Report über die AP6-Entwicklung des vergangenen Jahres sowie den Stand der Planungen (R 6.0.2)

Version 2011-10-31
Arbeitspaket 6
verantwortlicher Partner Universität Würzburg

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

Dokumentstatus: Final
Verfügbarkeit: öffentlich
Autoren:

Mareike Laue, UWÜ
Thorsten Vitt, UWÜ
Toolverantwortliche, vgl. die Tabellen am Beginn jeder Toolbeschreibung

Revisionsverlauf:

Datum	**Autor**	**Kommentare**
	Mareike Laue	Zusammenstellung
31.10.2012	Thorsten Vitt	Einleitung, Überarbeitung

Inhaltsverzeichnis

0 Einleitung

Die Entwicklungen im vergangenen Jahr standen vor allem unter dem Zeichen der Produktivversion 1.0 des TextGridLab, die zum Digital-Humanities-Festakt am 12. Juli 2011 veröffentlicht wurde. Naturgemäß konzentrierten sich die Entwicklungen deshalb vor allem auf diejenigen Tools, die in jener Version integriert waren, und dabei besonders darauf, die Tools möglichst stabil und bug-arm zu bekommen, sowie sie an die Neuerungen im TextGridRep anzupassen.

Um Redundanzen zum vorherigen Report zu vermeiden, enthält dieser Bericht für die meisten Tools nur eine Kurzzusammenfassung (Toolstandstabelle), die aus dem TextGrid-internen Wiki übernommen wurde, sowie ggf. kurze Angaben zu wesentlichen Neuerungen.

Das *Import-/Exportwerkzeug* für das TextGridLab wurde von Grund auf neu entwickelt, hierzu ist deshalb in Abschnitt 1.6 eine vollständigere Beschreibung vorhanden. Für den *Text-Text-Linkeditor* wurde unter Mitwirkung externer Forschungsprojekte als potentieller Nutzer ein Pflichtenheft neu entwickelt, das entsprechend hier in Abschnitt0abgedruckt ist. Auch für den Noteneditor gibt es ein komplettes Pflichtenheft (Abschnitt 6.1).

1 AP 6.1

1.1 Metadaten – Editor/ - Annotationen

Verantwortlich	Yahya Al-Hajj
Arbeitspaket	AP6.1
Entwicklung	Jira-Komponente • Subversion • Entwicklerdoku
Grundfunktionalität	siehe Metadaten-Editor/Pflichtenheft
TODOs bis zur Version 1.0 (Features)	• Anpassung an das neue Metadatenmodell
Produktionsreife	100% im Monat 01.2010
Einschätzung Entwickler	100% Produktionsreife erreicht
Teststatus	Der Metadaten-Editor ist in vielen Tools im Lab integriert und wird deswegen viel getestet.
Abhängig von ...	• Navigator, XML-Editor
abhängige Tools /Services	• Aggregation, New-Dialog, Import-Dialog
Sonstiges	—
letztes Update dieser Tabelle	2011-08-17

Der Metadateneditor erfuhr vor allem durch die komplette Überarbeitung des TextGrid-Metadatenmodells eine grundlegende Überarbeitung.

1.2 XML-Editor

Verantwortlich	Thorsten Vitt
Arbeitspaket	AP6.1
Entwicklung	Jira-Komponente • Subversion • Entwicklerdoku
Grundfunktionalität	siehe XML-Editor/Pflichtenheft
Kernaufgaben bis zur Version 1.0 (erreicht)	• Annotationen im WYSIWYM-Modus • Unterstützung für eigene Stylesheets des Users • Höhere Geschwindigkeit bei Dateien mit großen Inline-Tags • Architektonische Abtrennung des ressourcenintensiveren WYSIWYM-Modus vom Rest • WYSIWYM-Modus wird jetzt nur aktiviert, wenn sinnvolle Voraussetzungen dafür erfüllt sind
Produktionsreife	100% im Monat 2011-05
Einschätzung Entwickler	100% Produktionsreife erreicht
Teststatus	• Ausführliche Tests u.a. durch Mirjam Blümm und Fotis Jannidis liegen vor • Regelmäßiger Einsatz z.B. beim Detmolder Musikwissenschaftsprojekt
Abhängig von ...	• TextGridLab
abhängige Tools /Services	• Text-Bild-Link-Editor
Sonstiges	—
letztes Update dieser Tabelle	2011-09-16

