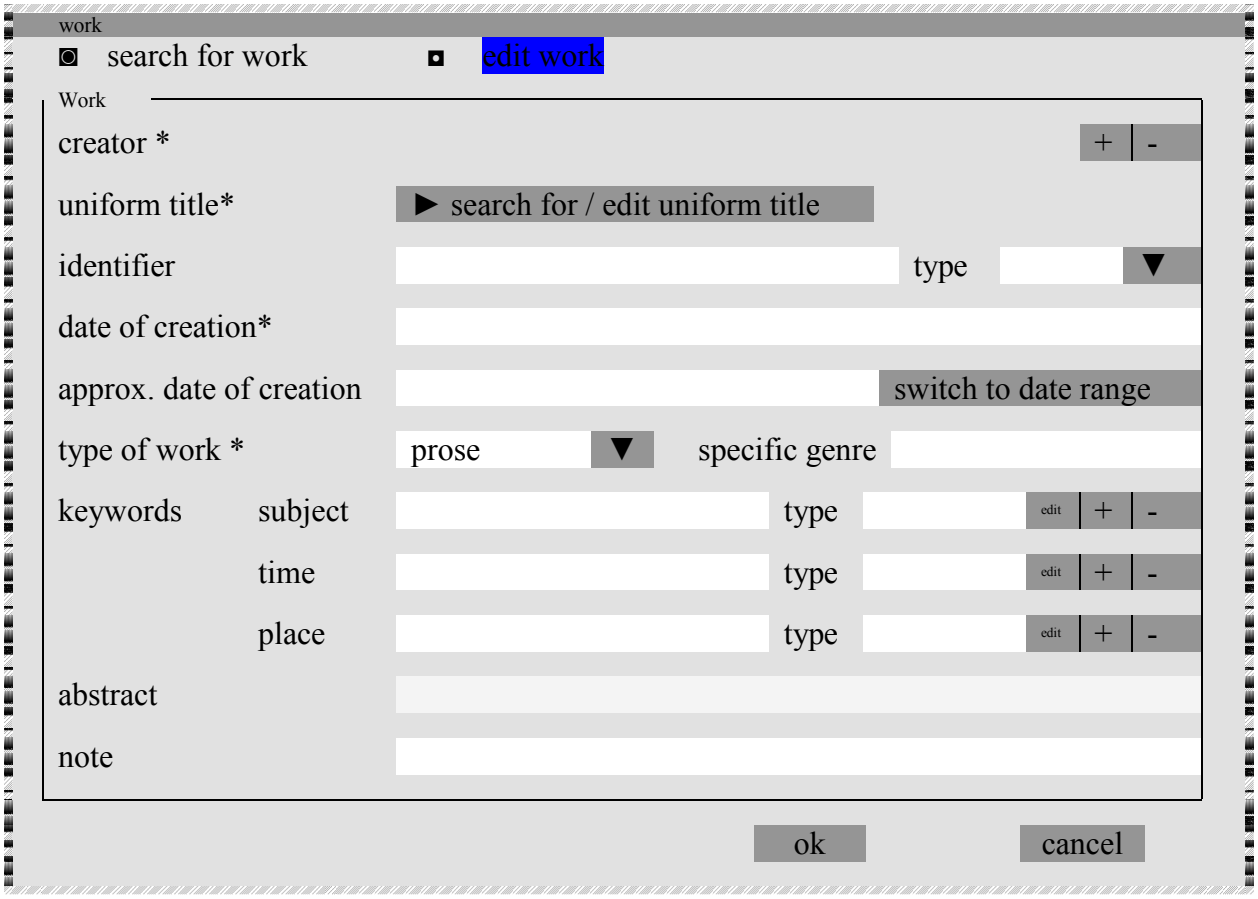
Die Ausgabe ist noch nicht mit einem Werk assoziiert

Fall 1: Das Werk ist in TextGrid gespeichert / wird über den Katalog gefunden.

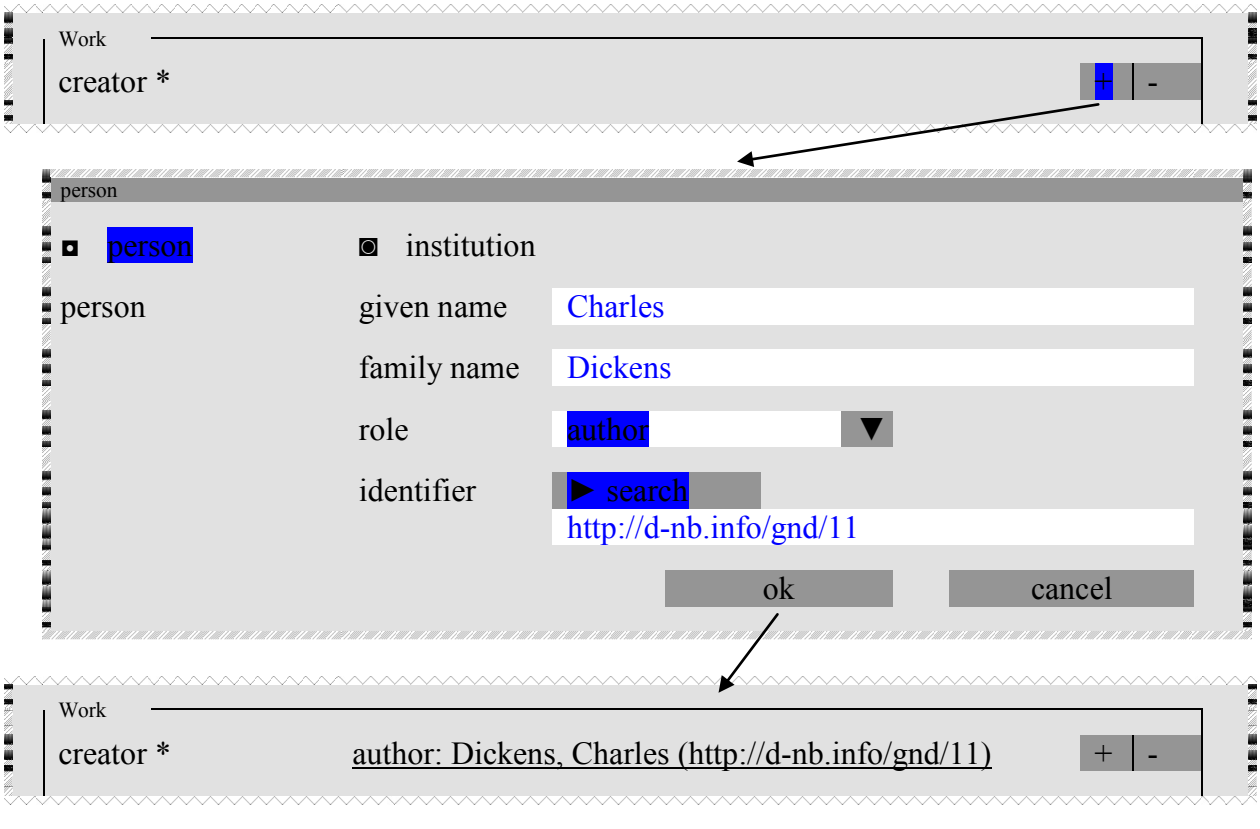


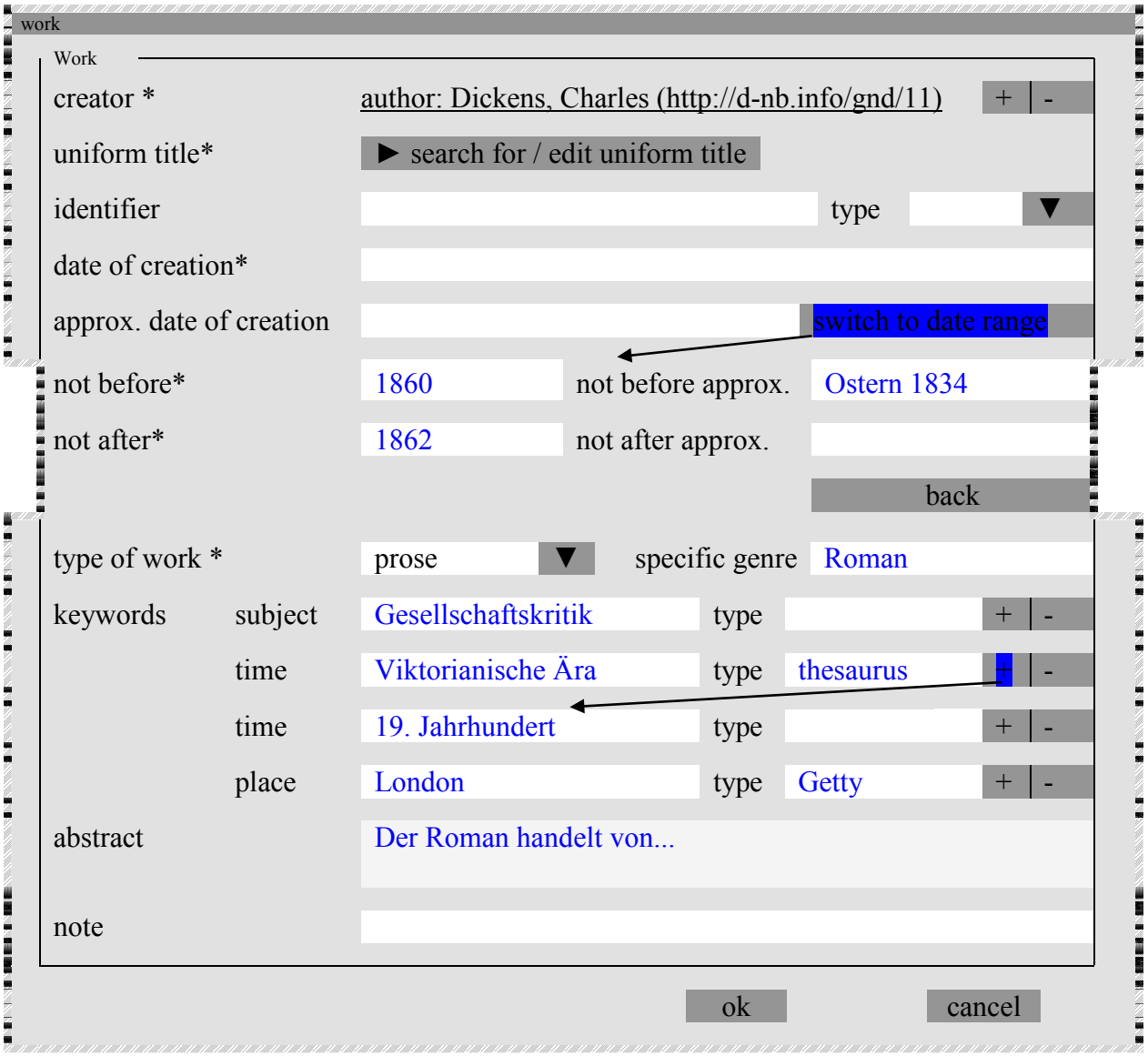
Die Angaben werden in die *editions*-Maske
Übertragen und mit der *work*-Maske verknüpft

Fall 2: Das Werk der Ausgabe wird neu angelegt



Wird eine Person oder Institution angegeben, öffnet sich das Popup-Fenster für Agents:




work
Work
creator *
author: Dickens, Charles (http://d-nb.info/gnd/11)
+
-
uniform title*
► search for / edit uniform title
identifier
type
▼
date of creation*
approx. date of creation
switch to date range
not before*
1860
not before approx.
Ostern 1834
not after*
1862
not after approx.
back
type of work *
prose
▼
specific genre
Roman
keywords
subject
Gesellschaftskritik
type
time
Viktorianische Ära
type
thesaurus
time
19. Jahrhundert
type
place
London
type
Getty
abstract
Der Roman handelt von...
note
ok
cancel

WORK

1.2. XML-Editor

1.2.1. Produktübersicht XML-Editor

Der XML-Editor bildet eine der zentralen Komponenten des TextGrid-Clients: Er ist ein interaktives Tool, das zur Neuerfassung wie zur nachträglichen (manuellen) Annotation von Texten in XML-basierten Formaten dient. Darüber hinaus wird er mit anderen TextGrid-Tools im Eclipse-basierten User Interface *TextGridLab* integriert und kann so etwa von anderen Tools zur Anzeige von XML-Inhalten benutzt werden.

1.2.2. Zielbestimmung

Der Editor muss unterschiedlichen Benutzergruppen zur Erfüllung ihrer Aufgaben dienen: *Textwissenschaftlern* muss es möglich sein, bei Erfassung und Annotation den Überblick über den Text zu wahren und gerade Auszeichnungen wie Überschriften und Hervorhebungen mit den aus dem Druckbild gewohnten Mitteln visualisiert zu sehen. *Bearbeitern* innerhalb eines Projekts soll über entsprechende Schemata möglichst präzise vorgegeben werden, welche Annotationen mit welchen Mitteln möglich sein sollen, um die Konsistenz in ihrem Projekt zu wahren – und die verfügbaren Annotationsmöglichkeiten müssen den Benutzern natürlich vermittelt werden. *Techniker* wollen etwa für Konvertierungsaufgaben auch einen Blick auf die präzise Serialisierung des XML-Dokuments haben.

Eine detaillierte Aufstellung der zu implementierenden Funktionen ist im Abschnitt *Produktfunktionen* zu finden.

Musskriterien

Die wichtigste Funktion des XML-Editors ist es, Benutzer, die keine PC-Fachleute sind, in ihrer Arbeit mit XML-Texten möglichst weitgehend zu unterstützen. Ein wichtiges Element dieser Unterstützung ist die Möglichkeit, die visuelle Komplexität umfangreichen Markups zu reduzieren, indem in einem Dokument verschiedene *Darstellungsformen* unterstützt werden. Der Benutzer kann zwischen ihnen frei umschalten: Eine XML-Quellcodeansicht mit Syntaxhighlighting, eine Darstellung, in der die XML-Tags ausgeblendet und der Text mithilfe eines Stylesheets gestaltet wird und eine hybride Darstellung mit symbolischer Darstellung der Elemente und Ausblendung der Attribute. Der XML-Editor muss XML-Dokumente entsprechend einem in einer der gängigen XML-Schemasprachen (DTD, W3C XML Schema, Relax NG) formulierten *Schema* bearbeiten und validieren können. Der Benutzer soll bei seiner Arbeit möglichst weitgehend unterstützt werden, indem ihm angezeigt wird, welche XML-Elemente an einer bestimmten Position erlaubt sind und außerdem, welche Elemente er bereits häufiger verwendet hat. Hinzu kommt die Möglichkeit zum Auszeichnen vorhandenen Texts per Maus.

Darüber hinaus müssen dem Benutzer Möglichkeiten zur Orientierung und Navigation in seinem XML-Dokument gegeben werden.

Der Editor muss sich eng in das Eclipse-GUI-Framework *TextGridLab* mit den anderen in TextGrid entwickelten Tools einfügen.

Wunschkriterien

Die Darbietung der möglichen Elemente bzw. des möglichen Inhaltsmodells an der Cursorposition sollte möglichst visuell und einfach verständlich geschehen. Zudem sollten die oft umfangreichen Wahlmöglichkeiten durch Beschränkung zugänglicher gestaltet werden, etwa in dem die Elementauswahl auf bereits verwendete Elemente oder eine vorkonfigurierte Auswahl reduziert werden kann.

Der Benutzer sollte die Oberfläche möglichst nach seinen Präferenzen konfigurieren können und sich so etwa häufig benötigte XML-Elemente oder Unicode-Zeichen auf Symbolleisten bzw. Tastenkombinationen legen können.

Die Bearbeitung sehr großer XML-Dokumente, ohne dass diese zuvor mit anderen TextGrid-Mitteln wie den Streaming-Tools zerlegt werden müssen, ist ebenfalls wünschenswert.

Abgrenzungskriterien

- Die Verwaltung von Objekten im Repository sollte von den allgemeinen TextGridLab-Tools wie dem Navigator übernommen werden (AP2).

- Für die automatisierte Transformation von XML-Daten existieren die Streaming-Tools, insbesondere der Streaming-Editor.

- Die Erzeugung von XML-Schemata ist nicht Aufgabe des XML-Editors. Hierfür existieren spezielle Werkzeuge, für TEI-Anpassungen etwa das TEI-Tool *Roma*[1]. Zur Bearbeitung von XML-Schemata im TextGridLab wird der XML-Schema-Editor aus dem Eclipse-Webtools-Projekt[2] in das TextGridLab eingebunden.

- Masken zur Eingabe von bereichsspezifischen, stark strukturierten Angaben wie etwa Metadaten, werden mit dem Metadateneditor definiert.

1.2.3. Produkteinsatz

Der XML-Editor wird vorwiegend zur Transkription existierender Quellen, zur Annotation vorhandener Texte, zum Verfassen neuer Texte und zur Vorschau automatisierter Bearbeitungen genutzt:

Transkription existierender Quellen

Existierende Quellen wie etwa Handschriften oder Drucke werden von Editoren oder Bearbeitern mithilfe des XML-Editors erfasst. Die Vorlagen können dabei sowohl bereits als Digitalisate als auch nur in originaler Form als physische Vorlage vorliegen. In letzterem Fall ist zu erwarten, dass die Transkription möglicherweise mit dem Laptop vor Ort in einem Archiv durchgeführt werden muss, aus dem das Original nicht entfernt werden darf.

Auszeichnung, Bearbeitung und Kommentierung vorhandener Texte

Bereits in XML- oder (digitaler) Textform vorliegende Dokumente werden vom Editor oder Bearbeiter mit zusätzlichen Annotationen versehen, um Kommentare ergänzt, bearbeitet und neu arrangiert. Vorhandene Annotationen werden bearbeitet.

---

[1] http://tei.oucs.ox.ac.uk/Roma/

[2] Webtools-XSD-Editor-Link: TODO

Verfassung neuer Texte

Originäre Texte des Editors werden erfasst und ausgezeichnet – etwa editorische Notizen oder einleitende Kapitel.

Einsatz in Komponenten wie dem Linkeditor

Mit Hilfe des Text-Bild-Linkeditors (vgl. Abschnitt 2.1) können Links zwischen Digitalisaten in Bildform und deren TEI-Transkriptionen erzeugt werden. Der XML-Editor dient hier der Anzeige der Transkription und muss eine Möglichkeit bieten, um darin Links zu markieren.

Vorschau automatisierter Bearbeitungen

Editoren, Bearbeiter und technische Betreuer benutzen Streaming-Tools wie den Lemmatisierer oder den Streaming-Editor zur automatischen Bearbeitung von XML-Dokumenten. Der XML-Editor wird als Komponente zur Darstellung und manuellen Überprüfung der Ergebnisse dieser Tools verwendet.

