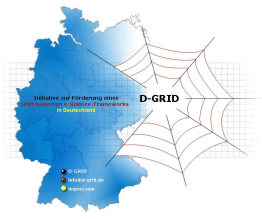
TextGrid Architektur

Version – Januar 2007
Arbeitspaket - AP 3, Report 3.2
verantwortlicher Partner - SUB, DAASI

TextGrid

wissenschaftliche Textdatenverarbeitung -
ein Community-Grid für die Geisteswissenschaften



Bundesministerium
für Bildung
und Forschung

Projekt: **TextGrid**

Teil des D-Grid Verbundes und der deutschen e-Science Initiative

BMBF Förderkennzeichen: 07TG01A-H

Laufzeit: Februar 2006 - Januar 2009

Dokumentstatus: stabil und eingefroren, wird kontinuierlich weiterentwickelt

Verfügbarkeit: öffentlich

Autoren:

Andreas Aschenbrenner, SUB
Peter Gietz, DAASI
Martin Haase, DAASI
Frank Knoll, DAASI
Christoph Ludwig, FH Worms
Wolfgang Pempe, Saphor
Markus Sosto, Saphor
Thorsten Vitt, TU Darmstadt

Inhaltsverzeichnis

Überblick

TextGrid ist ein Infrastrukturprojekt, das auf Grid-Technologien aufbaut. Es bedient eine Community aus den Geisteswissenschaften und versteckt dabei die Komplexität der Technologie weitestgehend. Angelehnt an das Paradigma der Service Oriented Architecture, kombiniert TextGrid dazu Grid-Technologien und Web Services in einer offenen Schichtenarchitektur.

Dieses Dokument gibt einen globalen Überblick über die TextGrid Architektur. Es baut dabei auf die Analyse der Bedürfnisse der Community in den "TextGrid Szenarien" [1] genauso wie auf die aktuellen Entwicklungen in der Grid / (Semantic) Web Community. Diese **Referenzarchitektur** definiert den technischen Rahmen und die Terminologie, nach der die weiteren Projektphasen arbeiten, auf einem entsprechenden Abstraktionsgrad. Eine Feinspezifikation für die in den Szenarien und in diesem Dokument identifizierten Einzelkomponenten existiert bereits ansatzweise an separater Stelle und wird während der TextGrid Projektlaufzeit sukzessive fortgeführt (s.a. die Reports zu AP2).

Aus den in den Szenarien entwickelten Anwendungen und Kontexten ergeben sich aus technischer Sicht folgende **Ziele** für die Referenzarchitektur:

- TextGrid ist eine **offene, generische Infrastruktur** - Dem Benutzer werden nur dort Vorgaben gemacht, wo dies unumgänglich ist. Datenformate, Metadaten und dergleichen können weitgehend den jeweiligen Anforderungen entsprechend angepasst werden. Anwendungen sind konfigurierbar, und es können sogar eigene Anwendungen in die TextGrid Umgebung integriert werden.
- ermöglicht **spezialisierte Anwendungen** und tiefgehende Erschließung - Wie der erste Punkt schon sagt, muss TextGrid die spezifischen Anforderungen der Textwissenschaften und der Wissenschaftler unterstützen, um akzeptiert zu werden. Die Anwendungen der Textwissenschaftler sind oft hinreichend komplex, und deren übergreifende Umsetzung in einem offenen Umfeld ist entsprechend eine Herausforderung (die diese Referenzarchitektur über die stufenweise Integration von Inhalten und Anwendungen bewältigt).
- und motiviert **Partizipation** - Die Community ist wichtig für TextGrid, zum Aufbau der Datenbasis, für die Breite der Anwendungen, und zum Erhalt der Infrastruktur. Die Architektur erlaubt die aktive Teilnahme an TextGrid auf jeder Ebene. Die Verwendung von verbreiteten Standards sind nur ein Aspekt davon. Je größer die Community desto größer der Nutzen von TextGrid für die Community.

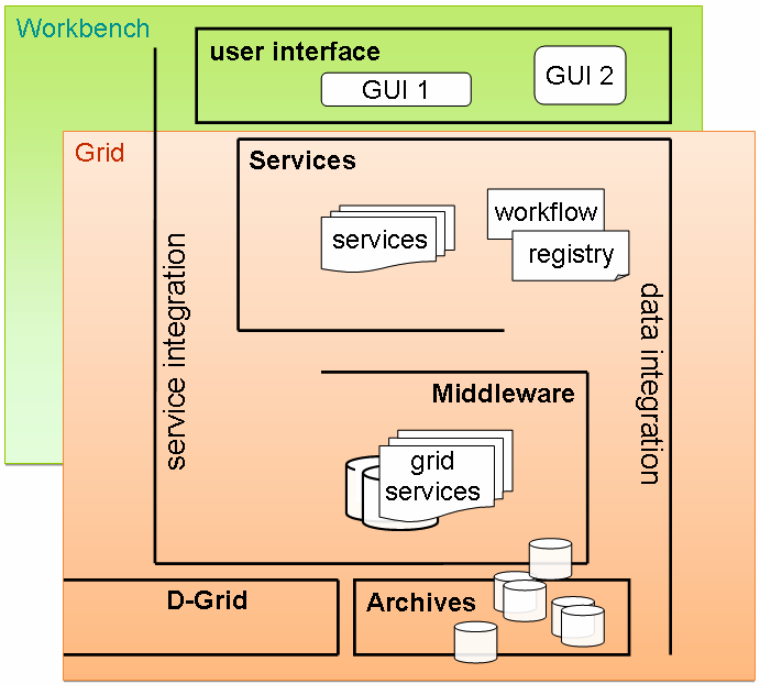
Alle diese Ziele sind in jeder Komponente der Gesamtarchitektur manifest. Die Komponenten sind in vier Schichten eingeteilt

1. **Benutzerumgebungen** - Dies ist das Tor zu TextGrid. Unterschiedlichen Benutzem mit ihren speziellen Anforderungen werden spezifische Benutzerumgebungen angeboten(z.B. Offline-Funktionalität, zugeschnitten auf Arbeitsprozesse in der

[1] TextGrid Szenarien sind in einem separaten Dokument entwickelt worden, das über die TextGrid Webseite verfügbar gemacht wird. Die Szenarien gehen auf die Arbeitsweisen verschiedener Zielgruppen ein, und entwickeln daraus eine mögliche Einbettung in TextGrid.

literaturwissenschaftlichen Edition). Neue Benutzerumgebungen können mit geringem Aufwand auf die anderen Schichten der TextGrid Architektur aufgesetzt werden.

2. **Service Layer** - Hier werden komplexe fachspezifische Funktionalitäten mittels Web Services (basierend auf gängigen Standards wie SOAP, WSDL, etc.) gekapselt, und können in beliebiger Zusammensetzung in beliebigen Benutzerumgebungen eingebunden werden. Die Service Ebene ist auch durch externe Initiativen erweiterbar.
3. **Middleware** - Ohne die TextGrid Basisdienste in der Middleware geht nichts. Die Middleware verknüpft Grid Technologien mit anderen (u.a. semantischen) Technologien, um die grundlegenden Funktionalitäten eines verteilten Datengrids nach den Anforderungen der Textwissenschaften umzusetzen.
4. **Archive** - Externe, heterogene Textarchive werden in TextGrid integriert und in der Middleware virtualisiert. Textarchive lassen sich stufenweise integrieren, wobei jede Stufe höhere technische Anforderungen aber auch durch die stärkere Integration mehr Möglichkeiten bietet.



TextGrid Gesamtarchitektur

Während die Grundarchitektur gängigen Schichtenmodellen in dem Bereich folgt, gibt es eine Reihe von **Tools**, die schichtenübergreifend spezielle Funktionalitäten umsetzen.

- *interaktive Tools* haben ein Benutzerinterface und sind auf Benutzereingaben ausgerichtet [Benutzerschnittstelle]
- *Streaming Tools* bestehen aus zwei Teilen: einem im Batchbetrieb steuerbaren "Enactor" und einem GUI-Frontend, mit dem Aufträge (bzw. Konfigurationen) für den Enactor entwickelt werden können. Die GUI-Komponenten sind eng mit den interaktiven Tools integriert. Die Enactors können jeweils sowohl im Rahmen eines Workflows durch den Workflow-Enactor als auch direkt aus ihren jeweiligen GUI-Frontends angestoßen werden. [Service Layer, Benutzerschnittstelle]

- *Basistools* - wie das Recherche-Tool haben Komponenten in allen Schichten [Middleware, Service Layer, Benutzerschnittstelle]
- *Hilfstools,* werden nur als Bestandteil eines anderen Werkzeugs aufgerufen ohne eine eigene Benutzerschnittstelle anzubieten [Service Layer oder Middleware]

Im Folgenden werden alle Schichten kurz charakterisiert, und die Komponenten in der Middleware sowie die stufenweise Integration der Archive beschrieben.

Benutzerumgebung

Eine Benutzerumgebung ist quasi das Tor zu TextGrid. Über sie kann ein Benutzer auf die für ihn relevanten Teile von TextGrid zugreifen. Mehrere unterschiedliche Benutzerumgebungen können auf den anderen Schichten von TextGrid aufsetzen. Dabei ist jede Benutzerumgebung auf einen bestimmten Nutzerkreis ausgerichtet, um sie möglichst intuitiv und effektiv bedienbar zu machen. Dazu werden Funktionalitäten gezielt ausgewählt und auf die Aufgaben und Prozesse des Nutzers angepasst.