1.3 Link-Editor Text

Verantwortlich	Thomas Selig
Arbeitspaket	AP 6.2
Grundfunktionalität	siehe Pflichtenheft unten
TODOs bis zur Version 1.0 (Features)	Nicht Bestandteil von Version 1.0
Produktionsreife	100% im Monat 2012-03
Einschätzung Entwickler	5% Produktionsreife erreicht
Teststatus	Geplant sind automatisierte Tests mittels Sikuli
Abhängig von ...	
abhängige Tools /Services	• TextGridLab
Sonstiges	—
letztes Update dieser Tabelle	2010-08-31

1.3.1 Produktübersicht Link-Editor Text

Der Link-Editor Text dient als Eingabehilfe für Links in XML-Dateien und verbindet Elemente von Recherchetool/Navigator und XML-Editor. Benutzer haben die Möglichkeit, Verknüpfungen zwischen beliebigen Elementen von zumindest für den Benutzer lesbaren TextGrid-Dokumenten in einer TTLE - TEI-Datei zu erstellen. Darüber hinaus wird der Link-Editor Text zur Validierung bereits existierender Verknüpfungen 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.

Die Darstellung verknüpfter Dokumente soll darauf achten, dass verbundene Elemente immer auf gleicher Höhe stehen (notfalls wird Whitespace eingefügt) (synoptische Ansicht).

Die einzelnen Links sollen typisierbar sein. Hierfür stehen sowohl feste Grundtypen als auch freie Typbezeichnungen zur Verfügung. Unterschiedlichen Typen sollen unterschiedliche Farben zugewiesen werden können.

1.3.3 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. Bei der Erstellung der Links müssen unterschiedliche Szenarien berücksichtigt werden:

1. Dokument auf welches der User Schreibrechte besitzt: Hier werden an den entsprechenden Stellen im XML eigene Tags zur Kennzeichnung von Beginn und Ende eines verknüpften Abschnittes eingefügt.

2. Publizierte Dokumente: Da publizierte Dokumente nicht mehr verändert werden können, wir hier in der TTLE Datei Beginn und Ende des verknüpften Abschnittes mittels absoluten Koordinaten (Zeile/Spalte) gespeichert.

3. Nicht publizierte Dokumente auf die der User lediglich Lesezugriff besitzt: Hier können keine beliebigen Textstellen für eine Verknüpfung ausgewählt werden. Es ist lediglich möglich, Verknüpfungen auf XML-Elemente mit eindeutiger ID zu erstellen. Alternativ kann angeboten werden, eine Kopie des Dokumentes zu erstellen, in welcher dann entsprechend Punkt 1 gearbeitet werden kann.

Die Verlinkung wird im Datenmodell über eine 1:n-Verknüpfung der Daten realisiert. Der ebenfalls häufig benötigte Fall einer 1:1-Verlinkung ist damit ebenfalls abgedeckt.

Die Visualisierung der verlinkten Dokumente soll durch die gleichzeitige Anzeige von jeweils zwei Dokumenten erfolgen. Dabei soll auf einfache Weise zwischen an einem Link beteiligten Dokumenten hin- und hergeschaltet werden können. Auch neue Dokumente müssen in dieser Ansicht angezeigt werden können, um neue Links erstellen oder existierende Links erweitern zu können. Bei der Anzeige sollen sich verlinkte Textfragmente immer auf gleicher Höhe in den Anzeigenfenstern befinden. Dies ist jedoch nur möglich, wenn sich die verlinkten Fragmente in unterschiedlichen Dokumenten befinden. Wenn diese synoptische Darstellung eine Änderung an der Struktur des Dokumentes voraussetzen würde, wird auf die synoptische Darstellung verzichtet. Die Visualisierung benötigt zusätzlich eine Navigations-Komponente, in der die Links sowie die in den einzelnen Links verlinkten Textfragmente (Baumansicht) angezeigt werden. Diese Baumstruktur soll nach unterschiedlichen Kriterien sortierbar sein.