1.2.4. Produktfunktionen

Unterstützung von XML- und Schema-Standards

Gängige XML-Standards sollen unterstützt werden. Dies betrifft einerseits die Validierung von XML-Dokumenten (bzw. die Überprüfung auf Wohlgeformtheit), andererseits die in Abschnitt 3.6.4.5 (Unterstützung bei der Eingabe von Auszeichnungen) dargestellten schemaabhängigen Funktionen.

Die zu unterstützenden Standards umfassen:

- XML
- Die Schema-Sprachen
  - Relax NG[3]
  - W3C XML Schema
  - DTDs

Einbindung in das TextGridLab

- Enge Integration in das sonstige TextGrid-Framework und insbesondere in das *TextGridLab*
- Nutzung des XML-Editors zur Vorschau und Ergebnisanzeige für Input und Output der TextGrid-Webservices, z. B. Tokenizer, Lemmatisierer oder Streaming-Editor
- Aufruf der Benutzungsschnittstellen der Streaming-Tools aus dem XML-Editor heraus

Visualisierung des XML-Dokuments

Anzeigemodi

Der Benutzer kann zwischen verschiedenen Anzeigemodi umschalten:

[3] Da Relax NG einerseits automatisch in W3C XML Schema übersetzt werden kann und andererseits eine Relax-NG-Unterstützung für den dem TextGrid-XML-Editor zugrundeliegenden WTP-XML-Editor in Arbeit ist, werden eigene Entwicklungen zu Relax NG in TextGrid zunächst zurückgestellt

- WYSIWYM[4]-Darstellung, in der Auszeichnungen durch aus dem Druckbild bekannte typografische Eigenschaften visualisiert werden (vgl. Abbildung 1). Die Konfiguration erfolgt mit *Cascading Style Sheets* (CSS), für gängige TEI-Konstrukte werden Stylesheets mitgeliefert. Liefert eine grundlegende Übersicht über den Text.

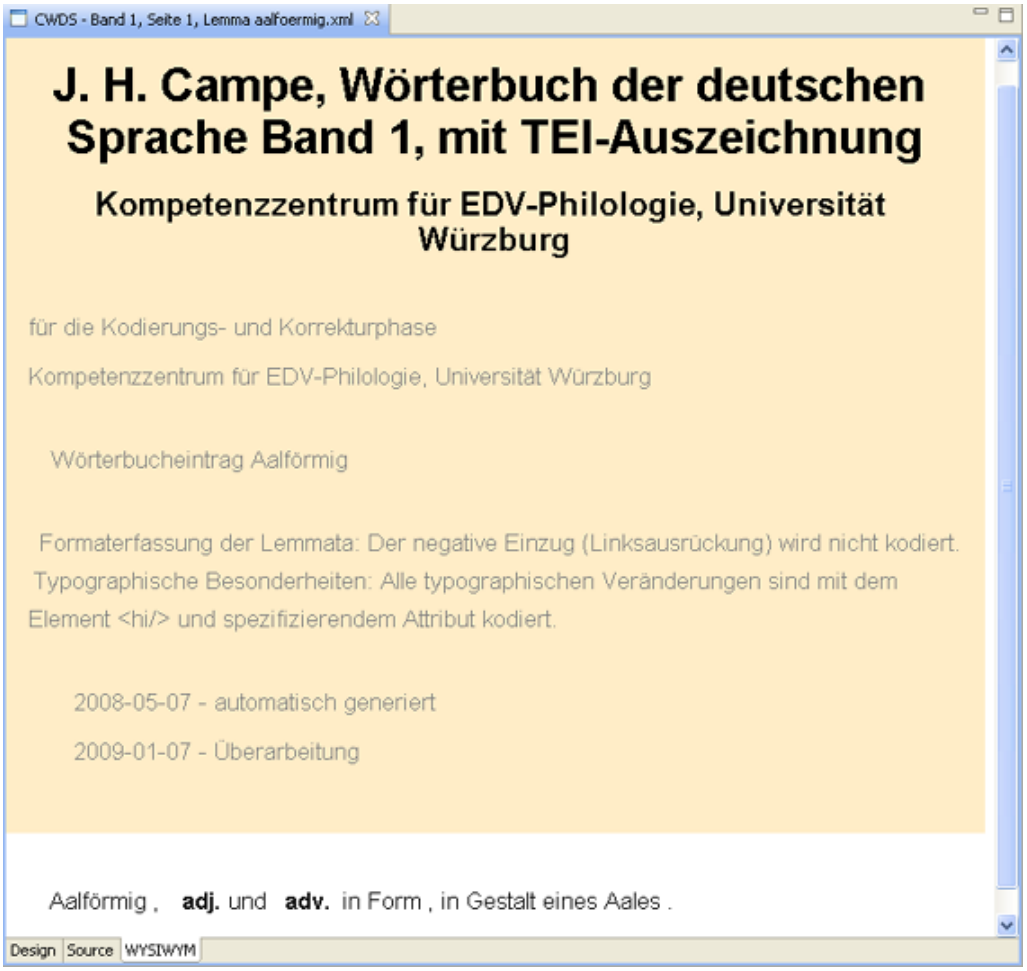


Abbildung 1: Beispiel für WYSIWYM-Darstellung

- WYSIWYM-Darstellung mit einer zusätzlichen symbolischen Visualisierung aller Auszeichnungselemente, vgl. Abbildung 2. Zur Durchführung und Kontrolle der Auszeichnungen, ohne den Überblick über den Text im Spitzklammerwald zu verlieren.

[4] What You Mean Is What You Get – eine Darstellung, die sich an einem möglichen Druckbild orientiert, aber (im Gegensatz zu WYSIWYG: What You *See* Is What You Get) statt auf eine 1:1-Umsetzung desselben eher auf eine sinnvolle Darstellung im Programmkontext zielt.

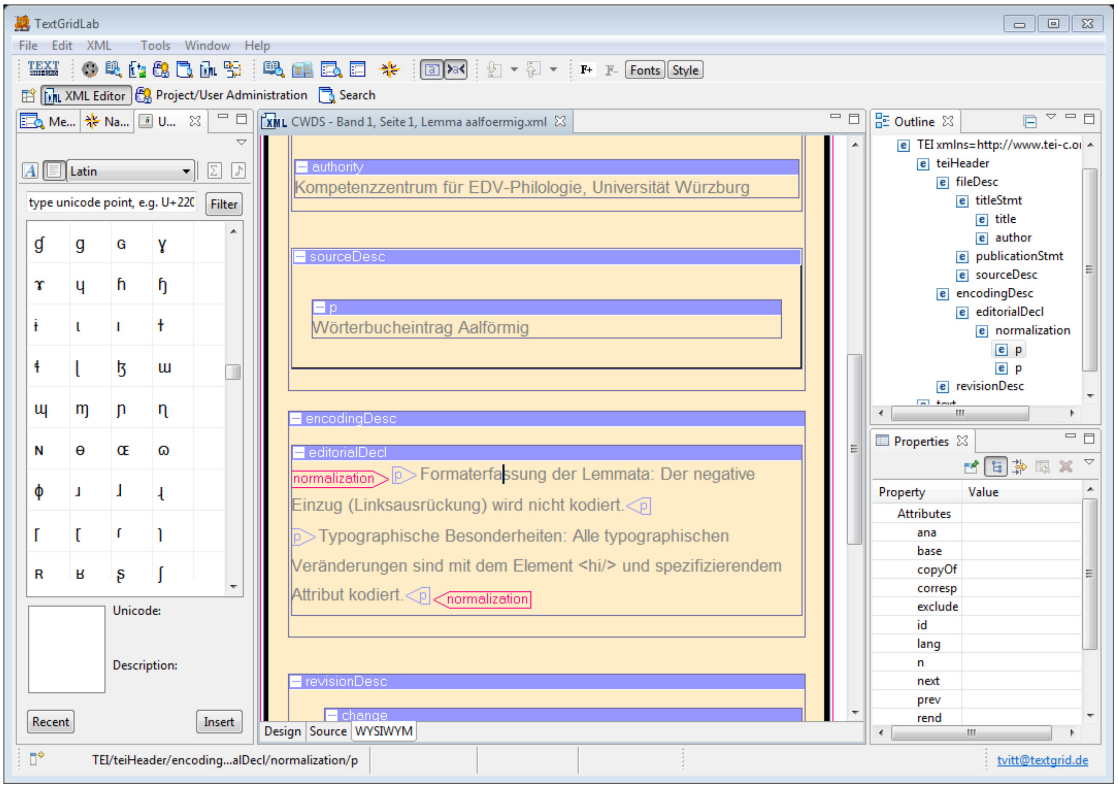


Abbildung 2: WYSIWYM-Darstellung mit symbolischer Visualisierung der Auszeichnung

- XML-Code mit Syntaxhighlighting. Zur präzisen Kontrolle über technische Details des XML-Dokuments.

weitere Visualisierungsfunktionen

- Anzeige von Tabellen im WYSIWYM-Modus
- Folding: Der Benutzer sollte unerwünschte Teilstrukturen des Dokuments (etwa bestimmte Abschnitte) ausblenden können, um sich auf den aktuellen Teilbereich zu konzentrieren (Wunschkriterium)
- Elementausblendung: Der Benutzer sollte unerwünschte Elemente des Dokuments (etwa mit dem Text des Editors, nicht des Autors, in einer Edition) mitsamt ihrem Inhalt ausblenden lassen können (Wunschkriterium)
- Anzeige von Bildern im WYSIWYM-Modus. Die Darstellung wird über CSS geregelt.

Unterstützung bei der Orientierung im Dokument

- Gliederungsansicht: Der Benutzer sollte bestimmte XML-Elemente als Gliederungselemente festlegen können. Daraus wird dann eine Gliederungsansicht generiert, mit der der Benutzer durch das Dokument navigieren kann (siehe die Ansicht rechts unten in Abbildung 2).
- Elementhierarchie: Die aktuelle Position in der Elementhierarchie des Dokuments sollte anzeigbar sein (etwa als *TEI/text/body/div/p*)
- XPointer/XPath: Die aktuelle Position im Element sollte angezeigt werden, etwa in einer XPointer- bzw. XPath-artigen Struktur wie *TEI/text/body/div[4]/p[8]*.

Unterstützung bei der Eingabe von Auszeichnungen

- Die an der Cursorposition entsprechend dem Schema möglichen Elemente und sonstigen Inhalte sollten dem Benutzer dargeboten werden. Das umfasst:

  - Eine Anzeige der an der Cursorposition bzw. für den markierten XML-Abschnitt erlaubten Elemente mit der Möglichkeit, das Element an der aktuellen Position bzw. um die aktuelle Markierung herum einzufügen.
  - Die Anzeige und Bearbeitung der vorhandenen und möglichen Attribute zum aktuellen Element.
  - Textauszeichnung durch Markieren des Texts und Auswählen des Auszeichnungselements.
- Die möglichen Elemente sollten auch mit der Tastatur in effizienter Art und Weise eingefügt werden können. Dies wird realisiert durch
  - Tag Completion in der Quellcodeansicht für alle erlaubten öffnenden Tags (mit Beschränkung auf diejenigen, deren Anfang bereits eingegeben wurde) bzw. die schließenden Tags aller offenen Elemente (mit automatischem Einfügen der »übersprungenen« schließenden Tags)
  - Eine vergleichbare einfache Elementauswahl an der Cursorposition in den anderen Ansichten, ebenfalls mit Unterstützung für das Eintippen eines Präfixes
- »Umwandlung« von Elementen, ohne dass der sonstige Text geändert wird. Realisierung zum Beispiel: Rechtsklick auf Tag → ›Tag ersetzen‹ im Kontextmenü wählen → Auswahlliste erlaubter Tags → Ersetzen von Anfangs- und Endtag.
- Kontextabhängige Einblendung der Dokumentation für XML-Elemente, wie sie beispielsweise aus den Dokumentationselementen der Schemata bzw. dem ODD-Schema[5] abgeleitet werden können, in den Hilfen zur Elementauswahl bzw. in einem eigenen Programmbereich.
- Beschränkung der Liste der möglichen Elemente auf die bereits im Dokument verwendeten
- Automatische Einfügung der minimalen durch das Schema vorgegebenen Grundstruktur
- Möglichkeit zur Zusammenstellung ›beliebter‹ Elemente, ggf. mit Attributen vorkonfiguriert, etwa auf einer Symbolleiste; Zuordnung zu Tastenkombinationen

Wunschkriterien umfassen:

- Visualisierung des Modells an der Cursorposition auf einfach verständliche Weise, etwa als Railroad-Diagramm (vgl. Abbildung 3)

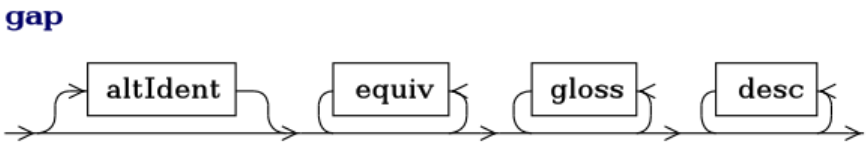


Abbildung 3: Das TEI-Element `gap` als Railroad-Diagramm.
(Quelle: Barry Cornelius, Sebastian Rahtz: *TV – A TEI Visualizer*)

Unterstützung für Sonderzeichen im Unicode

- Der Editor muss mit beliebigen Unicode-Zeichen umgehen können. Zeichen, die nicht dargestellt werden können, müssen dennoch erhalten bleiben.
- Für Sonderzeichen, die nicht in gängigen Tastaturbelegungen vorhanden sind, sollte es Eingabehilfen geben. Das umfasst

[5] http://www.tei-c.org.uk/wiki/index.php/ODD

– konfigurierbare Tastenkombination
– konfigurierbare Symbolleisten
– einen Auswahldialog bzw. eine Auswahlliste

Dabei sind insbesondere auch die *Private Use Areas* von Unicode zu berücksichtigen.

Textverarbeitungsfunktionen

Folgende übliche Textverarbeitungsfunktionen werden in den Editor integriert:

- Ausschneiden, löschen, Kopieren, Einfügen
- Suchen, Ersetzen (Unterstützung regulärer Ausdrücke) (Optionen: Richtung oben/unten/gesamt; Einschränkung auf bestimmte Zeilen;
- Suchen mithilfe von XPath-Ausdrücken

1.2.5. Produktdaten

Welche Daten sind aus Benutzersicht (langfristig) zu speichern

Der XML-Editor soll dem Bearbeiten und natürlich auch dem Speichern beliebiger XML-Dokumente dienen, eine besondere Unterstützung für TEI-konforme Dokumente ist sinnvoll. Darüberhinaus ist ggf. die Zuordnung von Dokumenten oder Namespaces zu Schemata und Stylesheets (für den visuellen Modus des XML-Editors) zu speichern, ebenso Anpassungen der Arbeitsumgebung durch den Benutzer, soweit dies nicht durch allgemeine Komponenten des TextGridLab geschieht, sowie die Zuordnung zum Baseline-Encoding-Adapter[6].

Wo liegen die Daten, von wo aus greift die Applikation darauf zu?

Die Daten liegen im TextGrid-Repository *TextGridRep* oder lokal vor. Im *TextGridRep* können sie über die Middleware-Schnittstelle TG-crud und deren TextGridLab-weiten Client angesprochen werden. Für die Offline-Arbeit z. B. in Archiven ohne Internetzugang muss es auch möglich sein, Daten aus dem Grid lokal zwischenzuspeichern und später wieder ins Grid zu übertragen oder lokal erfasste Daten ins Grid einzuspielen. Diese Funktionalität sollte für alle Tools zentral im TextGridLab implementiert werden.

Welche Datenmengen müssen verarbeitet werden

Grundsätzlich ist die Bearbeitung sehr großer XML-Dokumente wünschenswert. Als Muss-Kriterium ist jedoch zunächst nur die Arbeit mit Daten im Umfang von mehreren Megabyte umzusetzen, größere Dokumente können ggf. vor der Bearbeitung mithilfe der Streaming-Werkzeuge zerlegt werden.

1.2.6. Produktleistungen

Beim Bearbeiten bestehender XML-Dateien sollten möglichst die Eigenschaften gemäß den XML-Datenmodellen *XML Information Set* und *XQuery 1.0 and XPath 2.0 Data Model* erhalten bleiben. Zudem sind (auch wenn die Darstellung von der Verfügbarkeit entsprechender Schriften abhängt) Unicode-Zeichen auch jenseits der Basic Multilingual Plane zu erhalten.

[6] Zum Baseline Encoding vgl Blümm et. al.: *Die textsortenspezifische Kernkodierung für Dokumente in TEI* P5, insbesondere das Einführungskapitel (verfügbar von http://www.textgrid.de/berichte.html)

1.2.7. Benutzeroberfläche

Die Benutzeroberfläche sollte sich an gängigen Usability-Kriterien orientieren (vgl. etwa ISO 9241, Teil 10) und für Textwissenschaftler, Bearbeiter und technische Betreuer unterschiedlicher Computerexpertise benutzbar sein. Eine Anlehnung der GUI an die von der Textverarbeitung her bekannten Oberflächen wird angestrebt, soweit dies mit der spezifischen Funktionalität eines XML-Editors vereinbar ist. Eine enge Integration in das *TextGridLab* und mit den anderen TextGrid-Tools ist verpflichtend.

1.2.8. Nichtfunktionale Anforderungen

Einzuhaltende Gesetze und Normen, Sicherheitsanforderungen, Plattformabhängigkeiten

Die Software soll die gängigen Standards im XML-Bereich nicht verletzen und Unicode-Daten berücksichtigen. Sie muss auf allen Plattformen laufen, auf denen das *TextGridLab* läuft, mindestens auf aktuellen Linux-, Macintosh- und Windows-Plattformen.

Welche Nutzungsdaten sollen (oder dürfen nicht) von der Middleware erhoben werden

Nicht anwendbar.

Technische Produktumgebung

Software

Der Client wird im Rahmen des *TextGridLab* auf der Basis von Eclipse implementiert. Eine Serverkomponente ist nicht vorhanden.

Hardware

Es gelten die allgemeinen Anforderungen des TextGridLab. Aus Performancegründen ist einigermaßen aktuelle Hardware wünschenswert.

Produktschnittstellen

Der XML-Editor implementiert, soweit vorhanden, die Eclipse- bzw. TextGridLab-eigenen Schnittstellen, um eine möglichst enge Integration in das Lab bzw. mit anderen auf dieser Plattform aufbauenden Tools zu realisieren.

1.3. Linkeditor Text

1.3.1. Produktübersicht Link-Editor Text

Der Link-Editor Text dient als Eingabehilfe für Links in XML-Dateien und verbindet Elemente von Recherchetool und XML-Editor. Benutzer haben die Möglichkeit, Links zu beliebigen Elementen in TextGrid-Dokumenten in der aktuellen TEI-Datei zu erfassen. Für das Quelldokument des Links wird anschließend die passende URI samt Fragment generiert.

Darüber hinaus wird der Link-Editor Text zur Validierung bestehender Links eingesetzt.