Eine Benutzerumgebung bettet interaktive Tools und Streaming Tools ein. Interaktive Tools sind direkt in der spezifischen Technologie der Benutzerumgebung implementiert, während die Kernfunktionalität der Streaming Tools in Services (eine Ebene tiefer in der Referenzarchitektur) umgesetzt wird. Da der Service Layer durch standardisierte Schnittstellen weitgehend technologieunabhängig arbeitet, kann ein Service in verschiedene Benutzerumgebungen eingebunden werden. Es muss lediglich der Teil zur Bedienung des Services in der Technologie der Benutzerumgebung umgesetzt werden. Wegen der Wiederverwendbarkeit von Services und der damit einhergehenden Reduktion des Implementierungsaufwands wird möglichst viel Funktionalität in Services umgesetzt. Der Service Layer wächst unabhängig von den Benutzerumgebungen, und idealerweise sind Benutzerumgebungen um durch neue Services hinzukommende Funktionalitäten erweiterbar.

Während Services und damit auch Streaming Tools eine Anbindung an das Grid brauchen, können interaktive Tools in einer Benutzerumgebung auch weitgehend in einem offline Modus bedient werden.

Den XML Editor, zum Beispiel, ein interaktives Tool, kann der Benutzer für manuelle Editions- und Annotationsarbeiten auch ohne permanente Netzanbindung betreiben. Die für die Arbeiten nötigen Objekte (TEI-gesetzte Transkriptionen, etc.) lädt der Benutzer dazu im Vorfeld in seinen Client, und synchronisiert sie im Anschluss wieder mit den Daten im Grid.[2]

Das Recherche-Tool, auf der anderen Seite, holt sich bei jeder Benutzeranfrage die Ergebnisse aus der Grid Umgebung und kann deshalb nur online benutzt werden. Ebenso können Streaming Tools wie der Lemmatisierer oder der Kollationierer durch die Benutzerumgebung im Online-Modus eng mit dem XML Editor verknüpft werden, stehen jedoch ohne Netzanbindung nicht zur Verfügung.

Während einfache Bedienbarkeit ein Ziel für jedes Tool ist, sind die Aufgabenstellungen von einzelnen Nutzern oft hinreichend komplex und spezialisiert. Die Gratwanderung zwischen einfacher Bedienbarkeit und Flexibilität spiegelt sich in jedem Tool wider. Generell sind Textwissenschaftler durchaus komplexe Tools gewohnt (vgl. z.B. TUSTEP).

Während der Projektzeit von TextGrid wird hauptsächlich eine auf Edition, Annotation und Analyse ausgerichtete, Eclipse-basierte Umgebung[3] für Editionswissenschaftler und Linguisten entwickelt. In dieser Umgebung sind mehrere interaktive Tools (XML Editor, grafischer Link Editor, etc.) und Streaming Tools verknüpft (Kollationierer, etc.).

---

[2]Empfohlene Auszeichnungen, Metadaten, Versionierung, Konsistenzsicherung und andere Aspekte werden derzeit in einem TextGrid-Objektmodell entwickelt.

[3]Durch die *Rich Client Plattform* von Eclipse (http://wiki.eclipse.org/index.php/Rich_Client_Platform) kann ein Stand-alone-Client mit Offline-Funktionalitäten geschaffen werden. Eclipse ist leicht zu installieren, plattformunabhängig, und gut erweiterbar. Aus der großen Eclipse Community sind bereits eine Vielzahl von Plugins verfügbar.

Die genaue Beschreibung der Aufgabenbereiche und Arbeitsprozesse, die ein Tool abdeckt, sind in den TextGrid Szenarien beschrieben. Die Spezifikation der einzelnen Tools ist im TextGrid Funktionskatalog[4] umgesetzt.

[4] Ein Funktionskatalog aller im Rahmen der Projektlaufzeit umzusetzenden Komponenten (Tools) wird parallel zu deren Entwicklung geführt. Dieser Katalog wird öffentlich verfügbar gemacht, sobald die Spezifikation und Entwicklung fortgeschritten ist.

Service Layer

Ein "Service" ist eine Software-Einheit, die in einem Netzwerk eindeutig identifizierbar ist, über standardisierte Protokolle angesprochen, und dadurch in ein anderes System und die dortige Benutzerschnittstelle eingebunden werden kann. Ein Service kann dementsprechend an jedem Ort in dem Netzwerk gehostet sein, und auf einer beliebigen Plattform laufen bzw. in einer beliebigen Programmiersprache implementiert sein - solange der Service identifizierbar und die Schnittstelle zur Kommunikation bekannt ist, kann der Service von überall im Netzwerk angesprochen werden.

In einer "Service Registry" können Services ihre Dienste registrieren und beschreiben, sodass sie von anderen Programmen entdeckt werden können.[5] Dadurch können jederzeit neue Services in die TextGrid Umgebung aufgenommen werden. Z.B. könnte ein Lemmatisierer für Griechisch neben dem TextGrid Lemmatisierer für Deutsch hinzugefügt werden, oder es könnten sogar mehrere Lemmatisierer für Deutsch registriert werden, die jeweils verschiedene Algorithmen benutzen und somit für unterschiedliche Anwendungsfälle einsetzbar sind.

Services können beliebig untereinander kombiniert und wieder verwendet werden. Ein "Workflow Modul" kann automatisch oder nach den Vorgaben eines Benutzers den Datenfluss zwischen einer Reihe von Services dirigieren, und somit eine komplexe, auf spezifische Anforderungen zugeschnittene Funktionalität aus wiederverwendbaren Bausteinen zusammenstellen.

Üblicherweise versteht man unter "Web Services" die vom World Wide Web Consortium (W3C)[6] definierten Standards, die eine Serviceumgebung definieren. Unter den Standards sind SOAP und WSDL, XML-basierte Protokolle. Der TextGrid Service Layer ist über diese Protokolle definiert, und somit theoretisch mit allen SOAP-basierten Services interoperabel.

Services in TextGrid sind z.B. der Sortierer, der Daten anhand gängiger Standards und/oder manueller Konfiguration sortiert. Um ein möglichst generisches System zu erreichen, gibt jeder Service für die Input Daten möglichst wenige Vorgaben. So erwartet der Sortierer nicht notwendigerweise eine TEI-Datei in einem speziellen TEI Schema, sondern er arbeitet auch mit einer Liste von Wörtern. Der Kollationierer hingegen, ein weiterer TextGrid-Service,[7] erwartet zwei TEI Dateien als Input.

Generell kann man zwischen unterschiedlichen Arten von Services unterscheiden:

- rein Algorithmus-basierte Services, die ihre gesamte Funktionalität in ihrem Programmcode umsetzen. z.B. der Tokenizer
- Services, die auf eine Wissensbasis aufbauen. Die Wissensbasis ist unveränderlich, bzw. wird nur vom Erzeuger des Services gelegentlich aktualisiert, z.B. der Lemmatisierer
- Services mit einem Gedächtnis, das durch Aktivitäten der Benutzer entsprechend angepasst wird. z.B. das Bibliografie-Tool

---

[5] All dies sind wichtige Bausteine einer "Service Oriented Architecture", einem besonders robusten und flexiblen Software-Architektur-Paradigma, nach dem auch aktuelle Grid Middleware Systeme modelliert sind.

[6] World Wide Web Consortium (W3C), http://www.w3.org/

[7] Siehe die TextGrid Szenarien, bzw. den Funktionskatalog für eine detaillierte Beschreibung der Services.

Services können relativ schnell erzeugt und in die TextGrid-Umgebung eingebettet werden. Bei manchen Services könnte es Sinn machen, sie langfristig tiefer in die Grid-Umgebung zu integrieren und zu einem Teil der Basisdienste in der Middleware zu machen. Solche Dienste könnten z.B. die Metadatenverwaltung oder die TextGrid-weite Suche sein. Als Teil der Grid-Middleware und technologisch konform mit Grid Service Paradigmen, profitieren diese Dienste vor allem von der möglichen Verteilbarkeit (Skalierbarkeit, Robustheit) und dem direkten Zugriff auf die Hardware-Ressourcen. In dieser zweistufigen Integration kann der Schritt zur Aufnahme als ein TextGrid Basisdienst organisatorisch gesteuert und technisch unterstützt werden, um den Aufwand so gering wie möglich zu halten.

Um Homogenität innerhalb der Services in der TextGrid Umgebung zu erzeugen, wird eine Art Servicehierarchie als Empfehlung für Service-Entwickler etabliert.[8] Jede Schicht gibt hierbei eine Reihe von Konventionen vor. Je nachdem wo in dieser Hierarchie ein Service klassifiziert wird, setzt er die spezifischen Konventionen um. Dadurch wird z.B. garantiert, dass verschiedene Streaming Tools ähnliche Schnittstellen haben und analog angesprochen werden können. Allgemeine Konventionen für alle Services beinhalten z.B. die für die Authentifizierung und Sicherheit notwendigen Mechanismen. Eine dokumentierte Klassenhierarchie könnte den Service Layer strukturieren und zur Wiederverwendbarkeit von Modulen beitragen, die Standardisierung von Interfaces/APIs fördern, Programmierarbeiten und das Andocken von neuen Services (evtl. von externen Initiativen) erleichtern.

Die einzelnen vom Benutzer (entweder direkt oder über den Workflow-Editor) aufgerufenen Services erhalten die Zugriffsrechte des Benutzers, die insbesondere bei Datei-Zugriffen in Kraft treten. Solche authentifizierungs- und autorisierungsrelevante Informationen werden den Services mitgegeben. Über SOAP-eigene Methoden können diese auch von einem Service auf den nächsten übertragen werden. Die Services selbst können sich zusätzlich über X.509-Zertifikate authentifizieren.