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

Die Links sollen typisierbar sein. Jedem Typ kann dabei eine Farbe zugewiesen werden, mit welcher die Textfragmente in den Anzeigefenstern hinterlegt werden. Dies erlaubt es, möglichst schnell zusammengehörige Textfragmente zu identifizieren. Zwei Arten von Linktypen werden benötigt: Freiform-Typen und vordefinierte Typen. Den Freiform-Typen kann ein beliebiger (nicht durch einen vordefinierten Typ belegter) Name und ein einzelner Text (z.B. für einen Kommentar) zugewiesen werden. Die vordefinierten Typen haben einen festen Namen und besitzen eine feste Menge vorgegebener Attribute, die mit Text belegt werden können/müssen. Die vordefinierten Typen gehören dem jeweiligen Nutzer, der diese erstellt hat. Er kann diese Typen einzelnen Projekten zur Verfügung stellen. Wenn ein Linktyp einem Projekt zur Verfügung gestellt wurde, können alle Mitarbeiter des Projektes den Linktyp in diesem Projekt einsetzen. Diese restriktive Politik soll verhindern, dass Linktypen aus sehr langen Listen, mit unter Umständen ähnlichen oder identischen Namen aufwendig herausge-

sucht werden müssen. Die vordefinirten Linktypen werden im RDF-Format zentral in einem Triple-Store abgelegt.

Ein sehr einfach gehaltener Editor zur Erstellung von Linktypen, deren Upload in den Triple-Store sowie deren Zuweisung zu Projekten wird ebenfalls benötigt.

Textfragmente in beschreibbaren Dokumenten können noch verändert werden, nachdem diese Textfragmente in einen Link aufgenommen wurden. Unter Umständen kann dies die Verlinkung hinfällig machen. Beim Öffnen eines TTLE - Dokumentes muss deshalb die Integrität der Links sichergestellt werden. So könnten beispielsweise Start- oder End-Tags zur Kennzeichnung von Fragmenten entfernt worden sein, oder ein Textfragment könnte inhaltlich verändert worden sein (wird durch Hashsummen der Fragmente überprüft). In beiden Fällen muss der Nutzer über die Veränderung informiert werden.

1.3.4 Wunschkriterien

Das Suchen nach ähnlichen Textfragmenten in Kollektionen oder Editionen mittels unterschiedlicher Suchalgorithmen ist wünschenswert. Da dies je nach Umfang einer Kollektion sehr zeitaufwendig sein kann, sollte dieser Aspekt des TTLE in einen serverbasierten Web-Dienst ausgelagert werden. Das UI muss deshalb Möglichkeiten anbieten um solche Suchaufträge zu erstellen und abzuschicken, sowie um später die Ergebnisse abzuholen und anzuzeigen.

Häufig präsentieren Wissenschaflter ihre Ergebnisse auf eigenen Web-Portalen. Um eine einfache Integration ihrer mit Textgrid erzielten Ergebnisse in diese Portale zu erlauben, wäre ein API wünschenswert, über welches aus externen Applikationen direkt auf Links und deren verlinkte Textfragmente zugegriffen werden kann. Dies könnte entweder über einen Export der Daten aus Texgrid heraus oder über eine entsprechende Web Service Schnittstelle erfolgen.

Eine Visualisierung aller verlinkten Dokumente eines Projektes würde einen guten Überblick über das gesamte Projekt ermöglichen, während eine Darstellung aller Dokumente eines bestimmten Link-Typs die Entwicklung eines Dokumentes sehr gut verdeutlichen könnte. Beide Arten der Visualisierung wären für den TTLE wünschenswert.

Vordefinierte Typen sind ein wichtiger Bestandteil des TTLE. Ein kompfortabler Editor zur Verwaltung dieser Link-Typen wäre deshalb wünschenswert. Wichtig wären Funktionen zum Gruppieren von Links, zur Weitergabe von Link-Typen und Typgruppen an andere Personen, zur öffentlichen Bereitsstellung von Link-Typen und Typengruppen, sowie zur Veränderung von Link-Typen, die im Einsatz sind und zur Transformation eines Link-Typs in einen anderen.

1.3.5 Anforderungsquellen

- Blumenbach-Online**:**
  - 1:1-Verknüpfungen (eine Textpassage mit einer einzigen anderen Textpassage)
  - 1:n, multiple Verknüpfungen (eine Textpassage mit mehrerer anderen Textpassagen), ebenso der Text-Bild-Link-Editor
  - Verknüpfungen zwischen Textpassagen innerhalb eines Werks