1.3.2. Zielbestimmung

Der Link-Editor Text muss in der Lage sein, Link-Ziele aus unterschiedlichen Quellen übernehmen zu können, hierzu zählen das Recherchetool, der Projektbrowser, die XML-Outline-Darstellung, aber auch die manuelle Eingabe im XML-Editor. Die Datenübernahme soll für den Nutzer möglichst bequem sein, z.B. mittels Kontext-Menü oder Drag 'n' Drop.

Musskriterien

Die Hauptfunktion des Link-Editors Text ist es, Anwendern ohne tiefgreifende Computerkenntnisse eine einfache Möglichkeit zu bieten, Textfragmente auszuwählen um sie in ihrer Arbeit zu referenzieren. Das wichtigste Element ist hierbei die visuelle Unterstützung der User. Textfragmente sollen im Originaldokument möglichst einfach ausgewählt, übernommen und bei Bedarf zu einem späteren Zeitpunkt wieder angezeigt werden können. Das Editieren erstellter Links muss jederzeit möglich sein.

Mittels einer Undo-Funktion sollen Änderungen an Links bis zur nächsten Speicherung des Dokumentes rückgängig gemacht werden können.

Wunschkriterien

Das Suchen nach ähnlichen Textfragmenten im gleichen Dokument mittels unterschiedlicher Suchalgorithmen wäre wünschenswert. Ebenfalls wünschenswert wäre eine solche Suche über mehrere Textdokumente hinweg.

Falls die verlinkten Textdokumente geändert werden, wäre es wünschenswert, wenn mittels einer Ähnlichkeitsanalyse festgestellt werden könnte in welchem Maß sich der Text geändert hat, um unterhalb spezifischer Schwellenwerte die Verlinkungen beibehalten zu können.

1.3.3. Produktleistung

Beim Erstellen von Links auf Textdokumente dürfen die Originaldokumente nicht verändert werden. Es muss aber trotzdem sichergestellt werden, dass Links auf ihre Validität hin überprüft werden. Wenn der Inhalt eines verlinkten Fragments vom Eigentümer geändert wird muss der entsprechende Link mit einer Fehlermeldung markiert werden. Hierbei sollte resourcensparend mit Hash-Summen zu arbeiten.

1.3.4. Benutzeroberfläche

Die Benutzeroberfläche sollte sich nahtlos in das Look 'n' Feel des TextGridLab integrieren. Auch sollte die grafische Funktionalität sich an bereits vorhandenen Komponenten, wie z.B. dem Link-Editor Bild orientieren.

1.3.5. Nichtfunktionale Anforderungen

Einzuhaltende Gesetze und Normen, Sicherheitsanforderungen, Plattformunabhängigkeiten

Die Software soll die gängigen Standards im XML-Bereich nicht verletzen und Unicode-Daten berücksichtigen. Sie muss auf allen Plattformen laufen, auf denen die *Rich Client Platform* läuft, mindestens auf aktuellen Linux-, Macintosh- und Windows-Plattformen.

Welche Nutzungsdaten sollen (oder dürfen nicht) von der Middleware erhoben werden

Es handelt sich hierbei um keine Middleware-Komponente.

1.3.6. Technische Produktumgebung

Software

Der Client wird im Rahmen des *TextGridLab* auf der Basis von Eclipse implementiert. Eine Serverkomponente ist nicht vorhanden.

Hardware

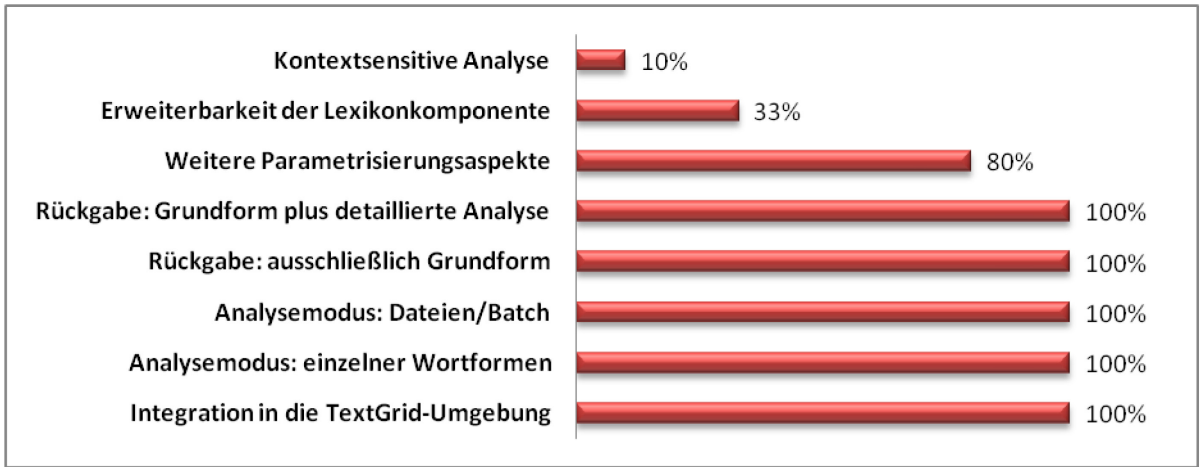
Es gelten die Anforderungen des *TextGridLAb*. Aus Performancegründen ist einigermaßen aktuelle Hardware wünschenswert.

Produktschnittstellen

Der Link-Editor Text implementiert, soweit vorhanden, die Eclipse- bzw. RCP-eigenen Schnittstellen, um eine möglichst enge Integration in die *Rich Client Platform* bzw. mit anderen auf dieser Plattform aufbauenden Tools zu realisieren.

1.4. Lemmatisierung

1.4.1. Tabelle: Muss- u. Wunschmerkmale



1.4.2. Produktübersicht

Das Lemmatisierer-Modul „Morphisto“ hat die morphologische Analyse hochdeutscher Wortformen zum Gegenstand. Es können ganze Dateien oder einzelne Wortformen analysiert werden, wobei Information zur jeweiligen Grundform (Lemma) sowie ggfs. auch zur Wortart und weiteren morpho-syntaktischen Merkmalen ausgegeben werden kann. Die technische Umsetzung basiert auf den frei verfügbaren SFST/SMOR-Tools der Universität Stuttgart. Für diese wurde eine freie Lexikonkomponente entwickelt, deren Abdeckung sich bezüglich der berücksichtigten Lemmata an der DeReWo-Liste der 30.000 häufigsten Wörter des DeReKo-Korpus (beides: IDS, Mannheim) orientiert. Die tatsächliche Abdeckung liegt jedoch, aufgrund der via SMOR implementierten morphotaktischen und orthographischen Regeln des Hochdeutschen, deutlich höher.

1.4.3. Zielbestimmung

Das Modul dient der Lemmatierung geschriebener Wortformen des Hochdeutschen. Lemmatisierung kann hierbei ausschließlich die Rückführung auf eine Grund- bzw. Zitierform meinen oder aber die Angabe weiterer morphosyntaktischer Merkmale einschließen. Der erstgenannte Punkt kann als erster Schritt bezüglich einer Wörterbuchabfrage, einer Verschlagwortung oder effizienten Indexerstellung für Zwecke des Information Retrieval gesehen werden. Der zweitgenannte Punkt ist insbesondere für linguistische Analysen, auch im Sinne einer entsprechend granularen Annotation von Textdaten, von besonderem Interesse.

Musskriterien

Es müssen einzelne Wortformen (interaktiv) sowie ganze Dateien, die tokenisierte Wortformen enthalten (Batch-Modus), lemmatisiert werden können.

Die Lemmatisierung muss hinsichtlich verschiedener Parameter konfigurierbar sein: Der wichtigste Punkt betrifft im Sinne der Vorüberlegungen zur Zielbestimmung die Auswahl, ob im Einzelfall ausschließlich eine Rückführung auf die Grundform gewünscht ist oder, darüber hinaus, auch eine detaillierte morpho-syntaktische Analyse ausgegeben werden soll.

Wunschkriterien

Hinsichtlich der genannten Parametrisierung ist es wünschenswert, weitere Aspekte des implementierten Moduls ebenfalls nach Bedarf an- bzw. abstellen zu können: Hierzu zählen die unscharfe Suche bzgl. Umlauten, ein Nachbearbeitungsschritt bzgl. Desambiguierung, eine Guesser-Komponente für unbekannte Wortformen und die Möglichkeit einer Datenkomprimierung für den Output. Außerdem soll die Möglichkeit bestehen, sich auf eine Analyse im Rahmen neuer deutscher Rechtschreibung oder alter deutscher Rechtschreibung beschränken zu können.

Es wäre ferner wünschenswert, die Lexikonkomponente bei Bedarf um projektspezische Wörterbücher erweitern zu können. Diesbezüglich gilt es jedoch zu bedenken, dass jede Veränderung bezüglich des Lexikons eine Neukompilierung der SFST/SMOR-Transduktoren nötig macht.

Darüber hinaus wäre es erstrebenswert, dem aktuellen Analysemechanismus einen kontextsensitiv arbeitenden Apparat zur Seite zu stellen, der die jeweils benachbarten Wörter bzw. den Satzkontext bei der Analyse berücksichtigt. Im Rahmen eines solchen, vermutlich probabilistisch zu basierenden Ansatzes, sollte es möglich sein, die jeweiligen Wahrscheinlichkeiten für Analysevarianten anzugeben oder, falls gewünscht, ausschließlich die jeweils wahrscheinlichste Variante als Ergebnis zurückzugeben.

Abgrenzungskriterien

Tokenisierung, wie sie etwa durch den TextGrid-„Tokenizer“ geleistet wird, kann als eine Vorbedingung für den Prozess der Lemmatisierung, zumindest hinsichtlich des Batch-Modus im Sinne der Analyse ganzer Dateien, angesehen werden, ist aber selbst nicht Gegenstand des hier beschriebenen Moduls.

Lemmatisierung kann seinerseits als eine Vorbedingung für die Abfrage/Suche in Wörterbüchern verstanden werden, wie sie innerhalb des TextGridLab durch das eigene „Dictionary Search“-Modul zur Verfügung gestellt wird.

1.4.4. Produktfunktionen und -daten

Als wesentliche Produktfunktionen sind der für einzelne Wortformen aufzurufende Analysemodus (*Lemmatize Wordform*, Input: Eingabewort, Output: Analysestring) und der Batchmodus (*Lemmatize File*, Input: Eingabedatei, Output: Analysedatei) für Dateien mit tokenisierten Wortformen zu unterscheiden. Entlang einer weiteren Dimension muss bezüglich der Analyse unterschieden werden, ob ausschließlich die jeweilige Grundform zurückgegeben (*Get Wordform*) oder zusätzlich morphosyntaktische Information beigefügt werden soll (*Get Analysis*). Die Daten können als einzelne Wortformen innerhalb von TextGrid-Objekten oder aber lokal als mittels UTF-8 kodierter plain text oder in tokenisiertem TEI/XML-Format (ebenfalls UTF-8 kodiert) als Dateien vorliegen.

1.4.5. Produktleistungen

Die Ausgabe erfolgt in einem toolspezifischen Fenster, vgl. hierzu 5. unten („Benutzeroberfläche“), wobei diese gemäß der jeweils gewählten Variante (*Get Wordform* oder *Get Analysis*) ausschließlich als Grundform(en) oder einschließlich der tieferen morphologischen Analyse erfolgt. Im letztgenannten Fall erfolgt die Ausgabe gemäß einer SFST/SMOR-spezifischen Syntax, die XML-ähnliche Marken. Im Batchmodus steht zusätzlich zum auch im *Lemmatize Wordform* – Modus verwendeten Ausgabeformat die Möglichkeit bereit, das Resultat als TEI-XML auszugeben und es in einer lokalen Datei abzulegen.

1.4.6. Benutzeroberfläche

Das „Lemmatizer“-Modul kann über den Punkt „Tools“ im Hauptmenü des TextGridLab ausgewählt werden. Anschließend muss innerhalb des aufklappenden Untermenüs eine Auswahl zwischen den Funktionen „Lemmatize Wordform“ (einzelne Wortform, interaktiv) und „Lemmatize File“ (Datei mit tokenisierten Wortformen, Batch-Modus) getroffen werden, woraufhin ein toolspezifisches Fenster geöffnet wird, welches die weitere Interaktion mit dem Benutzer, etwa hinsichtlich des zu analysierenden Terms bzw. der Datei, der Kodierung, den Details der Konfiguration sowie schließlich bezüglich der zurückzugegebenden Analyse, zur Darstellung bringt.

Alternativ kann der Lemmatisierer, ausschließlich im Sinne der Funktionalität „Lemmatize Wordform“, auch mittels eines eigenen Icons in der Tool-Leiste angesprochen werden.

Eine dritte Möglichkeit, wiederum nur für die Analyse einzelner Wortformen, nicht also im Batch-Modus, stellt der Aufruf mittels Kontextmenü für ein markiertes Wort dar. Hierbei muss „Lemmatizer“ gewählt werden, woraufhin in einem aufklappenden Untermenü die Varianten „Get Wordform“ und „Get Analysis“ direkt zur Auswahl stehen. Ersteres entspricht ausschließlich der Rückgabe einer Grundform, während letzteres auch eine morphosyntaktische Analyse umfasst.

Die Parametrisierung hinsichtlich dieser Funktionalität wird ansonsten innerhalb des spezifischen Toolfensters entweder durch Direkteingabe in ein dafür vorgesehenes Feld mittels eines spezifischen XML-Formats, oder, deutlich benutzerfreundlicher, nach Auswahl des Punktes „Make Other Configuration“ durch Auswahl der gewünschten Punkte in einer sich öffnenden Maske erreicht. Im interaktiven Modus besteht darüber hinaus in Form des Feldes „German Wordform“ die Möglichkeit der Eingabe bzw. Änderung der zu analysierenden Wortform, während im Batchmodus die Funktion „Specify Input File“ bzw. ein Feld zur Eingabe des Pfades zur entsprechenden Datei bereitstehen. Zusätzlich kann im Batchmodus das Format der Inputdatei ausgewählt werden. Gemeinsam ist beiden Modi (in den entsprechenden Tabs „Lemmatizer“ bzw. „Batchlemmatizer“ betitelt), dass der Lemmatisierungsprozess schließlich mittels „Start Lemmatizer!“ in Gang gesetzt wird und dessen Resultate in einem vergleichsweise großen Resultatfeld im unteren Bereich dargestellt werden.

1.4.7. Nichtfunktionale Anforderungen

Einzuhaltende Gesetze und Normen, Sicherheitsanforderungen, Plattformabhängigkeiten

Plattformunabhängig, da als Web Service implementiert.

Welche Nutzungsdaten sollen (oder dürfen nicht) von der Middleware erhoben werden

Logging der Suchanfragen und Benutzerdaten.

1.4.8. Technische Produktumgebung

Software

Server:

Apache 2.0.x mod_python,
Python 2.4.x mit zusätzlichen Paketen:

- ZSI-2.0_rc3 (http://pywebsvcs.sourceforge.net)
- PyXML 0.8.4 (http://pyxml.sourceforge.net)
- SOAPpy 0.11.6 (http://sourceforge.net/project/showfiles.php?group_id=26590&package_id=18246)
- fpconst 0.7.2 (http://cheeseshop.python.org/pypi/fpconst/0.7.2)
- SFST Tools (http://www.ims.unistuttgart.de/projekte/gramotron/SOFTWARE/SFST.html)
- pysfst (http://home.gna.org/pysfst/)

Hardware

Für Server und Client getrennt: Entsprechend den allgemeinen Projektvorgaben.

Produktschnittstellen

Die Modul-spezifischen Web Services für die interaktive Analyse und den Batchmodus wurden via SOAP implementiert und stehen auf http://ingrid.sub.uni-goettingen.de bereit.

1.5. Sortieren

Das Sortiertool ist seit der ersten Projektphase als Webservice verfügbar und kann über das Workflowtool angesprochen werden. Dokumentation kann in TextGrid-Report 2.3 (aus der ersten Projektphase) gefunden werden.

1.5.1. Produktübersicht

Anordnung von gegebenen Zeichenketten gemäß kulturellen und fachlichen Erwartungen. Transformation ungeordneter Zeichenketten in geordnete. Konventionen vom Nutzer vorgebbar. Bekannte nationale und europäische Standards, etwa DIN 5007, NFZ44-001 (AFNOR), TK 34.1 (SIS) und ENV 13710 (CEN) werden direkt unterstützt.

1.5.2. Zielbestimmung

Wenn ein Textdokument - beispielsweise ein Index oder eine Sammlung von Wörterbuchartikeln - sortiert werden soll, so muss das entsprechende Tool zunächst die Möglichkeit bieten, Sortierfelder und -schlüssel zu definieren, Stoppwörter anzugeben, die bei der Sortierung ignoriert werden, und nicht zuletzt die gewünschte Sortierreihenfolge festzulegen.

Die in vielen Editionsprojekten bewährten Module #SORTIERE bzw. #SVORBEREITE aus TUSTEP verlangen, dass das Sortieralphabet für jeden Schlüssel explizit angegeben wird. Dies ist angesichts der Fülle der in Unicode zur Verfügung stehenden Zeichen nicht mehr zeitgemäß; von einem modernen Sortierer kann man erwarten, dass die Auswahl des Sortieralphabetes durch die Angabe eines Kulturraumes möglich ist; genauer gesagt durch die Benennung der Lokale, welche die Sortiergepflogenheiten des entsprechenden Kulturraumes wie in den einschlägigen Standards festgelegt beschreibt. Weil es immer noch Sonderfälle oder in den Standards nicht erfasste Sonderzeichen geben kann, muss es ferner möglich sein, die durch die Lokale gegebenen Sortierregeln zu ergänzen oder modifizieren.

Musskriterien

Anordnung von gegebenen Zeichenketten gemäß kulturellen und fachlichen Erwartungen. Transformation ungeordneter Zeichenketten in geordnete. Konventionen vom Nutzer vorgebbar. Bekannte nationale und europäische Standards, etwa DIN 5007, NFZ44-001 (AFNOR), TK 34.1 (SIS) und ENV 13710 (CEN) werden direkt unterstützt.

Bestimmte Wörter können durch Angabe einer Stoppwortliste aus der zu sortierenden Liste herausgefiltert (sort), bzw auch mehrfach vorkommende Wörter gestrichen werden (uniq).

1.6. Streaming-Editor

Produktübersicht

Der Streaming Editor I ist ein WebService-Wrapper um einen XSLT[7]-Prozessor.

[7] http://www.w3.org/Style/XSL/

Zielbestimmung

Musskriterien

Regelbasierte Transformation eines XML-Objekts anhand eines XSLT-Stylesheets.

Wunschkriterien

XSLT 2.0 bietet die Möglichkeit, sowohl mehrere Dateien einzulesen, als auch in mehrere Ausgabedateien zu schreiben. Der Streaming-Editor sollte dies für TextGrid-Objekte und deren Metadaten unterstützen, indem Zugriffe auf TG-crud entsprechend unterstützt werden.