Zusammenfassend lässt sich sagen, dass die atomaren Funktionalitäten von Services beliebig ausgewählt und zu komplexen Anwendungen kombiniert, und in eine oder mehrere Benutzerumgebungen integriert werden können. Jeder neue Service erweitert den Funktionskatalog und somit auch die Nutzerkreise, die durch TextGrid angesprochen werden können. Die Offenheit des Service Layers soll auch die Partizipation von externen Initiativen motivieren, obwohl gewisse Konventionen für eine effektive Arbeitsumgebung notwendig sind. Externe Service Provider könnten langfristig sogar auf einer kommerziellen Basis ihre Services anbieten, oder Unterstützung bei der Integration von neuen Funktionalitäten in die TextGrid Umgebung anbieten.

[8] Ein ähnlicher Ansatz wird vom e-Framework, eine Standardisierungsinitiative von JISC in Großbritannien und DEST in Australien, verfolgt. Dort wird unterschieden zwischen *Service Genres* und *Service Expressions.* siehe http://www.e-framework.org

Middleware

Die Middleware verwaltet verteilte Ressourcen in TextGrid und virtualisiert diese für homogenen Zugriff. Als ein Kernaufgabengebiet in TextGrid sorgt die Middleware für die Virtualisierung der vorhandenen Speicherressourcen (Storage) in geographisch verteilten Knoten und erzeugt ein homogenes virtuelles Archiv. Hierbei versucht die Middleware die Brücke zwischen einer möglichst generischen Ausrichtung (Andocken an die D-Grid Integrationsplattform und Vernetzung mit anderen Grid Projekten) und spezifischen Anforderungen (eine möglichst komfortable Schnittstelle für die TextGrid-spezifischen Dienste) zu schlagen.

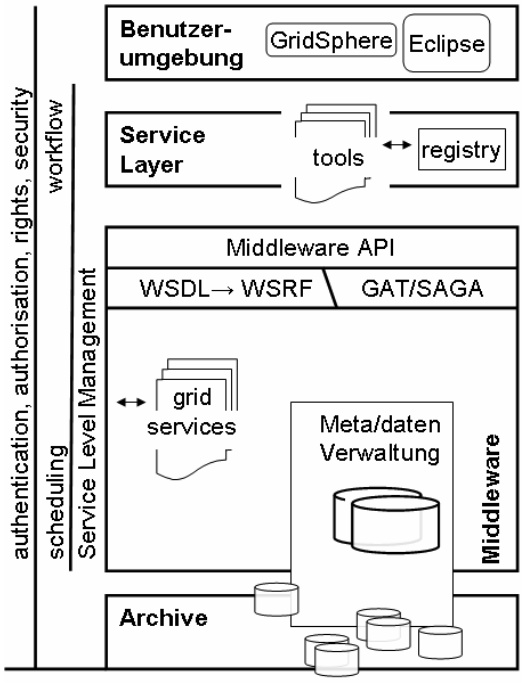
Es gibt verschiedene Ausprägungen von Grid-Konzepten und -Technologien in TextGrid. Ein zentraler Fokus liegt in der Schaffung eines Datengrids zur Verknüpfung externer Archive, Datenreplikation und Metadatenverwaltung. Die Schaffung eines "Service Grid" - eine offene Plattform von Funktionalitäten, die beliebig untereinander kombiniert werden können - wurde im letzten Kapitel vorgestellt. Zentrale Funktionalitäten können als Basisdienste schrittweise in die TextGrid-Middleware integriert werden, wo für ihre Verfügbarkeit und Skalierbarkeit besonders gesorgt werden kann (z.B. mittels verteilter, evtl. redundanter Datenbanken, etc.). Basisdienste bieten generische Funktionalitäten, die möglichst nicht auf spezielle inhaltliche Konzepte (z.B. Objektmodelle, bestimmte Schemata oder Metadaten) festgelegt sind.

Nachhaltiger Datenspeicher ("Data Grid") und der Aufbau einer offenen Diensteplattform ("Service Grid") ist Teil der Kernmission von TextGrid und in der ersten Phase gegenüber Ansätzen zur Verteilung der Rechenlast ("Computational Grid") priorisiert. Aber es gibt auch eine Reihe von Szenarien, in denen umfangreiche Rechenleistungen nötig sind. Darunter sind die automatische Segmentierung von Digitalisaten, die Anwendung von Information Retrieval und Mining Technologien, etc. Obwohl diese Szenarien nicht Teil der ersten Projektphase sind, wird deren Umsetzung durch die Verwendung einer generischen Middleware vorbereitet, in der auch ein Computing Grid realisiert werden kann.

Die TextGrid Middleware wird mit dem Globus Toolkit aufgebaut – und einigen zusätzlichen Komponenten zum Datenmanagement. Wie bei der darüberliegenden Schicht "Service Layer" beschrieben, gibt es zwischen den beiden Schichten noch eine Abstraktionsschicht. In dieser Abstraktionsschicht wird übersetzt zwischen der Web Service-basierten Umgebung des Service Layers (i.e. WSDL, SOAP; ist HTTP basiert und statuslos), und der WSRF-basierten Gridinfrastruktur (statusbehaftete, sessionbasierte Services). Zusätzlich wurde als Teil der Middleware SAGA bzw. GAT zwischengezogen, das den Zugriff auf Globus Grid Services kapselt und vereinfacht, sowie einen zukünftigen Wechsel der Grid-Infrastruktur ermöglicht..

Einige Grid-Komponenten interagieren mit mehreren Schichten in der TextGrid Gesamtarchitektur oder überbrücken diese. In der Skizze erkennt man z.B. die AAR (Authentication, Authorisation, Rights Management) Komponente, die in allen Schichten manifest ist. Für das Workflow Management ist sowohl ein Workflow Enactor im Service Layer als auch die Scheduling Komponente in der Middleware notwendig; die Benutzerschnittstelle Workflow Editor ist zwar bedacht aber in der Architekturskizze nicht verzeichnet (der Balken unter "workflow" überdeckt lediglich die beiden Schichten "Middleware" und "Service Layer"), da sie rein die Benutzerschnittstelle darstellt. Alle diese Komponenten werden im Folgenden einzeln beschrieben.

Abstraktionsschichten



TextGrid API

Die TextGrid API ist eine aus reinen Web-Services bestehende Softwareschicht, die dem Service-Entwickler der „Service Layer“ eine TextGrid-spezifische Schnittstelle zu den folgenden Gridfunktionalitäten anbietet:

- Datenmanagement: Die Bearbeitung der in einem verteilten Dokumenten-Management-System vorliegenden Texte und Binärdaten werden durchdie folgenden Web-Services ermöglicht:
  - InputStreamService bzw. InputStreamBinaryService zum sequentiellen Lesen von Text bzw. Binärdaten,
  - OutputStreamService bzw. OutputStreamBinaryService zum sequentiellen Schreiben von Text bzw. Binärdaten und
  - RandomAccessService zum Lesen und Schreiben von Binärdaten an beliebigen Positionen.
- Metadatenmanagement: Für das Veröffentlichen und Wiederfinden von TextGrid-Objekten mitsamt beschreibenden Metadaten gibt es den AdvertService Web-Service. TextGrid-Objekte können dabei unter einem absoluten oder relativen Pfad veröffentlicht werden. Beim Veröffentlichen eines Objektes können das Objekt beschreibende Metadaten mitgegeben werden (Register), über die das Objekt auch wiedergefunden werden kann (Index). Die API stellt hierfür Dienste zur Registrierung und zum Retrieval von Informationsobjekten zur Verfügung . In einer späteren Version der Architektur werden neben den Datenressourcen auch die zur Verfügung stehenden Dienste über Metadaten recherchierbar sein, sowie die Rechenressourcen, auf denen die Dienste betrieben werden können.
- Konfiguration: Die Web-Services der Serviceschicht haben eine gemeinsame Konfigurationsinfrastruktur, die über den File-Service der Middleware zur Verfügung

gestellt wird. Hierdurch wird es möglich, ausgetestete und bewärte Konfigurationen der einzelnen Web-Services in Dateien, die von der Middleware wie andere Informationsobjekte verwalten und von den Web Services zur Konfiguration geladen werden können. Ein einheitliches Konfigurationsformat sorgt für eine weitere Homogenisierung der TextGrid-Web-Services. Selbstverständlich können die Web Services auch direkt über die durch SOAP gegebenen Möglichkeiten der Parameterübergabe konfiguriert werden.

- Logging: Die TextGrid-Middleware stellt ein zentrales Logging zur Verfügung, welches von den Web-Services über eine API verwendet werden kann. Dies ermöglicht auch ein einfaches zentrales Monitoring der Web-Services.
- Security: Grundlage der Security-Infrastruktur der Grid-Middleware sind X.509 basierte Zertifikate. Diese komplexe Technologie soll vor dem Benutzer gekapselt werden, um den Aufbau einer aufwendingen PKI für User-Zertifikatezu vermeiden. Die Textgrid-API muss also zwischen einer einfachen Authentifizierung über UserID/Passwort und einer Zertifikatsbasierten Authentifizierung, wie sie von der Grid-Infrastruktur wie Globus Toolkit (Grid Security Infrastructure, GSI) erwartet wird, ebenfalls moderieren. Darüberhinaus wird die Middleware eine Benutzerverwaltung realisieren (s.u. 4.3) und über die API deren Administration ermöglichen, also Accounts anzulegen, bzw. zu löschen, sowie Benutzern Attribute und Rollen zuweisen, die für Zugriffsrechte relevant sind.