    - Verknüpfungen zwischen Textpassagen verschiedener Werke
    - Aufgabe des TTL-Editors ist, die entsprechenden Auszeichnungselemente ("tags") standardisiert und komfortabel in die Volltexte einfügen zu können.
    - Da bei Blumenbachs Schriften auch Überlieferungs- bzw. rezeptionsgeschichtliche Fragen behandelt werden, wäre es hilfreich, wenn der TTLE auch typisierte Links unterstützen würde.
    - Visualisierung und Suche: Besteht die Möglichkeit diese Beziehungen in einem chronologischen Raster abzubilden. D. h. lässt sich bspw. beim Text-Text-Link-Editor die Text-Genese abbilden. (Realisierung über Metadaten, z. B. Unterscheidung zwischen Entstehungs- und Publikationszeitraum). Von Interesse sind hierbei vor allem die Darstellung von Filiationen, Manifestationen, Items etc.
- SAWS:
    - Texteinheiten: "Chunk of text (WITH CERTAIN PROPERTIES) namely, in the case of HyperHamlet: chunk of pre- or post-Hamlet text with some linguistic and/or semantic overlap with chunk(s) in Hamlet" → d.h. unter Gesichtspunkten der Textkritik müssen mitunter Texteinheiten ausgezeichnet und adressiert werden können, die nicht notwendigerweise semantische oder syntaktische Einheiten sind, sondern diese auch überlagern können
    - Beziehungen: "Parallels: Extant source/partner [?Translation, Verbatim, Variant] vs. Not extant [Claims (quality)]" → Set an Beziehungen, die durch Attribute o.ä. näher zu beschreiben sind.

→ Letztendlich benötigen beide Projekte die Möglichkeit, typisierte Links zu setzen und zu verwalten, um überlieferungsgeschichtliche Prozesse (im Rahmen der Textkritik) abbilden zu können

→ Idealerweise sollten die entsprechenden Attribute vom Adaptor-Manager in RDF-Tripel aufgelöst und in die TG-RDF-Datenbank geschrieben werden

→ Zur Verwaltung der hierbei eingesetzten Vokabulare bietet sich CONE an, das auch in anderem Zusammenhang in TextGrid eingesetzt werden wird

1.3.6 Produktleistungen

Siehe Muss- und Wunschkriterien.

1.3.7 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.

1.3.8 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

Es handelt sich hierbei um keine Middleware-Komponente.

1.3.9 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 Lemmatisierer

Verantwortlich	Andreas Witt
Arbeitspaket	AP6.1
Entwicklung	Jira-Komponente • Subversion • Entwicklerdoku
Grundfunktionalität	siehe Lemmatisierer/Pflichtenheft
TODOs bis zur Version 1.0 (Wunschmerkmale)	Nicht bestandteil von Version 1.0
Produktionsreife	100% im Monat ?
Einschätzung Entwickler	80% Produktionsreife erreicht
Abhängig von ...	• TextGridLab • Tokenizer
abhängige Tools /Services	
Sonstiges	—
letztes Update dieser Tabelle	20.09.11

1.5 Streaming-Editor XSLT, für XML

Verantwortlich	Thorsten Vitt
Arbeitspaket	AP6.1
Entwicklung	Jira-Komponente • [Subversion] • [Entwicklerdoku]
Grundfunktionalität	siehe
TODOs bis zur Version 1.0 (Features)	• Auflösung von TextGrid-URIs; Absprache mit TextGrid-URI-Resolver • Tests • Dokumentation
Produktionsreife	100% im Monat 2010-12
Einschätzung Entwickler	90% Produktionsreife erreicht, Tests fehlen noch
Abhängig von ...	• Workflow zum Ansprechen • Multiresolver zum URIs auflösen
abhängige Tools /Services	• div. Streamingtools: Vorverarbeitung • Workflow
Sonstiges	da das Workflow-Tool aus Version 1.0 herausgenommen wurde und so die vorgesehene Ansprechmöglichkeit für das Tool fehlte, ist der Streaming-Editor zwar eigentlich fertig, aber in den Tests und in der Produktivversion nicht berücksichtigt worden.
letztes Update dieser Tabelle	2011-08-12

1.6 Import-/Export-Tool (TextGridLab)

Verantwortlich	Vitt
Arbeitspaket	AP1 / AP6
Entwicklung	[Jira-Komponente] • [Subversion] • [Entwicklerdoku]
Grundfunktionalität	• Import einer überschaubaren Menge von Daten via Lab nach TextGrid • simple! Massenvergabe von Kernmetadaten • Linkauflösen mit der Linklibrary
TODOs bis zur Version 1.0 (Features)	Grundfunktionalität
Produktionsreife	100% im Monat 2011-05
Einschätzung Entwickler	Produktionsreife erreicht
Abhängig von ...	• Linklibrary • Metadaten-Editor
abhängige Tools /Services	
Sonstiges	Basis ist die TG-crud-EFS-Implementierung im Lab
letztes Update dieser Tabelle	2010-11-15

1.6.1 Produktübersicht Lab-Import/Export

Das **Importtool** soll es Lab-Anwendern ermöglichen, selbständig vorhandene eigene Daten zur weiteren Verarbeitung über das TextGridLab in den dynamischen Bereich des Text-GridRep zu importieren. Analog dazu soll ein **Exporttool** den Export von Objekten aus dem TextGridRep (mitsamt deren Metadaten) auf die Festplatte des Users ermöglichen.