Abgrenzungskriterien

Die XSLT-Implementierung Saxon 9 ermöglicht es, beliebige Java-Klassen auszuführen[8]. Dieses Feature muss aus Sicherheitsgründen deaktiviert werden.

Produkteinsatz

Streaming Tool, als Web Service gekapselt. Keine Grid-Features; diese werden von *TG-crud* zur Verfügung gestellt.

Produktfunktionen

Transformation von XML-Daten anhand eines XSLT-Stylesheets.

Die zu transformierenden Daten können ebenso wie das für die Transformation zuständige XSLT-Stylesheet als URIs adressiert werden. In diesem Fall sorgt eine Schnittstelle zu *TG-crud* für das Einlesen der Daten aus dem Grid – und generiert ein Objekt, in das das Ergebnis der Transformation geschrieben wird. Wie bei allen Tools, die auf Daten im Grid zugreifen, müssen auch hier die Zugriffsrechte berücksichtigt werden.

Daneben existiert eine Schnittstelle, die die zu verarbeitenden Daten aus den Parametern der SOAP-Message ausliest und das Ergebnis über die SOAP-Response direkt zurückschreibt.

Eine weitere Schnittstelle liest und schreibt die Daten base64-codiert aus der und in die SOAP-Message.

Produktdaten

Welche Daten sind aus Benutzersicht (langfristig) zu speichern

Quelle, Ziel, XSLT-Stylesheet; dies geschieht jedoch i.A. nicht durch den Streaming-Editor selbst sondern durch entsprechende Infrastrukturkomponenten.

Wo liegen die Daten, von wo aus greift die Applikation darauf zu

Die Daten liegen im TextGridRep oder können über den SOAP-Request mitgegeben werden.

Welche Datenmengen müssen verarbeitet werden

Beliebig große Datenmengen.

---

[8] http://www.saxonica.com/documentation/extensibility/functions.html

Daten-Format

Eingabedaten: XML, Ausgabedaten: beliebig, abhängig von XSLT-Stylesheet.

2.2.6 Produktleistungen

XML-Daten sollten 1:1 weitergereicht werden. Da die in die SOAP-Messages eingebetten XML-Daten seitens der client- und serverseitigen Frameworks hin- und her-escaped werden, besteht die Gefahr, dass bestimmte Sonderzeichen falsch interpretiert werden – deshalb auch die WebService-Instanz, die base64-codierte Daten liest und schreibt.

2.2.7 Benutzeroberfläche

Über die Workflowkomponente.

2.2.8 Nichtfunktionale Anforderungen

2.2.8.1 Einzuhaltende Gesetze und Normen, Sicherheitsanforderungen, Plattformabhängigkeiten

Plattformunabhängig, da als WebService implementiert.

2.2.8.2 Welche Nutzungsdaten sollen (oder dürfen nicht) von der Middleware erhoben werden

Logging, Durchreichen von Benutzerdaten, siehe Report 3.2 aus der ersten Projektphase: „TextGrid Architektur“.

2.2.9 Technische Produktumgebung

2.2.9.1 Software: für Server und Client, falls vorhanden

Server (XSLT 2.0, Java-Umgebung): Java 1.6, Apache Tomcat 6.0, Apache Axis2 1.3, Saxon-B 9.0 (XSLT-Prozessor, Open Sorce-Variante)[9].

2.2.9.2 Hardware: für Server und Client getrennt

Keine speziellen Anforderungen, Hardware sollte aus Performance-Gründen halbwegs aktuell sein, serverseitig sollten mind. 8 GB RAM zur Verfügung stehen.

2.2.9.3 Produkt-Schnittstellen

Java, XSLT 2.0:
http://134.76.176.136:8080/axis2/services/seditor?wsdl (docstyle, Dateinamen/URIs als Param.)
http://134.76.176.136:8080/axis2/services/seditor2?wsdl (docstyle, zu verarb. Daten als Param.)
Parameter:
infile (Eingabedaten/URI), *outfile* (Ausgabedaten/URI), *xslfile* (XSLT-stylesheet/URI), *params* (optionale Key-Value-Paare für XSLT-Stylesheet), *sid* (RBAC-Session-ID), *log* (Logging-Info).

[9] http://saxon.sourceforge.net

1.7. Tokenizer

1.7.1. Produktübersicht Tokenizer

Zerlegt einen Text in eine Folge logischer Einheiten (Tokens), hier: Wörter. Diese werden durch Anfang- und Endemarkierungen gekennzeichnet.

1.7.2. Zielbestimmung

Die Tokenisierung der Texte dient der einfacheren Weiterverarbeitung mit Folgetools, z.B. Lemmatisierer.

Musskriterien

- Bestimmung und Auszeichnung von Wörtern und (optional) Satzzeichen
- Die Verarbeitung 'komplexer' Tokens (Abkürzungen, Eigennamen, etc.) wird ebenfalls unterstützt (Preprocessing, konfigurierbar).

Wunschkriterien

- Einzelne Unicode-Bereiche (z.B. über Drop-Down-Liste) auswählbar.
- Preview bzw. Anzeige der tokenisierten Daten in einem separaten Fenster (GUI).

Abgrenzungskriterien

Zu diskutieren bleibt, ob der Tokenizer auch zur Bestimmung und Auszeichnung größerer Texteinheiten, z.B. Sätzen, eingesetzt werden soll. Auch hierfür existiert eine Empfehlung des Unicode Konsortiums, allerdings dürfte der Anteil an erforderlicher manueller Nachbesserung ungleich höher liegen als auf Wortebene, zumal dieser Anwendungsfall eher selten auftreten dürfte. Eher Anwendungsfall für die OCR-Nachbereitung.

1.7.3. Produkteinsatz

Streaming Tool, als WebService gekapselt. Keine Grid-Features; diese werden von den in der Middleware bereitgestellten FileServices zur Verfügung gestellt.

1.7.4. Produktfunktionen

Auszeichnung von Wörtern innerhalb von Textdaten

Dient der Auszeichnung von Wörtern innerhalb neu erfasster (manuell oder OCR – Plain Text) oder noch nicht bis auf Wortebene hinab ausgezeichneter (XML) Texte. Auf das so generierte Markup können weitere Tools wie der Lemmatisierer aufsetzen. Markiert Wörter zunächst anhand der vom Unicode Konsortium vorgeschlagenen Regeln: http://www.unicode.org/reports/tr29/tr29-9.html Diese Regeln basieren auf der Zuordnung einzelner Unicode-Zeichen zu Zeichengruppen: http://www.unicode.org/Public/UNIDATA/auxiliary/WordBreakProperty.txt Gewünschte Anfang- und Endmarkierung der Wörter können in der Konfiguration angegeben werden. Voreinstellung ist ‹w› ... ‹/w›, (vgl. http://www.tei-c.org/release/doc/tei-p5-doc/html/ref-w.html)

Optionales Preprocessing

Preprocessing, bei dem "Wörter", Namen, Patterns, in einer Konfigurationsdatei (XML) zu definieren sind, die in einem ersten Arbeitsschritt eingelesen und abgearbeitet wird – noch bevor die eigentliche Applikationslogik zum Einsatz kommt. Dieses Verfahren bietet sich z.B. für die Auszeichnung von (häufig textspezifischen) Zeichengruppen an, die nicht von o.g. Algorithmus als Wörter erfasst werden können, wie Eigennamen, Mehrwortlexeme (bedingt), Abkürzungen, Datumsangaben (als Pattern), Bindestrichwörter. Über die Konfigurationsdatei ist auch eine Zuordnung von Token-/Pattern-Listen zu Typen möglich, bspw. nach dem STTS möglich, z.B. ‹w type="stts:NE"›Rufus T. Firefly‹/w›, oder mit einer noch detaillierteren Auszeichnung der Eigennamen gemäß ihres Typs (Person, Ort, Organisation). NB: Das Preprocessing erfolgt ausschließlich über Patternmatching und kann deshalb kein "echtes" POS-Tagging durchführen. Darum können Mehrwortlexeme, Klitika (Clitics) und Kardinal-/Ordinalzahlen nur ansatzweise als solche erfasst werden. In der Konfigurationsdatei können außerdem im Quelltext vorkommende Elemente oder Bereiche angegeben werden, deren Inhalt nicht vom Tokenizer verarbeitet werden soll.

1.7.5. Produktdaten

Welche Daten sind aus Benutzersicht (langfristig) zu speichern

Quelldatei, Zieldatei, Konfigurationsdatei, Log-Datei.

Wo liegen die Daten, von wo aus greift die Applikation darauf zu

Die Daten werden vom TextGrid-Lab oder dem Workflow-Manager via SOAP an den Service übergeben.

Welche Datenmengen müssen verarbeitet werden

Beliebig große Datenmengen, die auch in Teilen aus der Middleware geladen werden können.

Daten-Format

Verarbeitet mit XML-Markup versehene Textdaten. Zeichensatz: UTF-8 ohne BOM (Byte Order Mark).

1.7.6. Produktleistungen

Die Daten dürfen abgesehen vom ausgewählten Markup nicht verändert werden (Zeichenformat, Zeilenfall, Linebreaks, etc.).

1.7.7. Benutzeroberfläche

- Kontextmenü-Eintrag für XML-Editor-Perspektive
- Maske im TG-Lab für Parameter-Übergabe. Für XML-Dateien Preview über XML-Editor – noch zu realisieren.

1.7.8. Nichtfunktionale Anforderungen

Einzuhaltende Gesetze und Normen, Sicherheitsanforderungen, Plattformabhängigkeiten

Plattformunabhängig, da als WebService implementiert.

Welche Nutzungsdaten sollen (oder dürfen nicht) von der Middleware erhoben werden

Logging, Durchreichen von Benutzerdaten

1.7.9. Technische Produktumgebung

Software: für Server und Client, falls vorhanden

- Server:
  - Apache 2.0.x mod_python,
  - Python 2.4.x mit zusätzlichen Paketen:
    - ZSI-2.0_rc3 (http://pywebsvcs.sourceforge.net)
    - PyXML 0.8.4 (http://pyxml.sourceforge.net)
    - SOAPpy 0.11.6 (http://sourceforge.net/project/showfiles.php?group_id=26590&package_id=18246)
    - fpconst 0.7.2 (http://cheeseshop.python.org/pypi/fpconst/0.7.2)
- Client:
  - ADB-Client-Stub (Axis2)
  - Eingabemaske für Parameter [geplant - oder auch nicht]

Hardware: für Server und Client getrennt

Keine speziellen Anforderungen, Hardware sollte aus Performance-Gründen halbwegs aktuell sein.

Produkt-Schnittstellen

- Service ist publiziert unter http://ingrid.sub.uni-goettingen.de/Tokenizer.wsdl
- Parameter:
  - indata (xs:string) – der XML-codierte, zu tokenisierende Text
  - config (xs:string) – Konfiguration, in XML syntax:

1.8. Bibliographie-Tool

Es gibt eine Reihe von existierenden Tools und Services zur Verwaltung bibliographischer Daten, etwa PubMan (aus dem eSciDoc-Kontext) oder Bibsonomy. In TextGrid wird deshalb kein Bibliographie-Tool von Grund auf neu entwickelt werden, sondern es wird eine existierende Lösung eingebunden und so angepasst, dass die im Folgenden beschriebenen Anforderungen möglichst erfüllt werden.

1.8.1. Produktübersicht

Das Bibliographie-Tool ist ein Werkzeug, das dazu dient, **bibliographische Daten** entweder aus bereits vorhandenen Datenbeständen zu importieren, insbes. aus Bibliothekskatalogen (z.B. BDSL, zvdd, Kalliope) und den TEI-Headern (Metadaten) bestehender (TextGrid-)

Objekte, Bibliographien über eine definierbare Maske zu erfassen, zu bearbeiten, zu verwalten und an einer frei bestimmbaren Position in einer Datei einzufügen / abzuspeichern bzw. bibliographische Daten in best. Standard-Formaten zu exportieren (z.B. TEI, MODS). Das Tool unterstützt Formatierungskonventionen (Zitierformate) nach üblichen Standards (z.B. MLA Style). Es gibt zahlreiche Berührungspunkte / Überschneidungen mit anderen Tools (Metadaten-Annotation, XML-Editor, etc.).

Die Erstellung der bibliographischen Daten steht üblicherweise am Anfang eines Projektes, begleitet dies jedoch auch permanent (Daten werden in der Regel kontinuierlich ergänzt bzw. verbessert). Sie werden in vorhandenen Katalogen und Verzeichnissen recherchiert bzw. selbst (nach Autopsie) neu erstellt.

Genauso wie Daten importiert werden können, sollen Daten auch (durch verwendete Standards) in bestehende Kataloge exportiert werden können. Die Daten können als Ganzes oder in Teildatensätzen exportiert werden, es kann eine Auswahl der zu exportierenden Felder getroffen werden.

Musskriterien

Ausgangspunkt

- jedes *Projekt* hat eine Bibliografie, in der die Referenzen "flach" verwaltet werden; ggf Verwaltung durch Tags
- Einträge können dupliziert werden (von anderem Projekt, mit anderen Rechten bzw. um Zitierung unterschiedlicher Kapitel zu ermöglichen)

Rechte

- eine Projektbibliografie kann sein:
  a. öffentlich lesbar oder nicht lesbar
  b. von Mitarbeitern bearbeitbar oder nicht bearbeitbar
  → Default-Einstellung ist öffentlich lesbar und von Mitarbeitern bearbeitbar

Bildschirm (Mockup)

- zentral ist ein Schirm mit den Bibliografiedaten - jede Zeile ist ein Eintrag; ein Eintrag kann ein Textgrid Objekt sein, oder ein reiner Metadatensatz
- Doppelklick auf eine Zeile öffnet das Metadaten-Annotationstool - man kann die Metadaten bearbeiten, und/oder direkt eine Datei zu dem Metadatensatz hochladen (wird zu einem TextGrid Objekt)
- Es gibt eine eindeutige Identifizierung (automatisch generiert) für jeden Eintrag.

Entryt...	Author ▲	Title	Year ▼
Misc	{American Council of Learned Soci...	Social Sciences: Our Cultural Commonwealth	2006
Misc	Anderson	E-Science for the Arts and Humanities: A discussio...	
Misc	Berliner and Polk	{CVS}: Concurrent Versions System	
Article	Beynon et al.	Human Computing - Modelling with Meaning	2006
Incollecti...	Bradley	Text Tools	
Book	Busa	Index Thomisticus	1974
Misc	Caesar and MacCalla	e-Humanities in a Digital Society	
Article	Childs et al.	A Single-Computer Grid Gateway Using Virtual Mac...	2005
Article	Colazzo et al.	A Typed Text Retrieval Query Language for {XML} Do...	2002
Misc	CollabNet	Subversion	
Article	Corcho et al.	An overview of S-OGSA: A Reference Semantic Grid ...	2006
Incollecti...	Crane	Classics and the Computer	
Manual	Cunningham et al.	Developing Language Processing Components wit...	
Techrep...	on Digital Archive Attributes	Trusted Digital Repositories: Attributes and Respon...	2002
Book	{European Commission}	From Grids to Service-Oriented Knowledge Utilities	2006
Misc	Fish	Better {SCM} Initiative: Comparison	

Abbildung 3 - Jabref TextGrid.bib

Daten

- TextGrid Kernmetadaten - alles andere wird (vorläufig) "weggeschnitten"
- optional weitere Felder aus dem TEI-Header können angezeigt und manuell nachgetragen werden
- Verwaltung von Bibliothekssignaturen/Standorten (mehrere)
- Verlinkung mit Bibliothekskatalogen (extern)
- Daten können mehrfach vorgehalten werden, und haben keine Abhängigkeiten untereinander (wenn ein Datensatz geändert wird, ändern die anderen nicht mit)
  - zwischen Projekten, um z.B. unterschiedliche Rechte zu erlauben
  - innerhalb eines Projektes, für z.B. unterschiedliche Details (Kapitel x, Seite y)
- ein Zusatzeintrag ist ein Kurztitel (bibref), der automatisch unique vergeben wird, bzw manuell angepasst werden kann

Daten-Import

- aus dem Recherchetool können per Drag+Drop TextGrid Objekte in das Bibliografietool gezogen werden - Referenz zu TextGrid Objekt wird gelegt und Metadaten werden dargestellt
- Kataloge: zvdd und kalliope - diese Metadaten werden kopiert, potenziell unvollständig
- manuelle Eingabe
- ausserdem einlesbare Formate: MODS, TEI-Header
  - evtl. farbliche Darstellung, woher die Daten sind, aber keine explizite Trennung der Quellen in der Darstellung
  - Suche findet jeweils getrennt statt: in Recherche-Tool, in Katalogen oder in anderen (öffentlichen) Bibliografiedaten

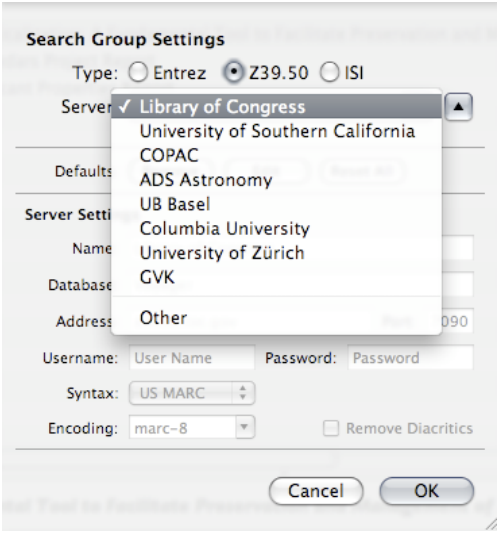


Abbildung 4 - Bibdesk: Katalogsuche

Daten-Export

betrifft beides

1. Generierung lesbarer Referenzen für existierende TextGrid-Objekte (aus den Metadaten oder TEI-Headern der entsprechenden Objekte)
2. Export eines Metadatensatzes bzw mehrerer Metadatensätze bis hin zu einer kompletten Bibliografie in einem Standard-Metadatenformat (z,B. MODS)

- einzelne Bibliografieeinträge sind über einen URI referenzierbar
- z.B. im XML-Editor kann eine Literaturliste in einem bestimmten Format erstellt werden (alle Werke sind einzeln mit Format-Fragment aufgelistet)
- mit Rechtsklick auf einen Bibliografieeintrag kann im Kontext-Menü über "Kopieren als ..." (z.B. TEI-Header) oder "Zitieren als ..." (vgl. MLA-Style) und der Wahl eines Formats eine formatierte Referenzierung kopiert werden
- Möglichkeit im Bibliografietool mehrere Einträge zu markieren und in einem Format nach Wahl zu exportieren
- Formate sind in einem ersten Schritt: TEI-Header, MODS (über bibutils: TeX, Endnote)