WSDL-WSRF

Für den Aufbau der TextGrid-Middleware wird die Middleware Globus-Toolkit-4 (GT4) eingesetzt. GT4 bietet in seiner API Web-Services an, die auf der Basis von WSRF (siehe Beschreibung in R3.1) erstellt wurden und somit „Web Services Addressing“ (WSA, siehe ebenfalls R3.1) für ihre Adressierung verwenden. Da WSA jedoch den Rahmen von einfachen, reinen Web-Services sprengt, können mit GT4 erstellte Web-Services nicht unmittelbar in der TextGrid-API angeboten werden, die ja möglichst einfache Web-Services anbieten will. Somit benötigt die Implementierung der TextGrid-API eine Zwischenschicht, die zwischen reinen Web-Services und WSRF-Web-Services vermittelt. Technisch bedeutet dies eine Übersetzung zwischen URLs und WSA-EndpointReferences, die in dieser Zwischenschicht stattfindet.

GAT/SAGA

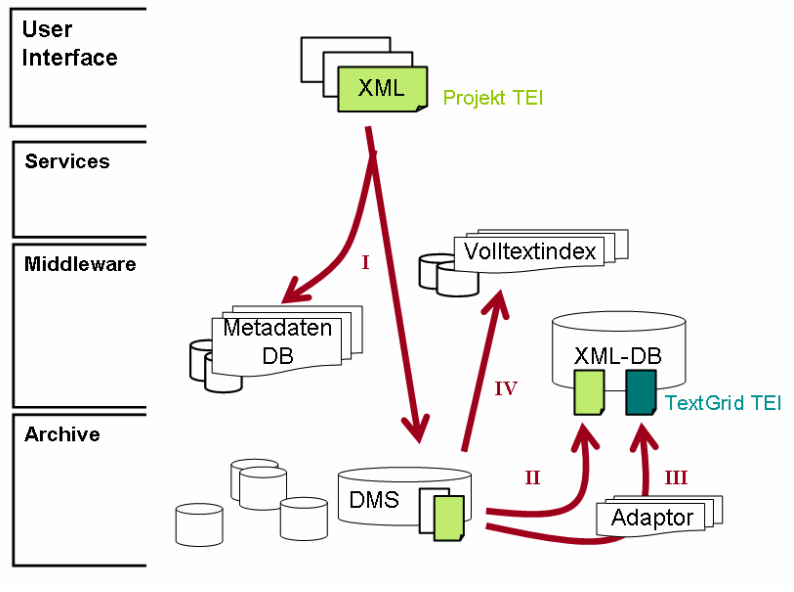
Da nicht alle Gridfunktionalitäten (z.B. GridFTP) der Globus-Toolkit-4-Middleware über WSRF-Web-Services ansprechbar sind, sondern über teilweise sehr komplexe Non-Web-Services-APIs, werden in der Implementierung der TextGrid-API die Toolkits GAT bzw. später dessen Nachfolger SAGA eingesetzt, die eine vereinfachte und flexible Schnittstelle zu Gridfunktionalitäten verschiedener Middleware-Toolkits – darunter auch GT4 – anbieten.

TextGrid-spezifische Basisdienste zur Datenverwaltung

Die Datenverwaltung setzt sich aus einer Reihe von Basisdiensten zusammen, die gemeinsam einen homogenen Zugang zu verteilten Datenbeständen ermöglichen und die speziellen Anforderungen der Textwissenschaften weitgehend kapseln. Die Funktionalitäten der Datenhaltung und die der textwissenschaftlichen Datenverwaltung sind in der Grafik als zwei verschiedene Schichten angedeutet, obwohl sie eng verknüpft sind. Dies soll verdeutlichen, dass die Datenhaltung über externe, heterogene Archive mittels eines Schichtenmodells realisiert ist, wie im folgenden Kapitel beschrieben wird. Das Schichtenmodell zur Archivanbindung bezieht

die Archive und deren technische Möglichkeiten in einem abgestuften Modell mit ein. Eine der Schichten ist ein veritables Datengrid mit Funktionalitäten zur Dokumentenhaltung (DMS).

Neben der DMS Komponente gibt es eine Kerninfrastruktur mit Verwaltungsaufgaben zur Metadatenverwaltung, zur Indizierung der Volltexte, und zur tiefgehenden Erschließung wissenschaftlicher Texte und Strukturdatenverwaltung (XML-DB). Diese Komponenten und deren Verknüpfung sind im Folgenden beschrieben.



Datenhaltung und -verwaltung in TextGrid

Datenhaltung - DMS

In TextGrid werden diverse Informationsobjekte vorgehalten, darunter vor allem Bilder (jpeg, tiff, png, etc), Texte (XML, TEI, etc.), als auch beliebige andere, auch proprietäre Formate (TUSTEP, Winword, etc.). Klarer unterschieden werden kann zwischen statischen und dynamischen Objekten. **Statische** Objekte werden einmal in einem Textarchiv registriert und verändern sich von da an nicht mehr; dies sind vor allem Bilddaten. **Dynamische** Objekte sind Teil der aktuellen Arbeit von Textwissenschaftlern und werden regelmäßig fortgeschrieben. Das DMS verwaltet einheitlich alle Objektarten, obwohl die (externen) Textarchive ihre Datenbestände selbständig verwalten und ggf. nur statische Objekte registrieren. Der individuelle Zugang zu den Informationsobjekten wird über die Metadaten ermöglicht, die auf die technischen, administrativen und inhaltlichen Eigenschaften der Objekte individuell zugeschnitten sind.

In der Datenhaltung werden eine Reihe von Aufgaben erfüllt:

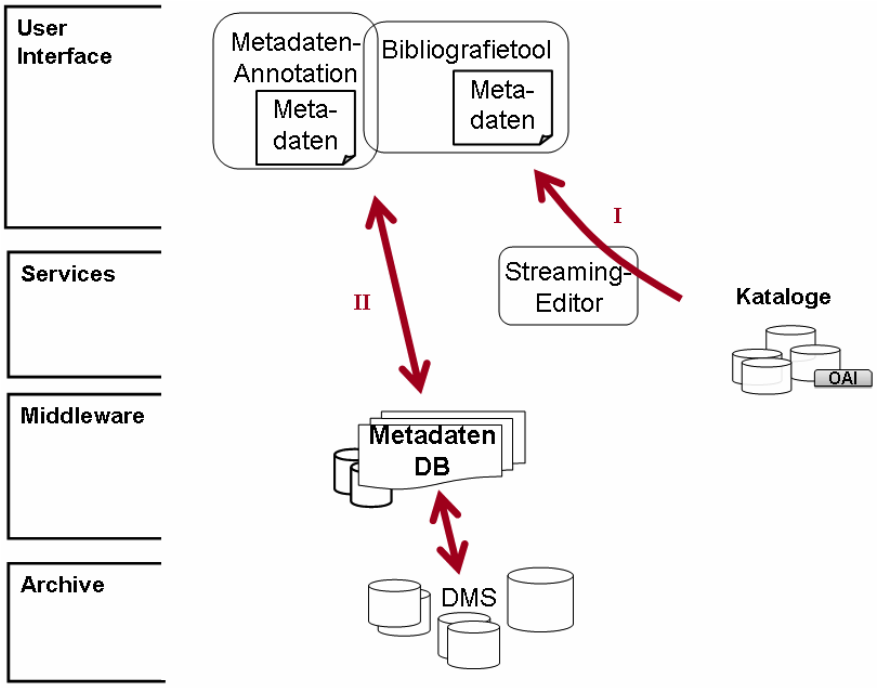
- In der **Datenstrukturierung** geht es darum, wo, in welcher Form und Struktur, welche Informationsobjekte vorgehalten werden.
    - Werden einzelne Dateien eines zusammengesetzten Informationsobjektes (z.B. ein Text mit Bildern) in Bundles zusammengehalten?
    - Werden die Informationsobjekte mit ihren Metadaten z.B. in METS Files gekapselt?

    - Wie sieht der Verzeichnisbaum aus?
    - Wie werden unterschiedliche Versionen eines Objektes untereinander verknüpft?, nur über ihre Metadaten oder spiegelt sich das auch in der Datenstrukturierung wider?
- Für den **Zugriff** sind verschiedene Mechanismen möglich bzw notwendig:
    - Eine eindeutige Identifikation der Objekte muss
        - auf ein einzelnes Objekt innerhalb eines Bundles verweisen können,
        - Zusammengehörigkeiten, unterschiedliche Versionen, etc. abbilden,
        - intern wie auch extern ansteuerbar sein.
    - In der Rechteverwaltung wird auf der einen Seite das interne Rechteregime umgesetzt (siehe die Rollenaufschlüsselung in den TextGrid Szenarien; z.B. welche Rolle innerhalb einer VO darf welche Objekte löschen, etc.), als auch die Rechte externer Textarchive eingebettet.
    - Wenn Informationsobjekte nach außen transportiert werden, werden sie mit ihren Metadaten gekapselt.
- Eine wichtige Eigenschaft des Datengrids ist die **verteilte, redundante Datenvorhaltung**. Zur Datenreplikation müssen einige Aspekte definiert werden:
    - Welche Objekte werden wo repliziert?, welche nicht?
    - Wieviel Speicherplatz steht bei den Textarchiven zur Replikation zur Verfügung? Überwachung des Speicherraumes - Auslastung einzelner Knoten (Speicherplatz vs. Speicherbedarf, Performance)
    - Mechanismen zur Erhaltung der Konsistenz zwischen den verteilten Replika und zum einheitlichen Zugriff.