1.6.2 Zielbestimmung

Das **Importtool** soll es den Benutzern möglichst einfach machen, vorhandene Dateien nach TextGrid zu importieren, und zwar sowohl neue Daten, die noch nie in TextGrid waren, als auch Daten, die zuvor aus TextGrid exportiert worden und ggf. mit loklen Tools bearbeitet worden sind.Metadaten sollen dabei wahlweise ebenfalls mitgegeben oder in einem Kernsatz generiert werden können.

Das **Exporttool** soll es Benutzern einfach ermöglichen, Daten und Metadaten aus TextGrid auf die lokale Festplatte zu exportieren, ggf. zum späteren Re-Import.

Bei beiden Tools ist zu beachten, dass Links zwischen gemeinsam importierten Objekten ggf. zwischen lokalen Datei- und Pfadangaben und TextGrid-URIs übersetzt werden müssen. Die Eingriffe in die zu im- bzw. exportierenden Daten sollten dabei möglichst kontrolliert bleiben.

1.6.3 Musskriterien

Import

- Import einer (überschaubaren) Menge von Daten aller Art nach TextGrid
- Metadateneingabe für die zu importierenden Daten möglich
- Der Benutzer muss nicht für jedes zu importierende Objekt Metadaten eingeben, um die Daten importieren zu können
- Umschreiben von relativen Links in solche mit textgrid:-URI beim Import für unterstützte Dateitypen, mindestens:
    - TEI P5
    - Text-Bild-Linkeditor-Daten
    - XML-Schemata
    - Aggregationen (Editionen, Kollektionen)
- Import von Daten, die aus früheren Beta-Versionen des Lab exportiert worden sind (vor Release von TextGrid 1.0)

Unterschiedliche Importszenarien:

1. neue Dateien, die noch nicht in TextGrid waren – hier sollten möglichst (vorläufig) sinnvolle Metadaten und ein sinnvolles Importverhalten »erraten« werden
2. Dateien, die (mitsamt ihren Metadaten) zuvor aus TextGrud exportiert worden sind. Hier sollen die alten Metadaten ebenfalls übernommen werden. Zudem gibt es hier nach Wahl des Nutzers unterschiedliche Möglichkeiten des Re-Import:
    1. als neue, unabhängige TextGrid-Objekte
    2. als neue Revisionen der ursprünglichen Objekte, falls der importierende Benutzer das Recht hat, neue Revisionen der ursprünglichen Objekte anzulegen

Export

- Export einer (überschaubaren) Menge von Daten aller Art aus TextGrid
- Umschreiben von Links mit textgrid:-URI beim Export für unterstützte Objekttypen (vgl. Import) in relative Dateinamen

Beide

- Reimportszenario: Exportierte Dateien mit lokalen Tools bearbeiten und wieder importieren
- Einbindung in die sonstige Benutzungsoberfläche
- Fortschrittsanzeige etc.

1.6.4 Wunschkriterien

- Einfache eigene Spezifikation der Dateitypspezifikationen fürs Rewriting
- Ggf. Anpassung der Dateinamen beim Export

1.6.5 Produktleistungen

Siehe Muss- und Wunschkriterien.

1.6.6 Benutzeroberfläche

Die Benutzeroberfläche sollte sich in das Look ’n’ Feel des TextGridLab und die Funktionalität des Betriebssystems integrieren. Das Hinzufügen von Dateien zum Import erfolgt wahlweise per Drag'n'Drop aus den Dateiverwaltungstools des Betriebssystems oder über Dateiauswahldialoge des Betriebssystems.