Verknüpfung TextGrid

- Erfassung Metadaten: bei Doppelklick auf einen Datensatz öffnet sich der *Metadateneditor*, und man kann direkt dort ein zugehöriges Objekt hochladen → wird zu TextGrid Objekt
- im *Schemagenerator* angegebene Einschränkungen für den Metadatensatz gelten auch im Bibliografietool. In der Bearbeitung im *Metadateneditor* kann das direkt übernommen werden. Weiters müsste dieser Aspekt auch für die Darstellung in der Tabellenansicht übernommen werden.
- Literaturlisten: ein Eintrag im Bibliografietool kann (in verschiedenen Zitierformaten) kopiert werden, und dann z.B. im *XML-Editor* eingefügt werden (ggf auch durch Drag+Drop)
- Tools wie z.B. der *Web-Publisher* greifen auf die REST-Schnittstelle des Bibliografietools zu (siehe Abschnitt "Daten-Export"), um ggf. druckbare Bibliografien zu generieren

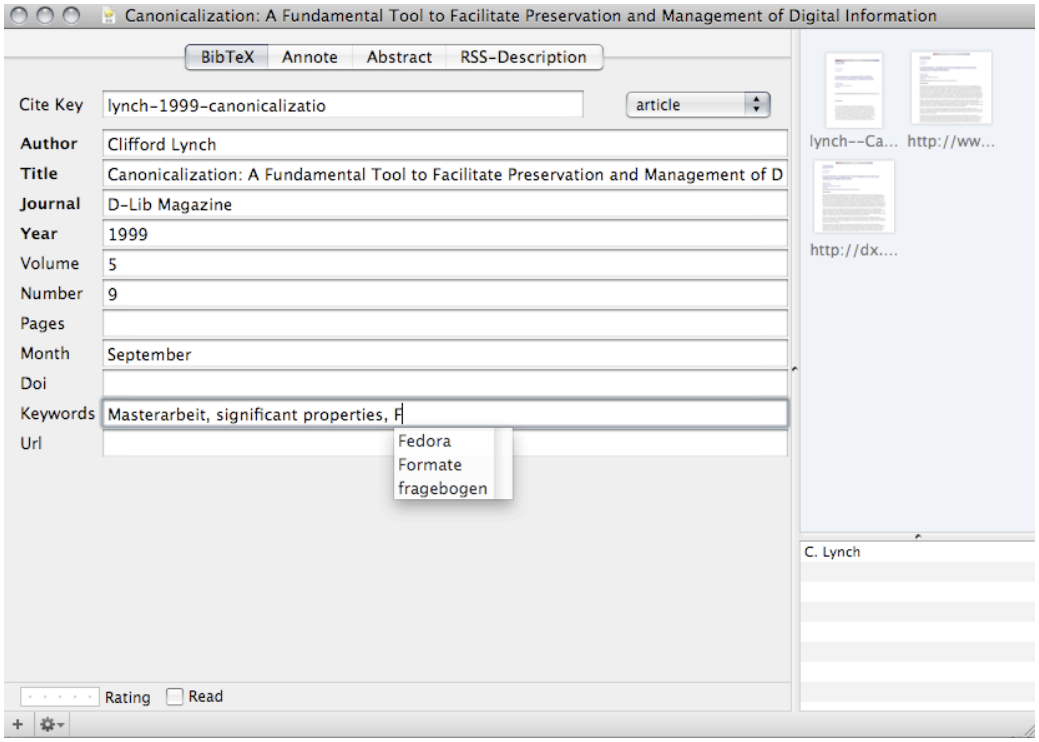


Abbildung 5 - Bibdesk: Metadateneingabemaske mit Upload, Tags (Autocompletion), etc

Wunschkriterien

- Auto-Ergänzungsfunktion (z.B. über ISBN)
- Verschlagwortung/sachliche Gliederung (mehrere) → Einbindung von Normdateien
- Verlinkung zum Volltext bzw. zu Exzerpten (extern)
- Verwaltung individueller bzw. kollaborativer Bemerkungen/Anmerkungen
- Verknüpfung zu zotero?
- Import aus http://www.bdsl-online.de/ (Dublin Core orientiertes Format)
- Export in MARC-XML, DocBook
- regelmässiger Export in das TextGrid als MODS Datei - zur Datensicherung (längerfristig)
- Verknüpfung mit Normdaten (Authority Files, PND, etc)
- Anbindung an die EDL (Katalog)

Abgrenzungskriterien

Die Erstellung einer dynamischen Bibliografie a la Bibtex \cite{} in Dokumenten muss von dem Editor verwaltet werden. Der TextGrid XML-Editor, OpenOffice, oder andere Editoren könnten dies über das TextGrid Bibliografietool ermöglichen, dies ist aber nicht die Aufgabe des Bibliografietools.

1.9. Web-Publisher

1.9.1. Produktübersicht Text Publisher (Web)

Ziel ist die Aufbereitung von TEI-Daten für die wissenschaftlich adäquate Publikation im WWW. Idealerweise konfigurierbare XSLT-Stylesheets als Grundlage für die Präsentationsform).

1.9.2. Zielbestimmung

Darstellung von einzelnen oder zu "Publikationen" zusammengefassten TextGrid-Objekten im Browser.

Musskriterien

Standardviews:

(über Templates auswählbar?)

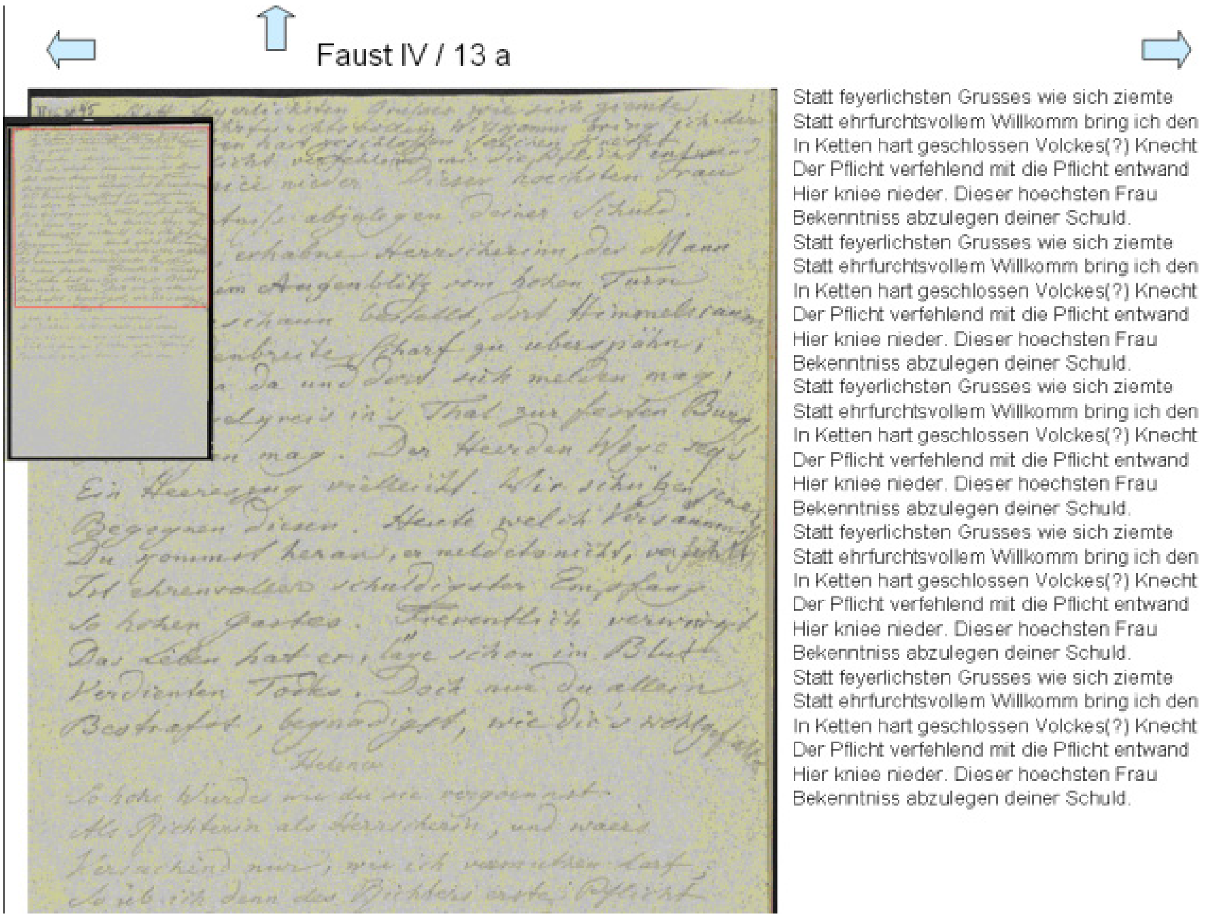
1. Text (TEI) (evtl. eingebettete Bilder)
2. Manuskript-Image und Transkription (nebeneinander)
    - Komplexere Templates z.B. für die Darstellung annotierter Glossen oder synoptische Ansicht von 1..n Textzeugen
3. Nur Images und Metadaten
4. Navigationsinseln, z.B. Einstiegsseite, Textsortenspezifische Gruppen

1) Text (TEI)

Sollte alle TEI-Tags und Attribute unterstützen und Standardeinstellungen für alle vorschlagen, die ausreichend kommentiert sind. → TEI-Stylesheets (v. Sebastian Rahtz)

2) Manuskript-Image und Transkription (nebeneinander)

Das könnte so aussehen:



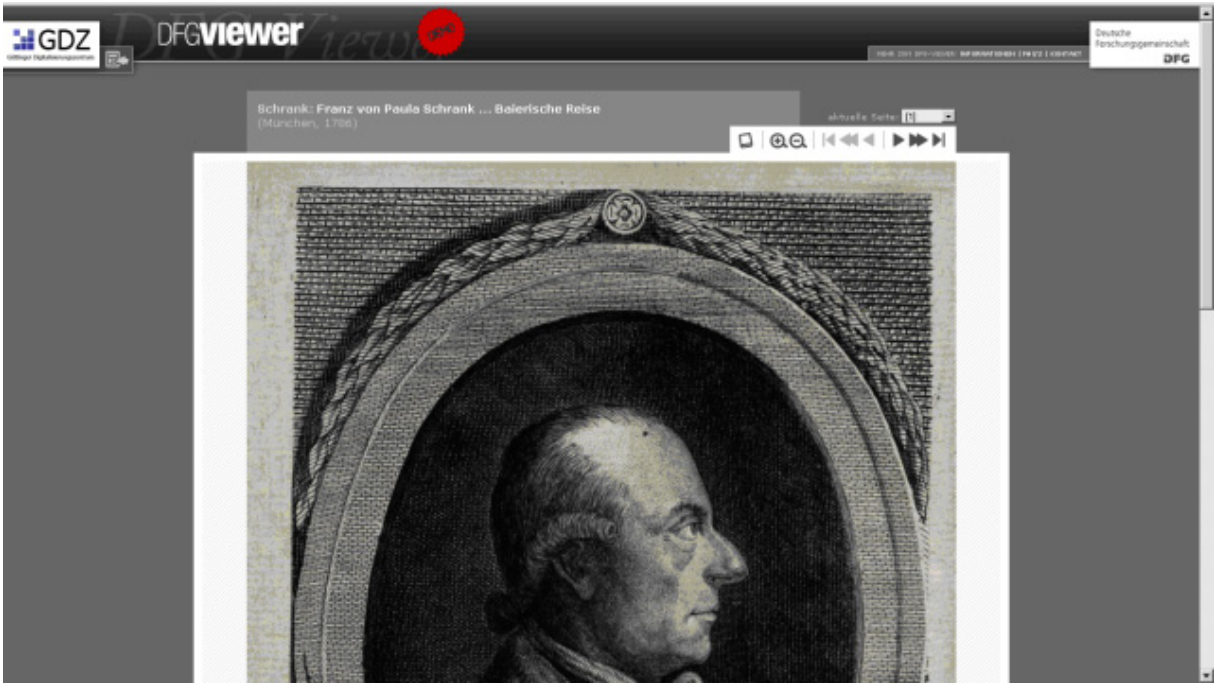
Das kleine Überblicksfenster sollte frei bewegbar und bis auf einen kleinen Platzhalter ausblendbar sein. Die beiden anderen Fenster (Bild und Text) sollten durch einen Teiler getrennt sein, der es erlaubt die Fenstergröße zu verändern (die Inhalte passen sich dann automatisch der Größe an).

Stufenplan für die Umsetzung:

1. Der rote Rahmen gibt den Ausschnitt auf dem Gesamtbild wieder.
2. Der rote Rahmen im Überblicksfenster sollte verschiebbar sein, so dass sich dann der Ausschnitt der Manuskriptwiedergabe verschiebt.
3. Auch der Text wird immer mitbewegt
4. Bewegt man die Maus über ein Wort im Manuskript, dann wird das Wort rechts hervorgehoben und umgekehrt, wenn man ein Wort anklickt, dann wird das links hervorgehoben.

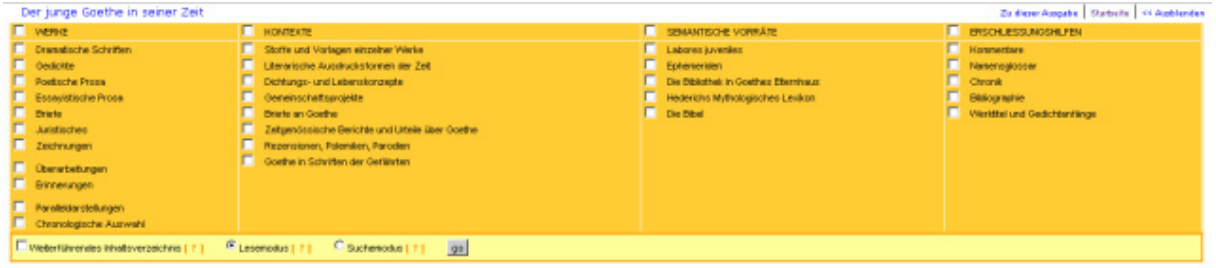
3) Nur Images und Metadaten

Könnte einfach die Daten in das METS/MODS Format des DFG-Viewers einwickeln und dann diesen zur Anzeige aufrufen



4) Navigationsinseln, z.B. Einstiegsseite, Textsortenspezifische Gruppen

Beispiel: Der Junge Goethe (*http://linglit193.linglit.tu-darmstadt.de:8080/exist/jgoethe/edition.xql?c=/db/jgoethe*):



Hier gibt es zum einen die Gruppierung der Texte nach bestimmten Gruppen, die dann wiederum Untergruppen enthalten können.

Hier also z.B. Werke → Dramatische Schriften:

Stellt sich die Frage, wie erstellen wir diese ‚Überblicke'?

Aus Benutzersicht wäre es wohl am leichtesten, wenn man Ordner (i.e. Aggregationen) anlegen könnte und dann diesen Ordnern Titel zuweisen könnte und die Inhalte der Ordner wiederum als Liste angezeigt werden könnte.

Im Projektbrowser: Rechts Anklicken bringt ein Menü, das u.a. die Zuordnung von Stylesheets zu Ordnern erlaubt oder auch erlaubt, Ordner aus der Publikation herauszunehmen (Sollte wiederum auch auf der Ebene des Objekts auch möglich sein.)

Gut wäre auch die Möglichkeit, links das Inhaltsverzeichnis auszuklappen:

Besonders interessant ist diese Möglichkeit, weil sie auch für die Suche verwendet werden kann, d.h. man kann hier ankreuzen, in welchen Texten man suchen will.

Außerdem gibt eine Reihe von Standarddateien, die man einem Projekt zurordnen können muss und die automatisch als Link auf der Einstiegsseite sichtbar sein sollten:

- **Projektbeschreibung** (kürzerer Titel)
- **Lizenz**
- **Kontakt**
- Außerdem natürlich **Suche**

Designer für den Webpublisher

Nach Möglichkeit relativ einfach halten, d.h.

1. Ein prinzipielles Konzept, wie man auf die Daten eines Projekts zugreift und die Daten über ein Art Pipeline, deren Komponenten differenziert definierbar sind, dann an den Webserver weitergereicht werden.
2. Vier Templates, die jeweils die oben genannten Use Cases exemplarisch lösen. Idealer Weise haben wir außerdem einen Designer im Lab, der es erlaubt, die Anordnung der Komponenten, die wir anbieten durch einfaches Anordnen der Platzhalter zu definieren, die dann z.B. via CSS das Template modellieren.

Wunschkriterien

Welche Anforderungen müssen nicht unbedingt erfüllt werden, sind aber sinnvolle Ergänzungen

- Publikation löschen bzw. zurückziehen?
- Möglichkeit, eine Suche über die Projektdaten in den Webauftritt zu integrieren
- Zu diskutieren bleibt noch das Rechtemanagement, d.h. besteht die Möglichkeit, "Vorab-Publikationen" nur einem ausgewählten Personenkreis zugänglich zu machen?
    - kryptischer URI
    - ODER Generierung .htaccess file
    - ODER Vergabe von Passwort bei Publikation, dieses muss Betrachter bekannt sein
    - ODER Anbindung an tg-auth (aufwändig!)

Abgrenzungskriterien

Welche Ziele sollen mit der Software bewusst nicht erreicht werden

- Keine komplexen Web 2.0 Anwendungen

1.9.3. Produkteinsatz

interaktiv oder/und Service, Einbindung in Workflow; Grid-Fähigkeit: Welche Grid-Features werden genutzt

- Interaktive GUI zum Zusammenstellen der Publikation

- Service-Komponenten:
    - Image Server
    - Lesen der Daten aus dem Grid (TG-crud, read)
    - Generierung der Publikation (Einspielen in XML-DB, Link-Resolving u.a.m.)

1.9.4. Produktfunktionen

Für jede Funktion ein Unterkapitel. Die Funktionen werden durch Anwendungsfälle (Use Cases) aus Benutzersicht beschrieben

Präsentation

- Generierung von (X)HTML

Navigation

- Navigation in HTML-Daten

Suche

- Volltext
- Suche in/nach Strukturdaten (XPath, XQuery)

1.9.5. Produktdaten

Welche Daten sind aus Benutzersicht (langfristig) zu speichern

- Quelldateien / -Objekte
- Stylesheets
- Konfiguration

Wo liegen die Daten, von wo aus greift die Applikation darauf zu

- XML-Quelldaten → eigene eXist-Instanz
- Grafiken: Web-Verzeichnis eines Apache HTTP-Servers

Welche Datenmengen müssen verarbeitet werden

Im Zweifelsfall beliebig große Datenmengen. Es empfiehlt sich jedoch aus Gründen der Performance und der Wartbarkeit, ggf. in kleinere und überschaubare Publikationen zu segmentieren und diese mit einer (manuell erstellten) Überblicksseite miteinander in Beziehung zu setzen.