Die hier besprochenen Aspekte des DMS werden derzeit gemeinsam mit Fachwissenschaftlern bei Textarchiven entwickelt und sind teilweise schon in Prototypen umgesetzt. Eine genaue Spezifikation aller Aspekte wird im TextGrid Meta/Daten Konzept und in der DMS Systemspezifikation vorgenommen. Grundlage sind hierbei die Datagrid-Funktionalitäten, die in GT4 zur Verfügung stehen (GridFTP, Reliable File Transfer RFT, Replica Management, etc.)

Metadaten

Eine Kernkomponente zur Verwaltung von und zum Zugriff auf Informationsobjekte ist die Metadatenverwaltung. In einem TextGrid-internen *Metadatenschema* sind diverse Aspekte abgedeckt, von deskriptiven Metadaten (z.B. Autor, Titel) zu Kontextmetadaten (dieses Objekt ist mit diesem anderen Objekt verwandt, etc.) zu administrativen Metadaten (Rechte, Objektherkunft, etc.). Metadatenschemata haben einen Teil, der verpflichtend auszufüllen ist (mandatory) sowie optionale Teile, und sind für jedes Objekt beliebig erweiterbar. All diese Metadaten werden in einer eigenen Datenbank verwaltet. Die Metadaten-Datenbank wird als ein Basisservice implementiert.



Metadaten Datenfluss

Metadaten können verschiedene Ursprünge haben:

1. ein Bearbeiter erstellt sie manuell im Tool zur Metadaten-Annotation oder im Bibliografietool
2. sie werden aus existierenden Metadatenkatalogen (z.B. Kalliope, zvdd) übernommen
3. importierte Informationsobjekte enthalten Metadaten (z.B. im TEI Header)

Neben dem Import aus unterschiedlichen Quellen können Metadaten auch in verschiedenen Formen exportiert werden (in ein Informationsobjekt eingebettet, als Dublin Core oder MARC Schema). Während der Kern der Metadatenverwaltung also die Metadaten-Datenbank ist, so sind die verschiedenen Möglichkeiten zum Import und Export, sowie der Abgleich zwischen der Metadaten-DB und dem DMS essenziell für eine effektive Metadatenverwaltung.

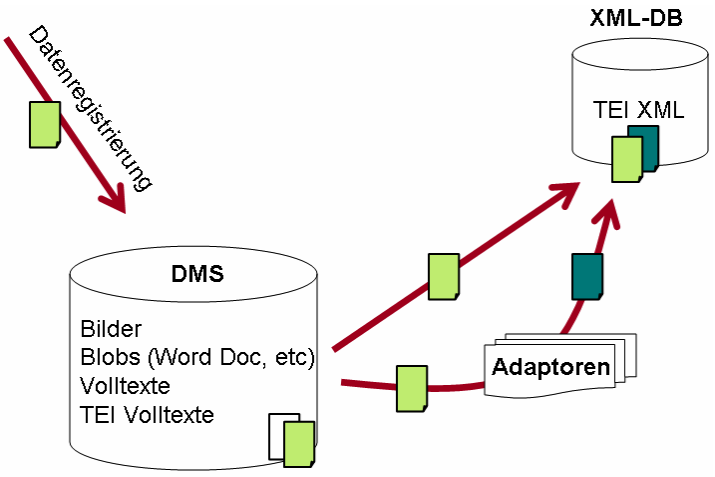
Textannotationen und Adaptoren

Neben den Metadaten und den Indexdaten sind für die textwissenschaftliche Datenverwaltung vor allem Annotationen eine Kernaufgabe. Annotationen sind ergänzende und vertiefende Informationen zu Texten, die bis zu einer feinen Granularitätsstufe (z.B. Wortebene) hinunterverzweigt sein können. Annotationen können viele verschiedene Ebenen einnehmen, darunter physische (z.B. hier war ein Kaffeefleck), textgenetische (z.B. der Autor hat in der ersten Überarbeitung dieses Wort gestrichen), linguistische (z.B. das Lemma dieses Wortes ist ...), semantische Beschreibungen und Stellenkommentare. Die Erschließung von Annotationen, die in XML Markup direkt im wissenschaftlichen Text verzeichnet sind, ermöglicht eine XML Datenbank. Informationsobjekte im DMS, die in XML vorliegen, werden zusätzlich in der XML Datenbank gespeichert und können mit den dort vorhandenen Mechanismen erschlossen werden (z.B. Abfrage, XQuery, etc.).

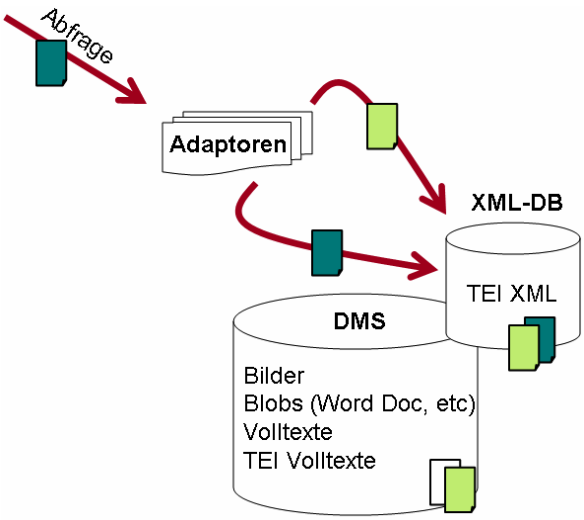
Da zumeist jedes Projekt ein individuelles XML Schema für seine spezifischen Anforderungen erzeugt, liegen die in TextGrid vorhandenen wissenschaftlichen Texte in heterogenen Formaten

vor. Adaptoren kapseln die heterogenen Formate und ermöglichen eine homogene Erschliessung der Objekte. Zwei Aspekte müssen hier bedacht werden:

1. Bei der Registrierung wird das Informationsobjekt sowohl in seinem Herkunftsformat als auch im TextGrid-spezifischen Format gespeichert. Operationen in der XML-DB operieren immer auf beiden Formaten. Hierdurch werden projektspezifische und TextGrid-weite Anfragen immer parallel ausgeführt.



2. Das Recherche-Tool bietet eine eigene Abfragesprache. Sollen auch Anfragen in einer projektspezifischen Form ermöglicht werden können, müssen diese spezifischen Anfragen zur TextGrid-weiten Suche entsprechend umgesetzt werden.



Schichten-übergreifende Grid Dienste

Authentifizierung, Autorisierung, Rechte, Sicherheit

Mitarbeiter von realen Organisationen bilden so genannte Virtuelle Organisationen (VO), um gemeinsam Rechnerressourcen, Daten und Dienste zu teilen. Autorisierungs- und Authentifizierungsvorgänge werden innerhalb einer solchen VO durchgeführt.

Eine VO ist ein zeitlich begrenztes oder permanentes Konsortium geographisch verteilter Mitarbeitern von realen Organisationen, Gruppen, ganzer Organisationseinheiten oder ganzer Organisationen, die Teile ihrer Ressourcen und Dienste, ihr Know-How für ein gemeinsames Ziel zusammenlegen und teilen.

TextGrid wird zunächst eine eigene Benutzerverwaltung auf Grundlage eines LDAP-basierten Verzeichnisdienstes aufbauen, welcher Grundlage sowohl für die Authentifizierung und Autorisierung (AA) sein wird. Die Authentifizierung wird in Form eines LDAP-Bind-Vorgangs durchgeführt, bei dem die Passwortübergabe über eine via SSL/TLS verschlüsselten Session übergeben wird. Für die Autorisierung werden zusätzlich zu den Authentifizierungsdaten (LoginID und Passwort) weitere Attribute gepflegt, die einerseits weitere Identifizierungsinformationen (E-Mail-Adresse, Kerberos Principle Id, etc.), Eigenschaften (Organisationszugehörigkeiten, etc.) sowie Rollenzugehörigkeiten abbilden. Über ein webbasiertes Interface können entsprechend Benutzerdaten gepflegt werden.

Eine solche Benutzerverwaltung kann für sich zunächst für AA-Vorgänge verwendet werden. Allerdings werden zusätzlich weitere Technologien implementiert, um eine grundsätzliche Skalierung zu gewährleisten. Zukünftig sollen ja nicht nur einige wenige Projektmitglieder Zugriff auf die TextGrid-Ressourcen erhalten, sondern eine beliebig große Anzahl von Benutzern. Dann macht es keinen Sinn, dass alle Benutzer zentral an einer Stelle verwaltet werden, Für eine solche Skalierung wird zusätzlich Shibboleth implementiert, welches Federated Identity Management ermöglicht. Dies bedeutet, dass Benutzer in Ihrer Heimatorganisation verwaltet werden und über einen Vertrauensbund Aussagen über Authentifizierung und Attribute über standardisierte XML-Strukturen (SAML-Assertions) ausgetauscht werden. Augenblicklich ist eine solche Föderation deutscher Hochschulen im Aufbau (DFN-AAI), die, sobald sie produktiv geworden sein wird, von TextGrid genutzt werden kann. So wird es möglich, z.B. allen Germanistik-Studenten der Universität Würzburg den Zugang zu TextGrid-Ressourcen über deren lokalen Benutzeraccount zu ermöglichen, also ohne diese in die TextGrid-Benutzerverwaltung aufnehmen zu müssen.