Die Benutzungsschnittstelle ist so organisiert, beim Hinzufügen von Dateien oder Objekten über Heuristiken versucht wird, möglichst viel Konfigurationsarbeit etc. den Benutzern abzunehmen – existieren Metadatendateien, werden die Metadaten daraus gelesen, Verzeichnisse werden beim Import zu Aggregationen, Rewriting-Regeln werden sinnvoll gesetzt, Metadaten werden ggf. aus dem Dateinamen etc. auf Anfangswerte gesetzt.

1.6.7 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

Es handelt sich hierbei um keine Middleware-Komponente.

1.6.8 Technische Produktumgebung

Software

Der Client wird im Rahmen des TextGridLab auf der Basis von Eclipse implementiert. Eine Serverkomponente ist nicht vorhanden; es wird über die allgemeinen im TextGridLab eingebauten Schnittstellen mit dem TextGridRep kommuniziert.

Produktschnittstellen

Import- und Exporttool implementieren, soweit vorhanden, die Eclipse- bzw. Lab-eigenen Schnittstellen, um eine möglichst enge Integration in das TextGridLab bzw. mit anderen auf dieser Plattform aufbauenden Tools zu realisieren.

1.7 Tokenizer*

Verantwortlich	Ubbo Veentjer
Arbeitspaket	AP6.1
Entwicklung	• Jira-Komponente • Subversion • Lab-Komponente • Service-Komponente
Grundfunktionalität	siehe Tokenizer/Pflichtenheft
TODOs	• Namespace-Unterstützung • Anderer SAX-Parser (derzeit 4Suite) • TG-610 • TG-661 • Umstellung auf REST(?)
Produktionsreife	100% im Monat 2012-03
Einschätzung Entwickler	85% Produktionsreife erreicht
Teststatus	• ...
Sonstiges	* Es gibt Überlegungen den Python basierten Tokenizer durch den GATE Tokenizer als CXF Webservice zu ersetzen. Siehe Tokenizer/Gate.

1.8 Bibliographie-Tool

Verantwortlich	Thorsten Vitt
Arbeitspaket	AP6.1
Entwicklung	[Jira-Komponente] • [Subversion] • [Entwicklerdoku]
Grundfunktionalität	siehe Pflichtenheft unten
Produktionsreife	100% im Monat 2011-11
Einschätzung Entwickler	5% Produktionsreife erreicht
Teststatus	kein Code, keine Tests
Abhängig von ...	• TextGridLab
abhängige Tools /Services	
Sonstiges	—
letztes Update dieser Tabelle	2010-10-06

1.9 Web Preview (fka Webpublisher)

Verantwortlich	Ubbo Veentjer
Arbeitspaket	AP6.1

Entwicklung	• Jira-Komponente •
	• Subversion
	  • Lab-Komponente
	  • Service-Komponente
	• Entwicklerdoku (FIXME)

Grundfunktionalität	siehe Webpublisher/Pflichtenheft

TODOs	• kleinere Reparaturen
	  • Links in HTML Ansicht reparieren (im XSLT)
	  • evt. "schönere" URIS (projects/TGPR45/publ/123123 statt ?project=TGPR45&publ=123123)
	• Zugriffssicherung (sortiert nach Implementierungsaufwand)
	  • Verzeichnisliste rausnehmen (da UUID schwer zu erraten), für Publikationen, die allgemein zugänglich sein sollen, kann ein NOID(-TextGrid)-URI vergeben werden.
	• Darstellung von Textbildlinkeditor-Files (TBLE-Files)
	  • Drag'n'Drop von TBLE-File in Webpublisher-View
	  • ⇒ Publisher publiziert zugehörige Bilder & XML-Dateien
	  • Anpassung XSLT-Stylesheet für Darstellung von TBLE-Files
	  • Darstellung Bilder mit IIP-Imageserver (Zoom), Javascript für klickbare Boxen im Bild
	  • zugehörige TEI Darstellung, per Javascript verknüpft mit Bild
	• Zugriff auf Original-XML

Produktionsreife	75% im Monat 2010-11
Einschätzung Entwickler	50% Produktionsreife erreicht

Teststatus	• erste Tests durch Fotis Jannidis (2009) • ...
Abhängig von ...	•
abhängige Tools /Services	•
Sonstiges	Evtl. gemeinsame Weiterentwicklung / Zusammenführung mit SADE (BBAW)
letztes Update dieser Tabelle	2010-10-06