Daten-Format

Eingabe:

- Grafik-Dateien (in TG unterstütze Formate)
- Textdaten in XML (standardmäßig unterstützt: Kerncodierung in TEI)
- Stylesheet(s) (XSLT, CSS) - falls Textdaten nicht in Kerncodierung vorliegen oder Standard-Transformation nach HTML modifiziert werden soll

Ausgabe:

- XHTML

1.9.6. Produktleistungen

Anforderungen an bestimmte Funktionen bezüglich Zeit (und Genauigkeit)

- Hohe/höchste Verfügbarkeit der Publikationen

1.9.7. Benutzeroberfläche

grundlegende Anforderungen (Fensterlayout, Dialogstruktur)

- Zusammenstellen der Publikation via Drag+Drop
- Eingabe von Titeln/Überschriften der Strukturelemente
- Metadaten für das Publikationsobjekt
- TODO:
    - Filter (via TG-search?) zur Auswahl der Objekte für größere Publikationen
    - Nutzer/Textsorten/Schema-spezifische Stylesheets auswählen
    - Auswahl des Templates für den Publikationstyp (s.o.)
    - ggf. Informationen für allfällige Zugangsbeschränkungen (z.B. .htaccess file generieren, muss noch diskutiert werden)

1.9.8. Nichtfunktionale Anforderungen

einzuhaltende Gesetze und Normen, Sicherheitsanforderungen, Plattformabhängigkeiten

- Frontend: Eclipse RCP
- Backend: Plattformunabhängig, da als WebService implementiert.

Welche Nutzungsdaten sollen (oder dürfen nicht) von der Middleware erhoben werden

Logging, Durchreichen von Benutzerdaten

1.9.9. Technische Produktumgebung

Software: für Server und Client, falls vorhanden

- Java 1.6
    - Client: REST-Client, Eclipse-Plugin
    - Server: Linux-Umgebung, Servlet-Container (Tomcat), Apache2 für Bilder, VIPS für Bildkonvertierung

Hardware: für Server und Client getrennt

Entsprechend den allgemeinen Projektvorgaben.

Produkt-Schnittstellen

- Web Service (SOAP/REST)

Vernetzt mit:

- TextGrid Lab (Frontend)
- TG-Crud (Backend)

2. AP 6.2

2.1. Text-Bild-Linkeditor

2.1.1. Stand der wichtigsten Features:

1. Bildbearbeitungsfunktionen: ca. 60%

2. Selektion Bildauschnitte: ca. 65%

3. Selektion Textabschnitte: ca. 65%

4. Standoff Verknüpfungsdatei: ca.:60%

2.1.2. Produktübersicht

Der Text Bild Link-Editor (auch Grafischer Linkeditor) ist ein Werkzeug, das es ermöglicht, Transkriptionstexte mit den zugehörigen digitalisierten Handschriften (Scans) zu verknüpfen. Der Nutzer kann dabei frei wählen, ob ganze Textseiten, Absätze, Zeilen, Einzelwörter oder sogar einzelne Buchstaben oder graphische Einheiten (etwa Markierungszeichen, Klammerungen, Formeln, etc.) mit den korrespondierenden Textabschnitten in der Transkription verknüpft werden. Die zur Verknüpfung benötigten Informationen speichert der TBLE in einer neuen Datei. Im Zusammenspiel der Transkriptionsdatei, der Bilddatei und der vom TBLE erzeugten Verknüpfungsdatei können digitale Texteditionen hergestellt werden, bei denen Transkription und Handschrift synoptisch dargestellt und vom Leser interaktiv aligniert werden können. Ferner können auch die Scans der Handschriften mit aktiven Zonen versehen werden, auf denen der Leser den korrespondierenden Transkriptionstext ein- und ausblenden kann. Die Verknüpfungsdatei des TBLE kann auch zur Positionierung von Textabschnitten im Satzspiegel bei diplomatischen Umschriften herangezogen werden.

2.1.3. Zielbestimmung

Der TBLE unterstützt die Herstellung von Editionen, in denen Transkriptionen und Faksimile gegenübergestellt werden (bspw. Historisch Kritischen Ausgabe im digitalen Medium). Dabei soll das Werkzeug durch eine weitgehend intuitive Nutzerführung auch Editionsphilologen, die lediglich über Grundkenntnisse in XML verfügen, die Vorbereitung von synoptischen und interaktiven Darstellungsformen (siehe oben) ermöglichen.

2.1.4. Musskriterien

Der TBLE bietet in der Bildanzeige (Image-View) alle Funktionalitäten, die nötig sind, um eine Handschrift als ganzes oder in ihren feinsten Details zu erfassen. Das Bild- und das Transkriptionsfenster können dabei frei positioniert werden, in der Standardeinstellung liegen sie untereinander. Für besonders großen Handschriften wird auch die Arbeit mit zwei Bildschirmen unterstützt, Handschriftenscan und Transkription können dann auf die beiden Bildschirme verteilt werden.

Im Bildfenster (Image View) ist die Kennzeichnung beliebiger Bildausschnitte in Form von Rechtecken oder Polygonen möglich. Ferner können Linien eingezeichnet und kodiert werden, die entweder als „magnetische“ Hilfe zur Ausrichtung einzelner Rechtecke genutzt werden können oder als typographische Grundlinie die Positionierung von Transkriptionen bei diplomatischen Umschriften unterstützen.

Auf der Seite des Texteditors können beliebige Textsegmente zur Verknüpfung mit Bildausschnitten selektiert werden. Hierbei werden zwei Anwendungsszenarien unterstützt: Sind in der Transkription die zu verknüpfende Textstellen bereits durch Elemente mit eindeutigen Identifikatoren (ID’s) ausgezeichnet, greift der TBLE auf diese ID’s zu; die Transkriptionsdatei bleibt in diesem Fall unverändert. Liegt hingegen noch keine passende Auszeichnung der Textabschnitte vor, markiert der TBLE die selektierten Textabschnitte durch <anchor>-Elemente.

Überlappungen sowohl der ausgewählten Bild- als auch Textabschnitte sind möglich.

2.1.5. Wunschkriterien

Da die Verlinkung von Text- und Bildsegmenten insbesondere bei umfangreicheren Transkriptionen recht Zeitaufwändig ist, besteht das Desiderat diesen Prozess zumindest teilweise zu Automatisieren. Während die Segmentierung des Transkriptionstextes durch den Tokenizer unproblematisch ist, muss für die automatisierte zeilen- oder seitenweise Segmentierung der Handschrift ein Verfahren entwickelt werden.

2.1.6. Abgrenzungskriterien

Im TBLE finden alle Arbeitsabläufe statt, die nötig sind, um die Verknüpfung von Textabschnitten mit Bildausschnitten zu kodieren. Eine weitere Bearbeitung der vom TBLE erzeugten Verknüpfungsdatei kann bspw. im XSLT Streaming-Tool erfolgen. Zur Publikation der Transkription mit den Faksimile-Scans können diese zusammen mit der Verknüpfungsdatei an den Webpublisher weitergegeben werden. Nach erfolgter Integration von DigiLib in die TextGrid-Umgebung sollen die Vorteile dieses Werkzeugs – insbesondere bei der Betrachtung sehr hochauflösender Bilder – für den TBLE nutzbar gemacht werden.

Workflow 1: Verknüpfung mit Schreibvorgang in der Transkriptionsdatei

Sind in der Transkriptionsdatei die zu verknüpfenden Textabschnitt noch nicht mit eindeutig identifizierbaren Elementen ausgezeichnet, werden diese Abschnitte vom Nutzer selektiert, per Knopfdruck werden die Abschnitte dann mit <anchor>-Elemente mit eindeutigen ID’s markiert. Dazu ist ein schreibender Zugriff auf die Transkriptionsdatei nötig.

Workflow 2: Verknüpfung ohne Schreibvorgang in der Transkriptionsdatei

Sind hingegen die zu verknüpfenden Textabschnitt bereits mit eindeutig identifizierbaren Elementen ausgezeichnet, liest der TBLE lediglich die Element ID's in die Verknüpfungsdatei ein. Ein schreibender Zugriff ist nicht notwendig. Die TextGrid Rechteverwaltung ermöglicht es, dass der Nutzer, der die Verlinkungen vornimmt, auf die Transkriptionsdatei ausschließlich Leserechte hat; versehentliche Änderungen in der Transkriptionsdatei sind somit ausgeschlossen. Die Bilddatei (Scan) bleibt in jedem Fall unverändert.

2.1.7. Produktfunktionen

Einbindung in die Workbench

Der TBLE ist vollständig in das TextGrid-User-Interface (TextGridLab) integriert. Er stellt eine eigenständige Perspektive dar, die den XML-Editor und den Navigator mit der Bildanzeige (Image-View) und einer Miniaturansicht (Thumbview) verbindet. Im Navigator werden die zu bearbeitenden XML- und Bilddateien ausgewählt; das Öffnen dieser Dateien initialisiert den TBLE. Im XML-Editor findet die Selektion der Textabschnitte statt, bereits verknüpfte Textabschnitte werden optisch hervorgehoben.

Bildbetrachtungsfunktionen

Die folgenden Funktionen gewährleisten das Lesen der enthaltenen Informationen sowie Markieren beliebiger kleiner oder großer Bildsegmente:

- Anzeige der Originalgröße
- Einpassen des gesamten Bildes in den Image View
- Horizontales oder vertikales Einpassen des Bildes in den Image View
- Zoomfunktionen (stufenloses Vergrößern und Verkleinern des Bildes)
- freies Verschieben des Bildes im Zoom-Modus
- Unterstützung der Orientierung im Zoom-Modus über eine Miniaturansicht (Thumbview). Die Ausschnittwahl in der Thumbview läuft synchron mit der Ausschnittanzeige in der Image View.
- Lupenfunktion: lokal begrenzter, schneller Wechsel zwischen normaler und vergrößerter Ansicht (Wechsel zwischen Detail- und Übersichtsperspektive)
- Drehen des Bildes mit fest vorgegebenem Raster (z.B. in 45° Schritten)
- Beliebig viele Images können parallel (kaskadierend) in die Image View geladen werden (bspw. Seiten einer Handschrift). Die Navigation in den Images erfolgt über Tabs.
- Als separates Fenster erlaubt die Toolbox die Auswahl einiger zentraler Bildbetrachtungsfunktionen sowie der Zeichenwerkzeuge für die Wahl der Bildausschnitte.

Selektion der Bildausschnitte

Die folgenden Zeichenfunktionen ermöglichen das Markieren beliebiger Informationseinheiten auf einem Bild (Image Scan):

- Rechtecke und Polygone mit beliebiger Anzahl an Eckpunkten
- beliebiges Ändern bereits gezeichneter Formen durch Anpassen der Größe, Form und/oder Verschieben
- Formen können sich beliebig Überlappen oder ineinander geschachtelt sein
- Formen können beliebig wieder gelöscht werden, beim Workflow 1 werden die korrespondierenden <anchor>-Elemente in der Transkriptionsdatei ebenfalls gelöscht.
- Zur schnelleren Markierung von Bildsegmenten gleicher oder ähnlicher Form (bspw. Wörter, Zeilen) können die Formen horizontal oder vertikal fortlaufend geklont werden.
- Zur Markierung von schräg verlaufenden oder auf dem Kopf stehenden Schriften können die Rechtecke per Mausaktion oder über einen Eingabedialog stufenlos gedreht werden.
- Das Einblenden eines Hilfslinienrasters mit frei wählbaren Abständen unterstütz das regelmäßige Positionieren von Formen
- Einzelne Hilfslinie können mit beliebiger Orientierung in das Bild gezeichnet werden
- Die Farbgebung von Formen (differenziert in aktiv und inaktiv) und Linien kann beliebig gewählt und auf die Farb- und Kontrasteigenschaften des Bildes abgestimmt werden.
- Die Hilfslinien können ein- und ausgeblendet werden, zur späteren Verwendung können sie mit den anderen Formen in der Verknüpfungsdatei abgespeichert werden.
- Über eine „Magnetfunktion“ (snap in) können Formen entlang einer Hilfslinie automatisch ausgerichtet werden.
- Als typographische Grundlinie können Hilfslinien auch dazu verwendet werden, neben der Angabe der Gesamtkontur einer Schrift auch deren Grundlinie zu kodieren (→Vorbereitung Positionierung bei diplomatischen Umschriften)
- Wünschenswert ist bei fortgeschrittenem Entwicklungsstand die Einbindung eines Kreis- oder Ellipsenwerkzeugs (ggf. Bezierkurvenwerkzeugs), welches das Anlegen typographischer Grundlinien unter halbkreisförmig geschriebene Textzeilen ermöglicht.
- Bei besonders dicht beschriebenen Manuskripten ist die Organisation der Formen in verschiedenen Layers erwünscht. So kann bspw. die erste Niederschrift in Layer 1 erfolgen, die Autorenkorrektur in Layer 2 und Eintragungen von fremder Hand in Layer

3. Eine Anzeige aller Layer sollte ebenfalls möglich sein. Jedem Layer sollte eine eigene Farbe sowie ein Identifikator zugewiesen werden können.

Kodierung von Schriftmerkmalen (Laufrichtung der Schrift)

Über das Attribut „writing mode“ des <text>-Elements in SVG kann die Laufrichtung der Schrift kodiert werden. Diese Angabe soll global für ein ganzes Dokument oder für jede Form einzeln vergeben werden können (Bsp.: Arabisches Zitat in lateinischem Text). Ein Label, das den eingestellten „writing mode“ in der Image View anzeigt, kann optional eingeblendet werden.

Selektion der Textausschnitte

Im Fall des Workflow 1 erfolgt die Auswahl der Textabschnitt durch selektieren der selben per Maus „ziehen“ oder über die Tastatur. Bereits verknüpfte Textabschnitte werden grafisch hervorgehoben. Um beliebige Überlappungen zu ermöglichen, müssen bereits als verknüpft angezeigte Textabschnitte erneut selektiert werden können.

Beim Workflow 2 genügt das Positioneren des Cursers in das angewählten Element. Erkennt der TBLE die ID des Elementes, sollte dies durch graphische Hervorhebung rückgemeldet werden. Auch hier ist auf die Möglichkeit der Überlappung zu achten (bspw. <w>-Elemente in <l>-Element)!

Verknüpfung von Text und Bild

Sind ein Textabschnitt und ein Bildausschnitt angewählt, kann über einen Schalter/Shortcut die Verknüpfung angelegt werden. Text- und Bilddatei stehen dabei in einer n:n-Beziehung, das heißt es können beliebige viele Texte mit beliebig vielen Bildern verknüpft werden. Der Status der Formen (verknüpft oder noch nicht verknüpft) wird graphisch angezeigt. Alle Informationen zu dieser Verknüpfung werden in die Standoffdatei geschrieben. Sämtliche topographische Informationen werden dabei im SVG-Standard kodiert, Koordinatenangaben werden in Prozentwerten abgenommen, um vollständige Freiheit in der Skalierung zu erreichen. Bei Anwendung des Workflow 1 werden die ID’s der <anchor>-Elemente nach dem Modell des Three-way-alignment aus TEI in <linkGrp>-Elementen verwaltet. Die Standoffdatei ist eine valide TEI-Datei (mit SVG-Erweiterung); ein Anzeigemodus erlaubt das Lesen und kontrollieren der Inhalte der Standoffdatei. Wünschenswert ist ein Eingabedialog, über den einige Parameter des TBLE an individuelle Nutzeranforderungen angepasst werden können, z.B. die Benennung der ID-Werte oder alternative Elemente zum <anchor>-Element. Die Individualisierung welcher Parameter von Interesse ist und welche Kontrollmechanismen gegebenenfalls dazu benötigt werden, soll in enger Absprache mit Editionsprojekten geklärt werden, die mit dem TBLE arbeiten.

Wird eine bereits verknüpfter Textabschnitt oder Bildausschnitt angewählt, werden synchron die korrespondierenden Text- oder Bildstellen graphisch hervorgehoben (→Kontrolle der Verknüpfungen.).

Ausschnittablage

Die Ausschnittablage bietet die Möglichkeit Bildausschnitte, bspw. Schriftproben aus einer Handschrift, in ein separates Fenster einzufügen und zu speichern. Bei der Entzifferung von

Handschriften kann dann ein Vergleich mit den Schriftproben erfolgen. Das Anlegen mehrerer solcher Ausschnittablagen sollte möglich sein.

2.1.8. Wünschenswerte Produktfunktionen

Erweiterte Bildanzeigefunktionen

Das ebenfalls in Eclipse entwickelte Softwareplattform „Edition Production & Presentation Technology (EPPT)“[10] mit ähnlicher Zielsetzung wie der TBLE bietet eine Reihe von erweiterten Bildanzeigefunktionen, die je nach technischer Kompatibilität in den TBLE übernommen werden sollen (bspw.: Overlay: paralleles Betrachten eines Bildes in verschiedenen Aufnahmemodi wie invertiert, Röntgenaufnahme oder andere).

Kodierung von graphischen Bildelementen in SVG

Eine Erweiterung des Selektionswerkzeugs für Bildausschnitte ermöglicht das Nachzeichnen graphischer Bildelemente (Streichungen, Symbole, graphische Skizzen, Pfeile, etc.) auf dem Bild Scan und kodiert diese Graphiken in SVG. Bei der Publikation können diese SVG-Graphiken dann wieder in den Text eingefügt und dargestellt werden.

(Semi)Automatische Segmentierung von Textabschnitten (Wörter, Zeilen) in Handschriftenscans

Bei sehr regelmäßig geschriebenen Handschriften sowie bspw. bei maschinenschriftlichen Typoskripten erscheint eine automatisierte Selektion der entsprechenden Bildausschnitte erwünschenswert. Zur Umsetzung dieser Funktion, soll geprüft werden, ob die Bildsegmentierung der OCR-Software für diese Aufgabe angepasst und in den TBLE integriert werden kann. Alternative Verfahren, die in letzter Zeit in anderen Projekten entwickelt wurden, sollen auf ihre Nachnutzbarkeit geprüft werden.[11]

Online und offline Workflow

Wird im TBLE online gearbeitet, können die Daten über die von der Middleware bereitgestellten File Services editiert werden. Zusätzlich sollte es möglich sein, Dateien lokal zu laden und auch lokal wieder abzuspeichern. Die so lokal erzeugten Daten können natürlich dann auch wieder ins Grid importiert werden.

Usability

Die Benutzeroberfläche soll so gestaltet sein, dass auch Editionsphilologen mit wenig Erfahrung mit XML-Technologie einen weitgehend intuitiven Zugang finden. Da das Verknüpfen von Text und Bild bei umfangreichen Editionen häufig ein repetitiver, routinenhafter Vorgang ist, sollen möglichst viele Funktionen auch über Shortcuts anwählbar sein, um möglichst wenig zwischen Maus und Tastatur wechseln zu müssen.