Auch vor der Existenz einer produktiven DFN-AAI bringt der Einsatz von Shibboleth Vorteile, weil durch diese Software ein Single Sign On realisiert werden kann, ein Benutzer sich also nur einmal authentifizieren muss und dann über einen definierten Zeitraum (z.B. 8 Stunden) für jeden Dienst als authentifiziert gilt. Es wird hierfür ein TextGrid Identity-Provider (IdP) implementiert, der auf den oben beschriebenen LDAP-Server aufgesetzt werden kann. Sobald an den Heimatorganisationen der Benutzer eigene IdPs entstehen, können die entsprechenden Benutzerdaten aus der TextGrid-Benutzerverwaltung gelöscht werden.

Durch die eigene TextGrid-Middlewareschicht ist es möglich, selbst eine Konvertierung bzw. ein Mapping der LDAP- bzw. Shibboleth-Authentifizierung zu X.509-Zertifikaten, wie sie im Grid benötigt werden, zu realisieren. Genau dies wird aber im Projekt GridShib implementiert, sodass anstelle einer Eigenentwicklung diese Software zum Einsatz kommen kann. GridShib ermöglicht es, die Authentifizierung bzw. Autorisierung und die Single Sign On-Funktionalität, die bei Shibboleth nur für Webdienste implementiert wird, auch für Prozesse, wie z.B. Web-Services zu verwenden.

Die Autorisierungsentscheidungen werden auf der Ressourcenseite (in unserer Architektur die über die TextGrid-Middleware verwalteten Informationsobjekte) über Zugriffsrechte, die sich auf Rollen und Attribute beziehen, realisiert. Die Software Permis bietet die Möglichkeit, Zugriffsrechte komfortabel zu spezifizieren und in standardisierten Formalen (XACML) abzulegen, sowie Zugriffsentscheidungen mittels solcher Policies zu treffen. Sowohl GridShib, als auch Permis werden im Rahmen des GAP-Projektes IVOM evaluiert und für den Produktiveinsatz implementiert. Die Ergebnisse dieses Projekts werden sukzessive in die TextGrid-Plattform einfließen. Es gibt verschiedene Rechteebenen (privat, Gruppe, öffentlich), sowie verschieden Rollen (Projektleiter, Bearbeiter, Benutzer, etc.), die im TextGrid-Szenarien-Text näher ausgeführt werden.

Zusätzlich zu den in den Heimatorganisationen gepflegten Attributen können weitere Merkmale von Benutzern, insbesondere Rollenzugehörigkeiten, in der VO selbst verwaltet werden. Auch hierzu kann ein zunächst eigener LDAP-Server implementiert werden. Hierüber kann z.B. ein Projektleiter Mitglieder in ein spezifisches Projekt aufnehmen, die ausschließlich Zugriffsrechte auf die Projektdaten erhalten. Auch hierfür gibt es im Gridbereich spezialisierte Software (VOMS und VOMRS). Insbesondere VOMRS kann mit relativ einfachen Mechanismen in eine GT4-Umgebung integriert werden, auch für den Betrieb mehrerer VOs. Dies ermöglicht, dass die inhaltlichen Projektleiter - der Textwissenschaftler - zuweisen kann, wer sich in seiner Arbeitsgruppe befindet und damit wer Projektdaten schreiben können soll. Neben direkten Projektmitarbeitern gibt es in einer AG auch Beobachter, die selektierte Daten vorübergehend lesen können.

Die Löschung von Daten ist nicht nur abhängig von den Zugriffsrechten: Sobald Daten veröffentlicht wurden (und damit evtl. von extern zitiert werden), dürfen sie nicht mehr gelöscht werden.

workflow, scheduling, resource brokerage

Die Workflow-Komponenten sind im Wesentlichen auf den zwei oberen Schichten (Benutzerumgebung und Service Layer) angesiedelt: der Workflow-Editor in der Benutzerumgebung und der Workflow-Enactor im Service Layer. Zusätzlich könnte man ein Scheduling in der Middleware-Schicht zum Aufgabenbereich des Workflows zählen.

Der Workflow-Editor ist ein interaktiver Bestandteil der Benutzerumgebung und erlaubt es, die Automatisierung von Arbeitsabläufen (Workflows) zu definieren. Hier können nicht-interaktive (also alle Streaming-) Tools koordiniert werden. Dazu erlaubt der Workflow-Editor eine weitgehend intuitive Spezifikation von Abläufen. Zu jedem einzelnen StreamingTool ist eine Maske verfügbar, über die es konfiguriert werden kann (Eingabe- und Ausgabedaten, falls nicht temporär, sowie weitere Konfigurationsoptionen des Tools). Hauptaufgabe des Workflow-Editors ist die Erstellung einer Prozessdefinition, die an den Workflow-Enactor übergeben wird. Eine zusätzliche, aber optionale Aufgabe des Editors ist das Anstoßen und die Überwachung der Ausführung des Workflows, welche im Enactor erfolgt.

Der Workflow-Enactor ist für die Ausführung des im Editor definierten Workflows zuständig. Die Zweiteilung Editor-Enactor hat sich bewährt und ermöglicht die Unabhängigkeit des Enactors von der konkreten Anwendung; hier kann deshalb auf vorhandene Software zurückgegriffen werden. Der Enactor empfängt die Prozessdefinition vom Editor und führt die darin spezifizierten Tasks in der festgelegten Abfolge aus. Im Falle von Textgrid muss der Enactor Web Services als Tasks ausführen können. Das Interface zum Enactor kann wiederum als Web Service realisiert werden; Funktionen zum Starten, Anhalten und Verwalten von verschiedenen Workflows sind hier verfügbar.

Ein Scheduling der Tasks innerhalb eines Workflows ist in der ersten Implementierungsphase nicht nötig, da als Tasks nur Web Services verwendet werden, die an eine bestimmte Maschine gebunden sind. Ein Scheduling von verschiedenen, ggf. konkurrierenden Workflows auf derselben Enactment Engine ist jedoch evtl. notwendig; eine gangbare Lösung wäre hierzu, jeden Workflow in einem separaten Thread zu realisieren und deren Koordination dem Betriebssystem zu überlassen. Ein „Daten-Scheduling“ (Staging) in der Middleware wäre dagegen für TextGrid wichtiger: z.B. werden in einem Workflow X die Daten A und B verarbeitet, diese könnten vom Scheduler vor Ausführung der Services zu einem Storage-Knoten in der Nähe der jeweiligen Service-Knoten repliziert werden. Dieses Replizieren soll möglichst im Hintergrund und vom Benutzer unbemerkt geschehen.

In einer zweiten Implementierungsphase soll der umgekehrte Weg erprobt werden: Nicht die Daten werden zum Service gebracht, sondern der Service möglichst nahe bei den Daten ausgeführt.

Service Level Management

Sobald TextGrid über den Prototypenstatus hinausgewachsen ist und eine größere Community TextGrid z.B. für Editionsprojekte nutzt, wird zwangsläufig die Erwartung bestehen, dass die Dienste und Daten in TextGrid rund um die Uhr („24/7“) verfügbar sind. Um dem gerecht werden zu können, müssen die Administratoren der angeschlossenen Rechen- und Storage-Knoten jederzeit über den Status (Auslastung, auftretende Fehler etc.) ihrer Systeme informiert sein. Wenigstens ein Teil dieser Informationen muss aber auch über das lokale System hinaus propagiert werden, um innerhalb TextGrids Scheduling-Entscheidungen treffen zu können, die systematische Datenreplizierung optimieren zu können usw.

Die Abfrage und Weiterverarbeitung dieser Statusinformationen soll in TextGrid mit den von GT4 MDS bereitgestellten Mitteln realisiert werden. MDS geht davon aus, dass für jede Ressource eine korrespondierende WS-Ressource existiert, im wesentlichen also ein zustandsbehafteter Web Service. Der Zustand dieses Web Services – technisch gesprochen die WS Resource Properties – spiegelt den Status der überwachten Ressource wieder und kann über das Aggregator Framework, das Teil von MDS ist, entweder regelmäßig abgefragt werden (polling) oder aktiv an geeignete Dienste gemeldet werden (subscription / notification).

MDS bietet bereits einen Index Service, in dem die Statusinformationen zahlreicher Ressourcen zusammengeführt und jederzeit abgefragt werden können. Anhand der aus diesem Index Service bezogenen Informationen kann beispielsweise der Datenscheduler entscheiden, in welchem Storage Knoten hinreichend Kapazitäten vorhanden sind, um hochaufgelöste Farbscans von Manuskripten zwischenzuspeichern; über WebMDS vermag andererseits auch ein Administrator, sich im Browser bequem einen Überblick über die in seiner Verantwortung liegenden Ressourcen zu verschaffen. (Dieser Mechanismus ist auch für die in die D-Grid Infrastruktur integrierten TextGrid-Knoten obligatorisch, um ein D-Grid-weites Monitoring zu ermöglichen. Voraussichtlich werden diese Datenquellen zugleich für TextGrid-interne Zwecke und für die Zustandsmeldung an die vom DGI betriebenen Aggregatoren genutzt werden können.) Ein Trigger Service erlaubt zudem, bei Überschreiten bestimmter Schwellenwerte (Plattenplatz nahezu erschöpft, Reaktionszeit eines Subsystems außerhalb des Toleranzbereiches o.ä.) automatisch Nachrichten an die zuständigen Administratoren zu versenden.