2 AP 6.2

2.1 Text-Bild-Link-Editor

Verantwortlich	Yahya Al-Hajj
Arbeitspaket	AP6.2
Entwicklung	Jira-Komponente • Subversion • Entwicklerdoku LinksObjekt-Beschreibung
Grundfunktionalität	siehe Text-Bild-Link-Editor/Pflichtenheft
TODOs bis zur Version 1.0 (Features)	• -
Produktionsreife	100% im Monat 01.2011
Einschätzung Entwickler	100% Produktionsreife erreicht
Einschätzung Fachwissenschaftler	75% (Simon Rettelbach, 2010-10-06) [Bedenke: Abhängigkiet neue URIs und Metadatenschema (PROBLEM GELÖST)]
Teststatus	• Der Text-Bild-Link-Editor wird von den Entwicklern immer noch entwickelt und modifiziert und dabei stetig getestet. • Diverse Tests auch von Fachwissenschaftlern wie z.B. Simon Rettelbach und dem Faustedition-Projekt
Abhängig von ...	• XML-Editor • Navigator
abhängige Tools /Services	• Glossen-Editor
Sonstiges	—
letztes Update dieser Tabelle	2011-08-17

2.2 Kollationierer

Verantwortlich	Mohamadou Nassourou
Arbeitspaket	AP6.2
Entwicklung	
Grundfunktionalität	• Auswahl der zu kollationierenden Texte • interaktive Definition von allgemeinen und lokalen Identitätsregeln • Definition von Ausnahmeregeln anhand des Kollationierungsergebnis und erneute Kollationierung • Verwaltung von Konfigurationen und Ausnahmeregeln in TextGrid"
Produktionsreife	
Einschätzung Entwickler	100% Produktionsreife erreicht
Teststatus	• Tests durch Werner Wegstein und Fotis Jannidis
Abhängig von ...	• CollateX
abhängige Tools /Services	•
Sonstiges	In TextGrid liegt der Fokus insbesondere auf der Entwicklung eines User Interface zu Vorbereitung und Ansprechen des Kollationierungsvorgangs. während die Softwarebibliothek zur Kollationierung und zur Darstellung der Ausgabe vom EU-Projekt Interedition übernommen wurde
letztes Update dieser Tabelle	2011-08-17

3 AP 6.3

3.1 Integration COSMAS

Verantwortlich	Andreas Witt
Arbeitspaket	AP6.3
Entwicklung	Jira-Komponente • Subversion • Entwicklerdoku
Grundfunktionalität	siehe unten
TODOs bis ende TextGrid	Suche in lizensierten Korpora
Produktionsreife	100% im Monat (voraussichtlich) 04.2012, u.a. abhängig von TG-License
Einschätzung Entwickler	75% Produktionsreife erreicht
Teststatus	• Ausführliche Tests von ? liegen vor • Regelmäßiger Einsatz durch ?
Abhängig von ...	• TextGridLab • TG-License
abhängige Tools /Services	—
Sonstiges	• bislang nur Goethe Korpus, andere lizensierte Koprora erst nach Integration von TG-License möglich • bislang kaum/keine Lingusisten in TextGrid, daher konnten wir bis jetzt keine nicht-IDS-internen Fachwissenschafler zum Testen ermuntern können
letztes Update dieser Tabelle	2010-09-22

3.2 Integration LEXUS

Verantwortlich	Andreas Witt
Arbeitspaket	AP6.3
Entwicklung	LEXUS Homepage •
Grundfunktionalität	siehe White-Paper
TODOs bis zur Version 1.0	• Suche in freien Korpora • Suche in lizenzierten Korpora • Integration in das TextGridLab
Produktionsreife	100% im Monat (voraussichtlich) 04.2012
Einschätzung Entwickler	75% Produktionsreife erreicht
Teststatus	• IDS hat Tests vorgenommen; ähnlich wie bei COSMAS nicht-IDS-internen Fachwissenschaftler konnten bislang gewonnen werden
Abhängig von ...	• TextGridLab
abhängige Tools /Services	—

Sonstiges	• in aktueller Beta enthalten:
	  • Suche in zwei verschiedenen Lexika (Iwaidja) und Auszug aus eLexiko (Lemma mit Buchstaben A)
	  • Verschiedene Suchoperatoren (is, contains, begins with, end with)
	  • Auswahlmöglichkeit das Datenkategie in Abhändigkeit des Lexikons

letztes Update dieser Tabelle	2011-09-22

4 AP 6.4

4.1 Integration Digilib

Verantwortlich	Robert Casties
Arbeitspaket	AP6.4
Entwicklung	
Grundfunktionalität	siehe unten
TODOs bis zur Version ?? (Features)	• Scaler service im Backend • Digilib Bildbetrachtungs-Frontend
Produktionsreife	
Einschätzung Entwickler (erste Schätzung)	30.11.2010 (Scaler Service) 30.12.2010 (Digilib Front-end)
Teststatus	
Abhängig von ...	• TextGridLab
abhängige Tools /Services	
Sonstiges	—
letztes Update dieser Tabelle	2010-10-22