Datentypen

Die Transkriptionsdatei muss eine XML-Datei sein, häufig wird sie TEI konform sein. Mit sehr großen XML-Dateien ist zu rechnen, deren Verarbeitung ist im XML-Editor vorgesehen (siehe Pflichtenheft XML-Editor). In jedem Fall können große Dateien über das XSLT Strea-

[10] EPPT wurde an den Universitäten von Kentucky und Georgia entwickelt. Der Quellcode des Programms liegt „Open Source“ zur Verfügung: http://sourceforge.net/projects/eppt/

[11] Vgl.: http://www.balisage.net/Proceedings/vol1/html/Cayless01/BalisageVol1-Cayless01.html und http://kups.ub.uni-koeln.de/volltexte/2009/2968/

ming Tool auch in kleinere Einheiten zerlegt werden. Die vom TBLE erzeugte Verknüpfungsdatei ist eine TEI-P5 konforme, um den SVG-Standard erweiterte XML-Datei.

Der Image Editor sollte alle gängigen Bildformate (TIFF, JPG, PNG, etc.) unterstützen; bei hochauflösenden Graphiken ist mit sehr großen Dateien (mehrere hundert MB) zu rechnen.

2.2. Kollationierung

Die Entwicklung des TextGrid-Kollationierers stand in einer Zeit an, in der verschiedene europäische Partner Interesse an der Entwicklung eines Kollationierers hatten. Hinzu kam, dass das recht verbreitete Werkzeug *Collate 2* von Peter Robinson eng mit einer Betriebssystemplattform (MacOS Classic) verknüpft war, die nicht mehr gepflegt wird und für aktuelle Hardware auch von Apple nicht mehr zur Verfügung steht: Damit geht diese Software verloren.

Der Kollationierer wird darum im Rahmen der ESF-COST-Action *Interedition*[12] unter dem Namen *CollateX*[13] entwickelt. TextGrid bzw. die Uni Würzburg beteiligt sich an der Entwicklung, darüber hinaus stellt TextGrid Softwarekomponenten zur verfügung, mit der sich *CollateX* in die restliche TextGrid-Architektur einfügt.

2.2.1. Produktübersicht

Zwei bis beliebig viele TEI-kodierte Dokumente werden verglichen und ihre Unterschiede werden nach TEI kodiert und notiert.

2.2.2. Zielbestimmung

Eine elektronische Edition ermöglicht es, grundsätzlich alle in den benutzten Quellen festgestellten Varianten darzustellen; der Platz ist nicht länger notwendig ein limitierender Faktor. Aber neben der reinen Dokumentation fällt einer Edition auch die Aufgabe zu, die Textgenese für den Leser (oder besser: Nutzer) nachvollziehbar zu machen. Wenn sämtliche Varianten gleichgewichtig dargestellt werden, so sinkt die Übersichtlichkeit rapide mit der Zahl der Quellen. Auch bei einer elektronischen Edition muss sich ein Editor also im Allgemeinen entscheiden, welche Variationen er als signifikant betrachtet. Der Editor muss deshalb einem Kollationierer in TextGrid Regeln übergeben können, welche Unterschiede als nachrangig behandelt werden sollen. (Etwa wenn der Editor der Schreibung eines Wortes mit dem Graphem *u* oder dem Graphem *v* keine besondere Bedeutung beimisst.) Als Teil dieser Regeln muss der Kollationierer auch Ausnahmelisten unterstützen.

Für einen Kollationierer ist es sehr aufwändig, völlig autonom zu erkennen, dass in einzelnen Quellen ganze Textblöcke ausgelassen oder gar verschoben wurden, speziell wenn die korrespondierenden Blöcke noch zusätzliche mehr oder weniger zahlreiche Varianten enthalten. Deshalb soll der für TextGrid zu entwickelnde Kollationierer sog. Aufsetzpunkte verarbeiten können, also Markierungen, von denen ausgehend der Vergleich aller Fassungen wieder synchronisiert wird. Weil diese Aufsetzpunkte nicht für alle Fassungen in der gleichen Reihen-

[12] http://www.interedition.eu/
[13] http://collatex.sourceforge.net/

folge defininiert sein müssen, kann der Editor damit dem Kollationierer Verschiebungen größerer Textblöcke manuell mitteilen.

Ähnlich wie bei der Lemmatisierung sind in der Literatur verschiedene Vergleichs-Algorithmen beschrieben. In einer der neuesten Arbeiten schlagen Spencer und Howe [SH2004] eine Methode vor, die ohne Festlegung auf einen Basistext auskommt und auf Verfahren aus der Bioinformatik aufbaut. Auch hier lässt sich nicht aus dem Ergebnis der Literaturrecherche ableiten, dass TextGrid einem der vorgeschlagenen Algorithmen den Vorzug geben sollte, weshalb die Entscheidung, welche Kollationierungsmethode als erste implementiert wird, nach pragmatischen Gesichtspunkten getroffen werden kann.

Musskriterien

Beschreibung:

Zwei bis beliebig viele TEI-kodierte Dokumente werden verglichen und ihre Unterschiede werden nach TEI kodiert und notiert.

Details:

Eine Kollationierung wird primär auf verschiedenen Fassungen des Originaltextes durchgeführt, kann aber auch auf der Auszeichnungsebene arbeiten

2.2.3. Daten-Format

Eingabe:

- Text, ggf. XML

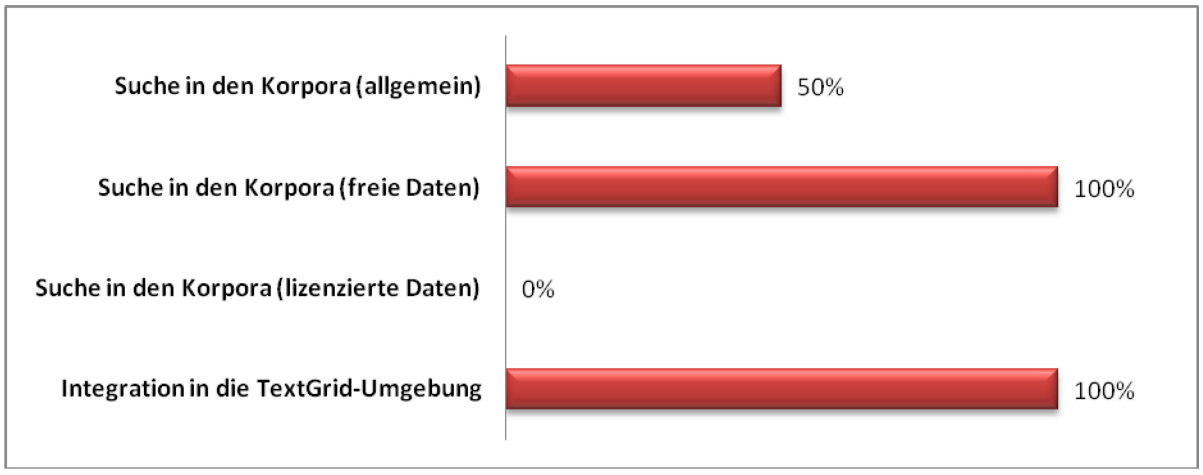
Ausgabe:

- XML / TEI (z.B. mindestens Parallel Segmentation http://www.tei-c.org/release/doc/tei-p5-doc/html/TC.html#TCAPPS)

3. AP 6.3

3.1. Integration COSMAS

3.1.1. Entwicklungsstand



3.1.2. Produktübersicht

Dieses Vorhaben soll das am Institut für Deutsche Sprache entwickelte Korpusanalyse- und -recherchesystem COSMAS in die TextGrid-Umgebung integrieren.

3.1.3. Zielbestimmung

Die Integration des Systems soll dem Textwissenschaftler ermöglichen, auf einfache Weise kontextabhängige Direktrecherchen und –analysen auf der am IDS verwalteten Datengrundlage durchzuführen.

Musskriterien

Die reibungslose Kommunikation mit COSMAS muss über entsprechende interne Schnittstellen sowohl seitens der TextGrid-Komponente als auch seitens COSMAS gewährleistet werden.

Wunschkriterien

Eine Erweiterung der Integration auch auf Datenbestände aus nicht-freien Korpora, für die geeignete Authentifizierungsmechanismen erforderlich sind, ist erstrebenswert.

Abgrenzungskriterien

Nicht anwendbar.

3.1.4. Produktfunktionen und -leistungen

Als Schnittstelle zum Korpusanalyse- und –recherchesystem COSMAS stellt dieses Modul die Möglichkeit zur Verfügung, die Analysewerkzeuge des Systems kontextbezogen auf Ausdrücke aus TextGrid-Objekten anzuwenden und sich die Ergebnisse anzeigen zu lassen.

3.1.5. Benutzeroberfläche

Das Modul wird im Rahmen der Rich Client Platform in die Eclipse-Oberfläche eingepasst und implementiert keine Benutzeroberfläche im eigentlichen Sinne. Die Funktionalität kann nach der Markierung eines Suchbegriffs über Kontextmenüs aufgerufen werden und das Ergebnis erscheint in einem separaten Fenster.

3.1.6. Nichtfunktionale Anforderungen

Einzuhaltende Gesetze und Normen, Sicherheitsanforderungen, Plattformabhängigkeiten

Ergebnisse aus gestellten Suchanfragen dürfen nur aus dem jeweiligen Datenbestand erhoben werden bzw. nur an den Nutzer ausgegeben werden, der die entsprechenden Rechte dazu hat. Liegen keine gesonderten Rechte vor, muss die Anfrage auf einer kleinen Untermenge, einem freien Grunddatensatz erfolgen.

Welche Nutzungsdaten sollen (oder dürfen nicht) von der Middleware erhoben werden

Es dürfen keine unanonymisierten Daten erhoben werden.

3.1.7. Technische Produktumgebung

Software

Das Modul wird in einer Client-Server-Architektur konzipiert. Während die clientseitigen Oberflächenkomponenten im Rahmen des TextGridLab in Eclipse eingebunden werden, sollen serverseitig WebServices angeboten werden, um die Funktionalität zur Verfügung zu stellen.

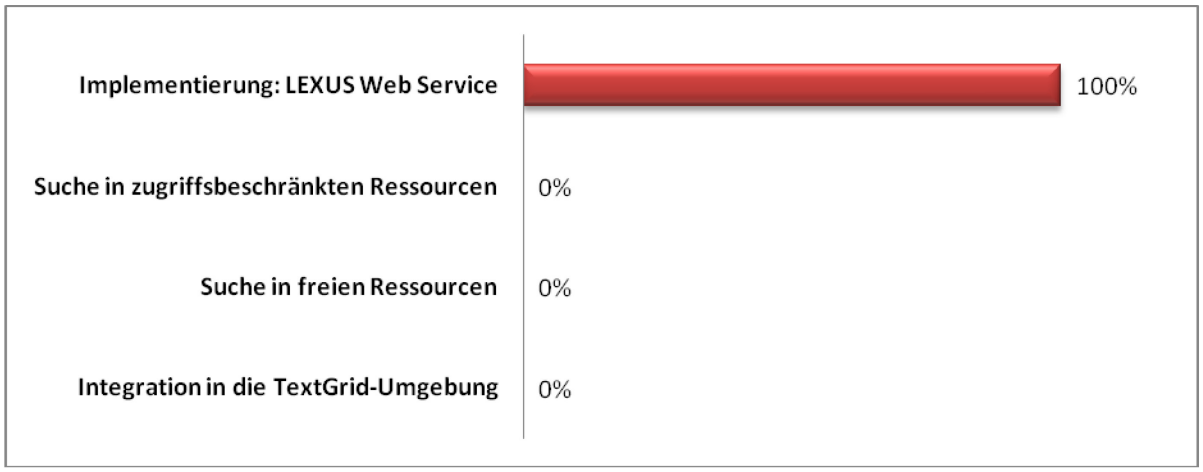
Hardware

Keine (zusätzlichen) Anforderungen.

Produktschnittstellen

Die WebService-Schnittstelle in der Client-Server-Architektur wird im Representational State Transfer (REST) Softwarearchitekturstil angeboten.

3.2. Integration LEXUS



3.2.1. Produktübersicht

Die Integration von LEXUS soll den Zugriff auf lexikalische Ressourcen ermöglichen, die am MPI für Psycholinguistik in Nijmegen mittels des genannten Tools erstellt und online bereit gestellt werden.

3.2.2. Zielbestimmung

Die Integration soll es Fachwissenschaftlern erlauben, auf einfache Weise Recherchen in über LEXUS verwalteten lexikalischen Ressourcen durchführen zu können.

Musskriterien

Es muss eine reibungslose Kommunikation mit LEXUS über interne Schnittstellen seitens TextGrid als auch seitens LEXUS erreicht werden. Der Zugriff auf in LEXUS freie Ressourcen muss möglich sein. Das Modul muss über die GUI des TextGridLab angesprochen werden können.

Wunschkriterien

Es wäre wünschenswert, einen reibungslosen Zugriff auf solche Ressourcen zu ermöglichen, die in LEXUS nur beschränkt freigegeben sind. Hierfür wäre eine Abstimmung hinsichtlich geeigneter Authentifizierungs- und Autorisierungsmechanismen zu erreichen.

Abgrenzungskriterien

Module, die ähnliche Ziele verfolgen, sind bereits in Form von „Dictionary Search“ und „Integration COSMAS“ umgesetzt. Ersteres Tool bindet das Trierer Wörterbuchnetz ein, letzteres hat dagegen die Integration des Korpusrecherche- und -analysesystems des IDS zum Gegenstand. Im Zuge des hier beschriebenen Moduls sollen dagegen lexikalische Ressourcen des MPI für Psycholinguistik in Nijmegen zugänglich gemacht werden.

3.2.3. Produktfunktionen, -daten und –leistungen

Eine API beschreibt den durch den Client geführten Selektionsprozess:

- Client sucht Lexikon für Suche aus

- Client ruft spezifische Informationselemente bzgl. des Lexikon ab
- Client wählt Datenkategorie für detaillierte Suchanfrage aus
- Liste lexikalischer Einträge wird entsprechend lexikon-spezifischem Schema zurückgegeben. Es gibt es ein Standard-Kodierungsschema für alle Lexika.

3.2.4. Benutzeroberfläche

Das Modul wird im Rahmen der Rich Client Plattform in die Eclipse-Oberfläche eingebunden. Es soll ggfs. auch kontextsensitiv nach Markierung eines Suchbegriffs aufgerufen werden können und Ergebnisse in einem Fenster anzeigen.

3.2.5. Nichtfunktionale Anforderungen

Einzuhaltende Gesetze und Normen, Sicherheitsanforderungen, Plattformabhängigkeiten

Ergebnisse aus gestellten Suchanfragen dürfen nur aus dem jeweiligen Datenbestand erhoben werden bzw. nur an den Nutzer ausgegeben werden, der die entsprechenden Rechte dazu hat. Liegen keine gesonderten Rechte vor, muss die Anfrage auf einer kleinen Untermenge, einem freien Grunddatensatz erfolgen.

Plattformunabhängig.

Welche Nutzungsdaten sollen (oder dürfen nicht) von der Middleware erhoben werden

Noch keine Angabe möglich.

3.2.6. Technische Produktumgebung

Software

Das Modul wird im Rahmen einer Client-Server-Architektur konzipiert. Während die clientseitigen Oberflächenkomponenten im Rahmen der Rich Client Platform in Eclipse eingebunden werden, werden serverseitig Web Services angeboten, um den Zugriff auf die lexikalischen Ressourcen am MPI zur Verfügung zu stellen.

Hardware

Entsprechend den allgemeinen Projektvorgaben.

Produktschnittstellen

Seitens des MPI wird bereits ein LEXUS Lookup Web Service, der via SOAP kommuniziert und die Suche in heterogen strukturierten lexikalischen Ressourcen ermöglicht. Die folgenden Methoden werden bereit gestellt:

- login
- getResource
- getResources

- getDataCategories
- getSchema
- search

Der Service wird zur Zeit in Interaktion mit den ebenfalls am MPI entwickelten Tools ANNEX und ELAN getestet.

4. AP 6.4

4.1. Integration Digilib

Digilib[14] ist ein Werkzeug zum Ansprechen von Ausschnitten oder skalierten Versionen großer digitaler Bilder über das Netz. D.h. ein großes Digitalisat muss nicht notwendigerweise ganz zum Client übertragen werden, sondern es kann vielmehr zunächst eine verkleinerte Version geschickt werden, auf der der Benutzer dann einen Ausschnitt markiert, der daraufhin in voller Größe übertragen werden kann.

4.1.1. Architekturmodell Digilib – TextGrid

Ziel dieses Dokuments

Dieses Dokument erläutert den ersten Entwurf eines Architekturmodells zur Einbindung von Digilib in TextGrid.

Im Folgenden werden lediglich Komponenten beschrieben, die den Digilib-Bestandteil in TextGrid ausmachen. Deren Anordnung innerhalb des TG Systems lässt sich aus der Grafik auf Seite 2 entnehmen.

Scaler:

Das Herzstück Digilibs ist ein Servlet und übernimmt die Aufbereitung und Berechnung der Bilddaten.

Gesteuert wird der Scaler über ein Parameter (z.B. gewünschtes Bild, Zoom Area,Markierungspunkte), die beim Aufruf übergeben werden. Der Scaler wird im Rahmen der allg. Digilib-Entwicklung um die Fähigkeit erweitert, im Grid Bild-Daten zu lesen. Dies wir über TG-CRUD geschehen.

DigilibActivator:

Der DigilibActivator ist vom Typ Plugin und aktiviert den Digilib Scaler mit den notwendigen Konfigurationen als Servlet innerhalb des Context.