Die in GT4 enthaltenen Komponenten für das Monitoring sind also bereits hinreichend, um TextGrid aufzubauen. Sobald die Zahl der TextGrid-Nutzer steigt und die Menge der in TextGrid gespeicherten Daten wesentlich über dem Umfang der von den TextGrid-Partnern eingebrachten Korpora liegt, werden die Informationen über den augenblicklichen Ist-Zustand nicht mehr genügen, um einen effizienten Betrieb zu gewährleisten. Dann werden auch Daten aus der Vergangenheit benötigt werden, aus denen sich Trends ableiten lassen, um z.B. Kapazitätsengpässe rechtzeitig vorhersehen und vorbeugen zu können oder um z.B. die Replikationspolicies in TextGrid so anzupasen, dass unnötige Datentransfers vermieden werden können. Die Entscheidung, wie ein solcher Dienst, der die Geschichte der TextGrid-Ressourcen aufbereitet, im einzelnen aussehen muss – ob etwa ein einfaches Log dieser Daten genügt, ob Kennzahlen berechnet werden müssen, in welche die länger zurückliegende Geschichte nur noch schwächer gewichtet eingeht, o.ä. –, kann erst sinnvoll getroffen werden, wenn Erfahrungen mit dem Betrieb von TextGrid vorliegen, d.h. gegen Ende der Projektlaufzeit.

Die Projektpartner bringen im Rahmen des Projekts ausschließlich Daten in TextGrid ein, die frei von urheberrechtlichen Einschränkungen sind, etwa weil der Autor vor mehr als 75 Jahren gestorben ist. Deshalb sind in dem genehmigten Projekt keine Arbeitspakete vorgesehen, um etwa die Nutzung von Texten abzurechnen, die nur unter einer kommerziellen Lizenz verfügbar sind. Das Accounting zielt deshalb nicht in erster Linie darauf ab, Nutzungsdaten für eine spätere finanzielle Abrechnung zu erheben. Vielmehr dient es dazu, im Fall der missbräuchlichen Nutzung von TextGrid-Ressourcen die dafür verantwortlichen Anwender identifizieren und im Extremfall von der weiteren Nutzung ausschließen zu können.

Voraussetzung ist hierfür, dass für alle Aktionen in TextGrid eine eindeutig zuordenbare ID des Nutzers, der die Aktion veranlasst hat, in Log-Files gespeichert wird. Dies betrifft in erster Linie das Anlegen oder Modifizieren von Dateien bzw. Datensätzen, um unangemessen hohen Speicherplatzverbrauch oder das Speichern sachfremder Daten in TextGrid unterbinden zu können, sowie die Aufrufe von TextGrid-Services, um einer Blockade der TextGrid-Recheninfrastruktur vorzubeugen.

Diese Logfiles müssen unbedingt auch vor dem Lesezugriff Dritter geschützt sein, denn die Akzeptanz von TextGrid in der Community wäre erheblich gefährdet, wenn Nutzer nicht darauf vertratuen könnten, dass keine Außenstehenden anhand der Logfiles Details über die aktuellen, noch nicht zur Veröffntlichung vorgesehenen Forschungsdaten erfahren könnten.

Als ID können de facto nur die von der AAI bereitgestellten Benutzerinformationen dienen. Dabei muss es sich aber nicht notwendig um Klarnamen der Anwender handeln; beispielsweise kann ein Shibboleth Identity Provider so konfiguriert werden, dass das ausgestellte Authentisierungstoken lediglich pseudonymisierte Nutzerinformationen enthält. Die TextGrid-Logfiles enthielten dann nur dieses Pseudonym. Wenn die TextGrid-Administration Kontakt zu einem in einem Logfile genannten Anwender herstellen will, dann muss sie sich an die Betreiber des Identity Providers wenden; diesen obliegt dann zu entscheiden, ob das Interesse von TextGrid hinreicht, um eine Aufdeckung des betreffenden Pseudonyms zu begründen. Die Regeln, nach denen diese Entscheidungen getroffen werden, sind notwendiger Teil einer VO-Vereinbarung.

Wie oben ausgeführt sind derzeit in TextGrid keine technischen Mechanismen vorgesehen, um evtl. vorhandene Verwertungsrechte an Texten, die über TextGrid zugänglich sind, durchzusetzen oder die in diesem Zusammenhang fälligen Lizenzgebühren zu erheben. Das TextGrid-Konsortium sieht hier aber grundsätzlich einen Bedarf für Entwicklungen: denn nicht nur entstehen in TextGrid neue Werke mit eigenständigem Urheberrechtsschutz – deren Autoren entscheiden sich möglicherweise gegen eine Veröffentlichung innerhalb TextGrids, wenn sie ihr Verwertungsrecht nicht effektiv durchsetzen können. Es kann aber nicht im Interesse von TextGrid sein, wenn Texte, die bereits vollständig ausgezeichnet und für die Weiterverarbeitung mit TextGrid-Tools geeignet sind, dem möglichen Zugriff durch Dritte innerhalb der Plattform dauerhaft entzogen werden. Außerdem entgeht TextGrid so der größte Teil der zeitgenössischen Literatur oder Korpora, die selbstverständlich ebenfalls Gegenstand textwissenschaftlicher Arbeiten sind. Denn kaum ein Verlag wird bereit sein, die Texte, an denen er das Recht hält, in TextGrid zur Verfügung zu stellen, wenn er nicht weiter kontrollieren kann, wer zu welchen Konditionen den Text z.B. kopieren und ausdrucken darf.

In diesem Bereich besteht also nach wie vor Forschungs- und Entwicklungsbedarf, zumal den Verwertungsinteressen der Rechteinhaber ein Interesse der Forscher gegenübersteht, dass die Kosten für den Zugang zum Gegenstand ihrer Forschung, nämlich den Texten, nicht prohibitiv teuer wird und dass im Falle einer Lizenzierung die technische Durchsetzung der Rechte der Verlage und Autoren nicht den fairen Gebrauch der Daten verhindert. Angesichts des

notwendigen Aufwandes für eine Lösung, die diesen Interessen gerecht wird, muss dahingestellt bleiben, ob im Rahmen von TextGrid noch Ansätze hierfür entwickelt werden können. Eine vollständige Lösung kann auf jeden Fall nur in einem weitergehenden Projekt entstehen, optimalerweise eingebettet in ein umfassendes Projekt zum Service Level Agreement.

Archive

Metadaten und Datenobjekte werden in TextGrid in der Archiv-Schicht verwaltet. Sie ist Teil der TextGrid Middleware, wird aber aufgrund ihrer besonderen technischen und organisatorischen Eigenschaften separat aufgeführt.

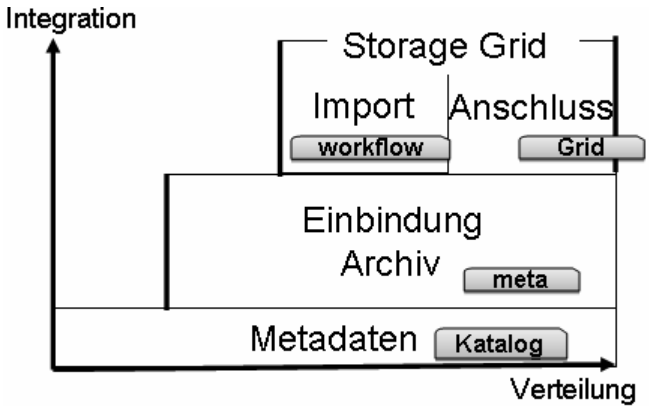
Archive sind in einem virtuellen Speicherraum verknüpft. Textarchive im In- und Ausland sind eingeladen, sich an diesen Speicherraum anzuschließen. Im Kern ist der Speicherraum ein Datengrid, mit allen Vorteilen der verteilten und redundanten Vorhaltung der Ressourcen bei deren gleichzeitiger Integration. Um existierenden Textarchiven den Einstieg so einfach als möglich zu machen, gibt es neben dem Datengrid auch die Möglichkeit über in der Bibliotheks-Community verbreitete Schnittstellen wie OAI oder Zing eine Einbettung der Datenobjekte zu erwirken.

Dadurch entstehen einige abgestufte **Schichten**, aus denen der TextGrid Speicherraum zusammengesetzt ist. Mit jeder Schicht steigt die Integration und damit auch die Möglichkeiten in TextGrid.

1. Metadatenintegration - Metadaten externer Kataloge (z.B. zvdd[9]) werden verlinkt; Datenobjekte oder Referenzen dazu liegen nicht notwendigerweise vor
2. Archiveinbindung - Metadaten werden importiert, die Datenobjekte bleiben in den Archiven

3a. Import - Import der Datenobjekte in die TextGrid Infrastruktur

3b. Datengrid Anschluss - externes Archiv betreibt Grid Infrastruktur



Mit jeder Schicht ist auch ein Archiv **Schlüssel** (archive layer code) verknüpft, der für eine optimale Integration der einzelnen Archive in das TextGrid sorgt. Dazu sind darin technische und organisatorische, sowie auch Aspekte zur Interoperabilität und Qualität behandelt:

- Struktur und semantische Verknüpfung
  - Metadatenmodell
  - Objektmodell
  - evtl. Selektionskriterien
- technische Integration
  - Protokolle (bzw. Grid Software Komponenten)
  - Anforderungen an die Verfügbarkeit des Systems

[9] Zentrales Verzeichnis Digitalisierter Drucke, zvdd (www.zvdd.de).

Die Verwaltung dieses Speicherraumes ist für den **Nutzer** weitgehend transparent. Einzig aus Einschränkungen und Möglichkeiten mancher Tools und Services (siehe Funktionenkatalog) kann der Nutzer Rückschlüsse ziehen:

- nur ein Teil der Referenzen im Bibliografietool sind auch als Digitalisate oder als Volltexte in TextGrid verfügbar.
- Spezialabfragen im Recherche-Tool sind nur auf die Untermenge der in TextGrid verfügbaren Objekte möglich, die im entsprechenden Spezialformat vorliegen.
- Bestimmte Tools verlangen eventuell ein Spezialformat. So verlangt z.B. der Print Publisher die von TextGrid für literaturwissenschaftliche Editionen empfohlene Auszeichnung[10].