5 AP 6.5

5.1 Glossen-Editor

Verantwortlich	Clemens Radl
Arbeitspaket	AP6.5
Entwicklung	
Grundfunktionalität	siehe Glossen-Editor/Pflichtenheft
TODOs	• Anpassung des Text-Bild-Linkeditors: In der jetzigen Form erfüllt der TBLE die Anforderungen des Glossenprojekts, ist also erledigt. • Bereitstellung eines Präsentationstools: in Form eines XSLT, das für den Web-Previewer geeignet wäre, geschehen, letztlich muss aber ein eigenständiges Tool entwickelt werden, daher bleibt dieser TODO noch offen
Produktionsreife	Gesamtpaket: 04-2012 (Publisher in Vollversion: 11-2011)
Einschätzung Entwickler	40% (80% Publisher, 0% Editor, wobei das Verhältnis Publisher:Editor als 1:1 angenommen wird und Vorarbeiten, die auch den Editor betreffen (Datenformat, Adaptor, ...) nicht in diesen eingerechnet wurden)
Teststatus	—
Abhängig von ...	
abhängige Tools /Services	
Sonstiges	—
letztes Update dieser Tabelle	2011-09-20

6 AP 6.6

6.1 Noteneditor

Verantwortlich	Julian Dabbert
Arbeitspaket	AP6
Entwicklung	Subversion •
Grundfunktionalität	siehe Noteneditor/Pflichtenheft
TODOs bis zur Version 1.0	• intuitives GUI Konzept erarbeiten • Editor entwickeln
Produktionsreife	100% im Monat 2012-02
Einschätzung Entwickler	55% Produktionsreife erreicht
Teststatus	—
Abhängig von ...	—
abhängige Tools /Services	—
Sonstiges	—
letztes Update dieser Tabelle	2011-04-12

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

6.1.3 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.

6.1.4 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.

6.1.5 Abgrenzungen

Ein Export nach MusicXML wird über ein XSLT realisiert, welches von Perry Roland (Virginia 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.

6.2 Produktfunktionen

6.2.1 Unterstützung von MEI

Die Verwendung von MEI als natives Datenformat ist notwendig.

6.2.2 Einbindung in TextGrid

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

6.2.3 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.

6.2.4 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.

6.2.5 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.

6.3 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.

6.4 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.

6.5 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.

6.6 Technische Produktumgebung

Die Implementierung erfolgt in den gleichen technischen Rahmenbedingungen wie die übrige Client-Software des TextGrid Projektes, also Java 6 auf der Rich Client Platform auf Basis von Eclipse.

7 AP 6.8

7.1 OCR

Verantwortlich	Mayce Al Azawi
Arbeitspaket	AP6.8
Entwicklung	Jira-Komponente
Grundfunktionalität	siehe Pflichtenheft
TODOs (Features)	Verbesserung der Fraktur Erkennung und Sprach Modellierung für die Fraktur Dokumente
Produktionsreife	100% im Monat 2011-12. Das System wird in TextGrid getestet und Bugs (z.B. Carsh, syntaktisch falscher XML Output, etc.)
Einschätzung Entwickler	89% Produktionsreife erreicht. (Wir sind im Zeitplan)
Einschätzung Fachwissenschaftler	Es ist noch nicht veröffentlicht zu Fachwissenschaftler. Nach der Produktionsreife wird von Fachwissenschaftler eingeschätzt.
Teststatus	• Ausführliche Tests durch Martin Hellmann, Mohamadou Nassourou liegen vor • Regelmäßiger Einsatz bei Mayce Al Azawi
Abhängig von ...	• TextGridLab
abhängige Tools /Services	• Workflow-Editor
Sonstiges	Die Fraktur-Erkennung ist Verfügbar. Der OCR-Service läuft von den Workflow. Wir haben das Fontane Modell verfügbar zustellen. Und wir arbeiten noch auf die Sprachmodellierung und das Ersch-Gruber Modell.
letztes Update dieser Tabelle	2011.08.29