Digilib Client:

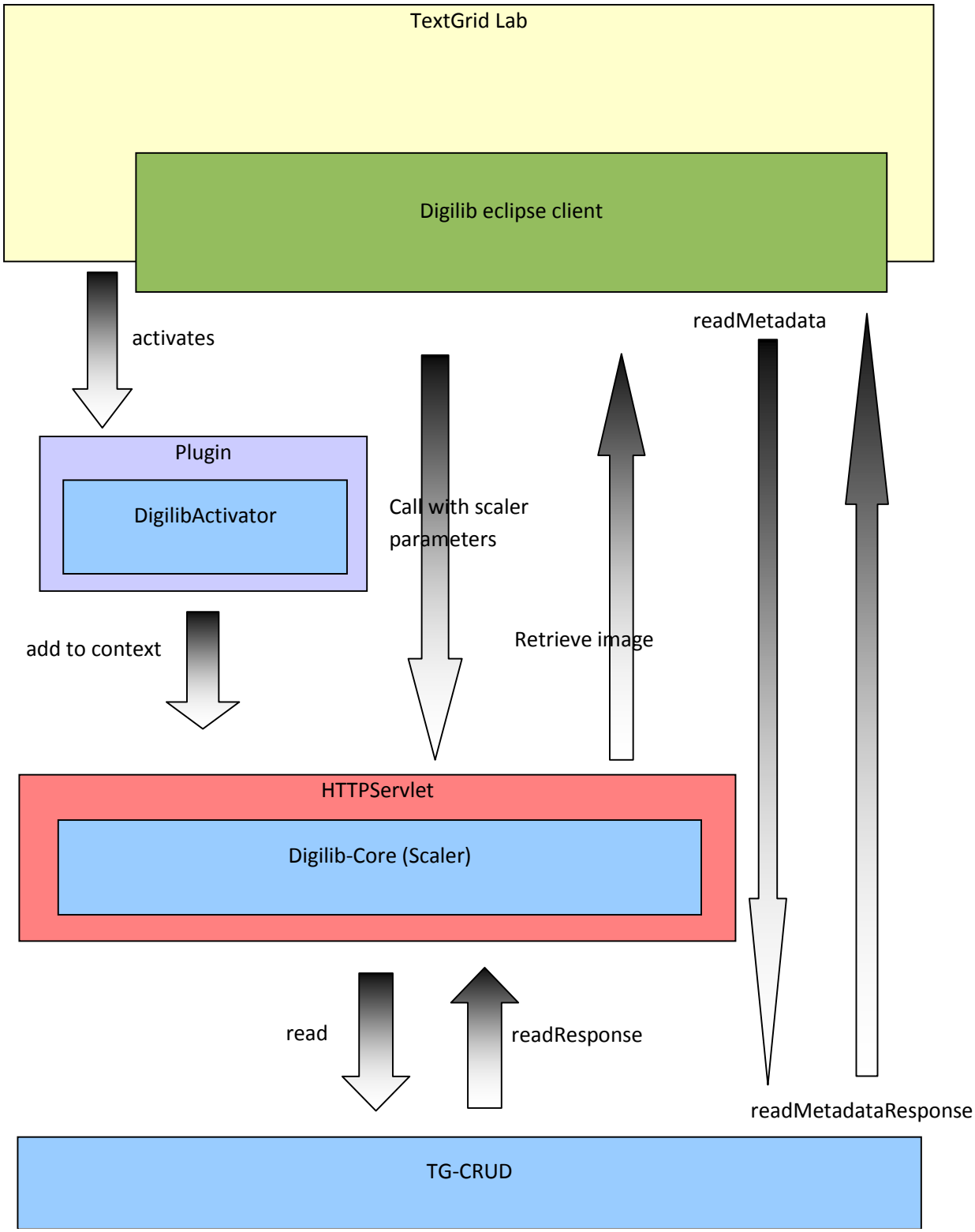
Der Digilib Client steuert den Scaler, indem er ihn mit den erforderlichen Parametern (s.o.) aufruft.

[14] http://digilib.berlios.de/

Der Client ist eclipse-spezifisch, bietet jedoch im Wesentlichen dieselben Funktionen wie der Web-Client.

Zusätzlich soll der eclipse-Client in der Lage sein, bei Bedarf direkt mit TG-CRUD zu kommunizieren und, falls vorhanden, Metadaten anzeigen.

Grafische Übersicht:



4.2. Strukturelle Editionen

Im Bereich *Strukturelle Editionen* wird AP 6.4 u.a. dem Webpublisher (vgl. Abschnitt 1.91.9 oben) zuarbeiten

5. AP 6.5

5.1. Glossen-Editor

5.1.1. Überblick

Glossen-Edition im Rahmen von TextGrid geschieht mit Hilfe zentraler Tools (XML-Editor, Text-Bild-Linkeditor, Text-Text-Linkeditor, Workflow-Editor, Recherche-Tool und Webpublisher) und deren glossenspezifischer Anpassung (durch Stylesheets, XML-Schemata und ein glossenspezifisches Baseline-Encoding). Der Glossen-Editor im engeren Sinne stellt eine Ergänzung dieser Tools für spezielle Bedürfnisse der Glossenedition dar. Dabei handelt es sich um Methoden der Datenerfassung, der Datenaufbereitung und der Visualisierung.

5.1.2. Visualisierung

Das Tool ermöglicht

- die Darstellung von spezifischen Textstrukturen in Glossenkommentaren
    - Textschichtung
    - Text-Kommentar-Beziehungen auf Wortebene
    - kommentarinterne Segmentierung und Sequenzierung
- die synoptische Präsentation verschiedener Glossenkommentare zum gleichen Text
- die synoptische Präsentation von Glossenkommentaren und zugehörigen Handschriftendigitalisaten.

5.1.3. Datenaufbereitung

Es werden Workflows zur Verfügung gestellt, welche die Verknüpfung kommentierter und kommentierender Inhalte auf der Basis vorstrukturierter Texte automatisieren.

5.1.4. Datenerfassung

Über die Datenerfassung mit dem XML-Editor hinaus wird eine vereinfachte Möglichkeit der Datenerfassung über eine Maske angeboten.

5.1.5. Abhängigkeiten von anderen Tools

Der Bereich der Glossenedition ist von den folgenden Tools, insbesondere den genannten Features abhängig:

- XML-Editor
    - Verwendbarkeit spezieller Stylesheets für die Visualisierung im WYSIWYM-Modus
- Text-Bild-Linkeditor

    - Verknüpfung mehrerer Bilder mit einer XML-Datei
    - Text-Bild-Verlinkung ohne Veränderung der Textdatei durch Verwendung vorhandener Elemente mit xml:id
- Text-Text-Linkeditor
    - Standoff-Verlinkung von Textdateien ohne Veränderung der einzelnen Textdatei durch Verwendung vorhandener Elemente mit xml:id
- Workflow-Editor
- Recherche-Tool
- Webpublisher

Der Glossen-Editor nutzt den TUSTEP-Service. Dessen Ausbau für mehrere Eingabedateien ist wünschenswert.

6. AP 6.6

6.1. Noten-Editor

6.1.1. Produktübersicht Noteneditor

Der Noteneditor bietet die Möglichkeit, im TextGrid-Kontext MEI-codierte Notentexte darzustellen und auf einfachem Niveau graphisch zu bearbeiten. Der Anspruch an die Qualität der Darstellung ist deutlich niedriger als bei ausdrücklichen Notensatzprogrammen. Alleinstellungsmerkmale des Noteneditors sind dagegen besondere Fähigkeiten im editorischen Bereich, etwa zur Darstellung von Varianten.

6.1.2. Zielbestimmung

Der Editor dient zwei unterschiedlichen Zielgruppen als Werkzeug: Korrekturleser erhalten eine einfache Möglichkeit zur Anzeige des codierten Notentextes, um eine visuelle Kontrolle der Codierung vornehmen zu können. Editoren hingegen können zügig im WYSIWYG-Stil Änderungen an einem MEI-Dokument vornehmen, wobei für speziellere Eingriffe die direkte Manipulation des Quelltextes über einen XML-Editor erfolgten sollte.

- Gedachtes Verwendungsszenario 1: Bildbetrachter mit Originalhandschrift (oder Notendruck) anzeigen und dazu parallel Neusatz mit dem Noteneditor erstellen
- Verwendungsszenario 2: Anmerkungen anzeigen, einfügen, bearbeiten, löschen
- Verwendungsszenario 3: Mehrere Versionen eines Notentextes „synchronisieren“ und in ein gemeinsames Dokument überführen

Erforderlich

Benötigt wird die Möglichkeit zur on-the-fly-Anzeige von Notentexten der sogenannten Common Western Music Notation (CWMN) als eine alternative Darstellungsform neben der Repräsentation im XML-Editor. Gleichzeitig ist eine Möglichkeit zur Eingabe der Noten in dieser graphischen Ansicht zu entwickeln. Dabei sollen alle gängigen und für eine wissenschaftliche Musikedition notwendigen Bestandteile eines Werks berücksichtigt werden können. Die Satzqualität muss nicht mit kommerziellen Notensatzprogrammen konkurrieren, aber doch ein eindeutiges und klares Notenbild erreichen. Die Bedienung sollte möglichst intuitiv sein, gleichzeitig aber nach Möglichkeit gängige Konventionen aufgreifen. Als zugrundeliegendes Datenformat wird MEI benutzt, allerdings sollte eine Konvertierungsschnittstelle zu

MusicXML möglich sein (siehe Ausschreibung zu AP 6.6.3). Die Darstellung von Varianten ist eine Fähigkeit, die diesen Noteneditor von allen bisher verfügbaren Notensatzprogrammen abhebt.

Weitere Features:

- Implementierung als eigenes view set, das auf Eingabe und Anzeige von Notentexten spezialisiert ist.
- Textuelle Bestandteile einer Partitur (Dynamikangaben, Instrumentbezeichnungen etc.) müssen sowohl aus vorgegebenen Listen ausgewählt als auch frei formuliert werden können.
- Die Taktangabe stützt sich auf ein eingefügtes Symbol mit frei einsetzbaren Ziffern.
- Semantische Constraints (z.B. Anzahl Zählzeiten pro Takt) sind zu ignorieren, um gegebenenfalls „ungültige“ Neusätze (z.B. übervolle Takte) erstellen zu können
- Varianten sind über verschiedene Layer einblendbar. Jede Layer hat eine (wählbare?) Farbe und kann ein- oder ausgeschaltet werden. Eine Layer wird als Referenz ausgewählt, zu der Abweichungen relativ erscheinen.
- Die graphische Benutzeroberfläche benötigt Undo/ Redo Funktionen.
- Das Einfügen von textuellen Kommentaren soll unterstützt werden.
- Die Bedienung des Programms erfolgt sowohl über Tastatur als auch über Mauszeiger.
- Der Noteneditor unterstützt MEI in der Fassung 2010-05. Spätere Änderungen am Schema werden nicht zwingend unterstützt.
- Systeme im Kleinstich (wie eigene Instrumentalsysteme) sollen unterstützt werden.
- Identifikation des Schreibers/Autors (auch innerhalb einer Quelle) als Attribut eines Objektes.
- Der Wirkungsbereich einzelner Objekte (etwa Dynamikangaben) soll klar ersichtlich sein.
- Der Noteneditor soll die nicht von ihm unterstützten MEI-Bestandteilen im Quelltext belassen.

Wünschenswert

Zusätzliche Fähigkeiten wie die Möglichkeit zum Erstellen von Bildschirmfotos und andere Werkzeuge, deren Bedarf sich im Projektverlauf ergibt, sind wünschenswert:

- Verstöße gegen semantische Constraints als Hinweis an Benutzer anzeigen
- Veranschaulichung von Unterschieden zwischen Varianten, etwa als Visualisierung eines Stemmas (Abweichungen und Stammbäume von Quellen – cf. Bindefehler & Trennfehler)
- Akustische Umsetzung der Dateien bleibt im Kontext des Projekts unwichtig, ggf. kann ein MIDI-Support erfolgen.
- Generalbaß-Notation
- Seitenbasiertes Arbeiten im Hinblick auf Wiedergaben in Druckform.
- Textgenetische Eingriffe im Text dokumentieren (Streichungen, Hinzufügungen, Vide-Vermerke, Rasuren, Überklebungen etc.)

Transkription existierender Quellen

Der Noteneditor sollte unter anderem in der Lage sein, die in MEI importierten Quelltexte von KernScores (http://kern.ccarh.org/) zu interpretieren und dem Code entsprechend darzustellen.

Verfassen neuer MEI-Dokumente

Zur Unterstützung von editorischer Quellenarbeit sind Wizards hilfreich, die zum Erstellen von neuen MEI-Dokumenten mit geringem Benutzeraufwand passende MEI-Dokumentgerüste anfertigen und dem Benutzer so Aufwand ersparen. Der Import von MusicXML Dateien wird unterstützt, indem die passenden XSLT-Skripte (von Perry Roland u.a.) eingebunden werden.

Abgrenzungen

Ein Export nach MusicXML wird über ein XSLT realisiert, welches von Perry Roland (Virgina University) zur Verfügung gestellt wird. Dazu muss ggf. bei mehreren enthaltenen Fassungen eine eindeutige Auswahl getroffen werden, da die Codierung von Mehrdeutigkeit in MusicXML nicht unterstützt wird. Eine dazu notwendige graphische Oberfläche kann im Rahmen des Noteneditors nur mit eingeschränkter Funktionalität entwickelt werden.

6.1.3. Produktfunktionen

Unterstützung von MEI

Die Verwendung von MEI als natives Datenformat ist notwendig.

Einbindung in TextGrid

Eine Einbindung in TextGrid und die Möglichkeit zur Verwendung des TextGrid Speichersystems zum kollaborativen Arbeiten auf MEI-Dokumenten ist notwendig.

Visualisierung des MEI-Dokuments

Das XML Dokument soll graphisch angezeigt werden in Form eines Neusatzes in CWMN, wobei für manche Elemente entsprechend notwendige Darstellungsformen ermittelt werden sollen. Änderungen am Notentext sollen unmittelbar dargestellt werden, so dass ein WYSIWYG-Konzept realisiert werden kann. Die intuitive Darstellung von Varianten soll es u.a. ermöglichen, unterschiedliche Fassungen eines Notentextes gegenüberzustellen und in Beziehung zu setzen.

Unterstützung bei der Orientierung im Dokument

Da das MEI-Dokument einen großen Umfang erreichen kann, ist eine Unterstützung des Benutzers durch eine logische sowie eine grafische Navigationsstruktur wünschenswert.

Arbeitsbeschleunigung für erfahrene Benutzer

Zur Beschleunigung von sich wiederholenden Aufgaben ist eine Lösung wünschenswert, die diese Arbeit für erfahrene Benutzer durch Abkürzungen und aufgabenorientiert strukturierte Eingabepfade beschleunigt.

6.1.4. Produktleistungen

Die Ausgabe des Noteneditors soll zu jedem Zeitpunkt des Bearbeitungsprozesses eine wohlgeformte MEI-Datei sein, die gegen das im Programm angegebene MEI-Schema validiert.

6.1.5. Benutzeroberfläche

Die Benutzeroberfläche sollte sich an gängigen Usability-Kriterien orientieren (vgl. etwa ISO 9241, Teil 10) und für Textwissenschaftler, Bearbeiter und technische Betreuer unterschiedlicher Computerexpertise benutzbar sein. Eine Anlehnung der GUI an die von gängigen Notensatzprogrammen her bekannten Oberflächen wird angestrebt, soweit dies mit der spezifischen Funktionalität eines XML-Editors vereinbar ist. Eine enge Integration in die Rich Client Platform und mit den anderen Tools der TextGrid-Workbench ist wünschenswert.

6.1.6. Nichtfunktionale Anforderungen

Die Software soll die gängigen Standards im XML-Bereich nicht verletzen und Unicode-Daten im MEI-Quelltext berücksichtigen. Sie muss auf allen Plattformen laufen, auf denen die Rich Client Platform läuft, mindestens auf aktuellen Linux-, Macintosh- und Windows-Plattformen.

6.1.7. Technische Produktumgebung

Die Implementierung erfolgt in den gleichen technischen Rahmenbedingungen wie die übrige Client-Software des TextGrid Projektes, also Java 1.5 auf der Rich Client Platform auf Basis von Eclipse.

7. AP 6.8

7.1. OCR

7.1.1. Produktübersicht OCRService

Der OCRService ist ein Dienst, der die Bilder übernimmt und den Textinhalte zurück gibt. Der OCRService bietet die Möglichkeiten für TextGrid-Clients eins oder mehrer Bilder zu dem Open-Source System OCRopus (www.ocropus.org) zu schicken. Das Ziel ist die Entwicklung eines hochwertigen, leistungsfähigen Systems zur Erkennung von Fraktur. TextGrid-Objekte werden von TG-crud ins Grid eingeschickt. Mit dem Hilfe von SOAP (Simple Object Access Protocol) werden die TextGrid-Objekte zu OCRopus übergetragen. Das Ergebnis wird als Dienst für TextGrid verfügbar gemacht, um Endnutzern die Erkennung von historischen deutschen Dokumenten zu ermöglichen.

7.1.2. Zielbestimmung

Musskriterien

Das System wird einspaltige, verzerrungsfreihe Frakturtexte in 11pt oder größer auf sauberen, konstrastreichen mit 600 dpi eingescannten A4-Seiten erkennen. Das OCR-Tool ist als Web Service gekapselt, und ist in den TextGridLab aufgerufen.

Wunschkriterien

- Fehlerrate < 2%
- Erkennung auf Dokumenten von 300dpi Auflösung

- Erkennung von Dokumenten minderer Qualität
- Erkennung von zweispaltigen Frakturtexten
- Der TextGrid-Client kann der Bild von dem Navigator zu dem Workflow einfugen und führt den OCR-Workflow durch. Der Ergebnis ist als Textformat abgeliefert.
- Möglichkeiten zur Adaption des Erkenners und der Sprachmodelle basierend auf externen Tools.

Abgrenzungskriterien

Die folgenden Ziele sind nicht als Teil dieses Projektes geplant:

- Benutzeroberflächen zur OCR Korrektur
- Benutzeroberfläche Bildvorverarbeitung
- Integration von interaktiven Adaptions- und Sprachmodellierungstools in die TextGrid Benutzeroberfläche.

7.1.3. Produkteinsatz

OCR-Tool ist als Web Service gekapselt.

7.1.4. Produktfunktion

Das OCR-System wandelt Dokumentbildsammlungen von Frakturtexten in Unicode Texte um. Das System wird als Webservice zur Verfügung gestellt. Einbindung nach Textgrid erfolgt durch standard Textgrid tools (Workfloweditor usw). Der Workflow-Editor implementiert, soweit vorhanden, die Eclipse- bzw. RCP-eigenen Schnittstellen, um eine möglichst enge Integration in die Rich Client Platform bzw. mit anderen auf dieser Plattform aufbauenden Tools zu realisieren. Der Benutzer kann die Bilder von den Navigator zum OCR-Workflow einfügen. Der OCR-Workflow wird ausgeführt und soll über in den SOAP-Messages übergebene Parameter (Bild-Datei) erfolgen. Der Ergebnis ist als Textformat abgeliefert.

7.1.5. Produktdaten

- Welche Daten sind aus Benutzersicht (langfristig) zu speichern : Quelldatei, Zieldatei.
- Daten-Format: PNG, JPEG, TIFF, und HTML Format.

7.1.6. Produktleistung

Das System soll vernünftige (reasonable) Genauigkeit für die Frakturerkennung geben. Es kann verschiedene Bildformate (PNG, JPEG, und TIFF) berabeiten.

7.1.7. Benutzeroberfläsche

Der OCR Service wird durch von aneren Teilnehmern entwickelte Standardfunktionen im TextGrid aufgerufen (Workflow Editor, Dokumentselektor usw).

7.1.8. Nichtfunktionale Anforderungen

Plattformunabhängig auf Client-Seite, da als WebService implementiert Welche Nutzungsdaten sollen (oder dürfen nicht) von der Middleware erhoben werden . Logging, Durchreichen von Benutzerdaten, siehe Report 3.2 „TextGrid Architektur“.

7.1.9. Technische Produktumgebung

Software Java, Eclipse, Apache Axis2, WebService, also SOAP, WSDL; mittelfristig BPEL,C++,Python.

Hardware Es gelten die Anforderungen der Rich Client Platform. Aus Performancegründen ist einigermaßen aktuelle Hardware wünschenswert. Produktschnittstellen Service ist gegenwärtig zugreifbar unter http://131.246.165.190:8180/axis2/services/OCRService?wsdl

Parameter in_image Quelldatei und ocr_output Zieldatei