Metadatenintegration

In der untersten Schicht werden Metadaten gesammelt und verwaltet. Die Herausforderung für diese Aufgabe ist das Zusammenführen heterogener Metadatenmodelle und das Abgleichen der Einträge. Mit dieser Integrationsarbeit beschäftigen sich eigenständige Projekte (z.B. zvdd oder Kalliope), auf deren Arbeit von TextGrid zurückgegriffen wird. Das Metadatenmodell von TextGrid ist mit dem von zvdd abgestimmt, und eine Integration daher relativ einfach möglich. Der *Schlüssel* hierzu ist:

- die Metadaten werden im entsprechenden TextGrid Metadatenformat angeboten (das interne Format ist hierbei nicht relevant, nur das nach außen hin angebotene)
- es existiert eine adäquate Schnittstelle
- die Richtigkeit der Daten wird durch das Textarchiv garantiert; etwaige Änderungen werden entsprechend an das TextGrid Metadatenimportmodul weitergegeben.

Archiveinbindung

Die Archiveinbindung baut auf der Metadatenintegration auf und bettet weitergehend existierende Datenobjekte in Textarchiven ein. Zur Durchsuchung der Volltexte oder weitergehenden Bearbeitung werden die Objekte in die TextGrid Umgebung importiert.

Die Kataloge und Archive werden über eine Schnittstelle abgegriffen und die Metadaten in der TextGrid Metadaten-Datenbank abgelegt. Textarchive können ihre Metadaten in TextGrid integrieren und damit Textwissenschaftler auf existierende Originale, Digitalisate, oder sogar Volltexte aufmerksam machen. Der *Schlüssel* zum Transfer der Metadaten ist:

- Metadatenintegration (unterliegende Schicht, siehe oben) ist gegeben
- die Informationsobjekte dürfen und können in TextGrid zur weiteren Verarbeitung transferiert werden

[10] Empfohlene Auszeichnungen, Metadaten, Versionierung, Konsistenzsicherung und andere Aspekte werden derzeit in einem TextGrid-Objektmodell entwickelt.

Datenimport

Nicht alle textwissenschaftlichen Projekte haben die Möglichkeit, ihre digitalen Objekte in einem Archiv vorzuhalten. Wo die Infrastruktur fehlt, können die Objekte direkt in die TextGrid Datengrid Umgebung eingeführt werden. Sollte dies Anklang finden und größere Datenmengen umgesetzt werden, könnten auf längere Sicht auch externe Dienste das Hosting von textwissenschaftlichen Daten anbieten und einen Knoten im TextGrid Datengrid bilden (Stichwort: Sustainability). Das textwissenschaftliche Projekt gewinnt dadurch

- Zugänglichkeit der Objekte
- Integration mit der TextGrid Umgebung und der vorhandenen Tools
- redundante Speicherung der Objekte

Um die Daten in die TextGrid Umgebung zu transferieren sind zwei Wege denkbar:

Eine Projektgruppe lädt ihre Einzeldaten manuell in TextGrid. Ein Modul zum Hochladen von relevanten Objekten unterstützt die Metadatenerzeugung so weit als möglich, aber de facto müssen viele Dinge von der Projektgruppe eingetragen werden.

- Aufbereitung der Metadaten nach dem TextGrid Metadatenformat
- Kompatibilität mit dem TextGrid Objektformat

Ist einem kleineren Archiv bereits die Metadatenintegration geglückt und haben die Metadaten eindeutige Referenzen auf die digitalen Objekte, dann können über das Metadatenimportmodul die gesamten Bestände abgegriffen werden.

- Metadatenintegration (unterliegende Schicht, siehe oben) ist gegeben
- Kompatibilität mit dem TextGrid Objektformat

Grid-Anschluss

Weitergehend als eine Archivintegration ist das Betreiben der TextGrid Middleware Komponenten zur Erstellung eines Datengrid bei den Textarchiven. Hierbei legt das Textarchiv seine Daten in einer von TextGrid entwickelten Software-Umgebung (inkl. Linux, Globus und DatenGrid-Komponenten) und in der empfohlenen Strukturierung ab. In einem ersten Schritt werden in Göttingen an der GWDG, im Rechenzentrum der Uni Würzburg, und am Institut für Deutsche Sprache in Mannheim Datengrid Knoten aufgebaut.

Das Betreiben eines Datengrid Knotens bietet für die Textarchive eine Reihe von Vorteilen:

- Teil der TextGrid Software-Umgebung ist ein Ablage-/ Document Management System für die Textarchive
- die empfohlenen Strukturen und Metadaten fördern Qualität und Interoperabilität
- Daten können zwischen Datengrid Knoten repliziert werden, was deren Sicherung und Verfügbarkeit entscheidend steigert

Um einen Datengrid Knoten aufzubauen, gilt dieser *Schlüssel*:

- Metadatenintegration (unterliegende Schicht, siehe oben) ist gegeben
- die TextGrid Software Umgebung läuft vor Ort, die entsprechenden Schnittstellen sind offen und von außen zugänglich

- zusätzlich zur Erfüllung des Eigenbedarfs gibt es einen Speicherbereich, in dem Daten von anderen Knoten repliziert werden können.

Conclusio

TextGrid etabliert e-Science in den Geisteswissenschaften und vereint dabei unterschiedliche Communities und deren Ansätze und Perspektiven: die Anforderungen und Ansätze der Textwissenschaften zur Erschließung und Bearbeitung wissenschaftlicher Texte; Grid Technologien für eine verteilte, generische Infrastruktur; und Konzepte zu Datenmanagement und Interoperabilität aus der Digital Library / Repository Community. Die Kombination dieser verschiedenen Bereiche ermöglicht den Aufbau einer offenen, homogenen Umgebung, die spezialisierte Anwendungen unterstützt und Kooperation und Partizipation ermöglicht.

Die Referenzarchitektur von TextGrid ist in vier Schichten aufgebaut: Benutzerumgebungen, Service Layer, Middleware und Archive. In jeder dieser Schichten kann sich eine unterschiedliche Zielgruppe in TextGrid einklinken: die Benutzer arbeiten in ihren Benutzerumgebungen; Spezialprojekte können aufbauend auf den vorhandenen TextGrid Funktionalitäten neue Funktionen im Service Layer integrieren; und Archive können in einem Archivverbund den Datenerhalt durch Replikationsmechanismen sicherstellen.

Das Grid findet analog auf verschiedenen Ebenen und für unterschiedliche Ziele statt:

1. ein Datengrid ermöglicht die Replikation und die Integration heterogener Daten;
2. Basisdienste bekommen in der TextGrid Middleware Zugang zu Gridressourcen, wodurch die Skalierbarkeit und Performanz der TextGrid Infrastruktur gewahrt wird; und
3. das Service Grid erlaubt die Integration lokaler Werkzeuge in einen homogenen, auf einander aufbauenden Dienstekatalog.

Vor allem drei Eigenschaften definieren jeden Aspekt der TextGrid Architektur und auch des Gesamtprojektes:

- TextGrid ist mehrschichtig **modular** aufgebaut, und folgt dem Paradigma einer Service Oriented Architecture. Dies unterstützt die Offenheit und Erweiterbarkeit von TextGrid, erlaubt eine iterative Entwicklung, ermöglicht eine funktionale Breite angepasst an die Aktivitäten der Community, und stellt die Robustheit des Systems sicher.
- Jedes Modul ist so **generisch** als möglich gehalten. Erst durch die Summe der einzelnen Module in der Kombination und Konfiguration durch den Nutzer oder in einer Benutzerumgebung wird eine komplexe und auf die spezielle Anwendung zugeschnittene Funktionalität erzeugt. Dadurch bleibt maximale Flexibilität erhalten, und die Wiederverwendbarkeit von existierenden Modulen wird gefördert. Dies betrifft sowohl einzelne Services, vor allem aber auch Kernfunktionalitäten und Basisdienste in der Middleware, die möglichst wenig inhaltliche Ausrichtung vorgeben und damit keine Vorgaben für die Nutzung machen.
- An verschiedenen Stellen realisiert TextGrid eine **stufenweise Integration**. Die fördert wiederum die Flexibilität des Gesamtsystems, und senkt die erste Einstiegshürde. Mit jeder Stufe kommen zwar neue technische wie organisatorische Anforderungen auf den Nutzer zu, jede Stufe bringt aber auch neue Möglichkeiten und Funktionalitäten in TextGrid.
  Die Integration der Archive ist abgestuft, und reicht von der Einbindung ihrer Metadaten in TextGrid bis zur Teilnahme am Datengrid (siehe Kapitel). Neue Funktionalitäten können zuerst lose als Service in TextGrid registriert werden, um dann

in einem weiteren Schritt als Basisdienst direkt in die Middleware integriert zu werden (siehe Kapitel). Nicht zuletzt die Empfehlungen für Formatschemata sind abgestuft[11], und jede von dem Nutzer übernommene Empfehlung eröffnet neue Funktionalitäten in TextGrid.

[11] Empfohlene Auszeichnungen, Metadaten, Versionierung, Konsistenzsicherung und andere Aspekte werden derzeit in einem TextGrid-Objektmodell entwickelt.

Anhang: Architektur Skizze (27.6.2006)