TextGrid
Virtuelle Forschungsumgebung
für die Geisteswissenschaften

Architekturskizze inklusive Identifikation der Lücken und der Optimierungsmöglichkeiten

(R 4.5.1)

Version 22. Februar 2013
Arbeitspaket 4.5
Verantwortlicher Partner GWDG

TextGrid
Virtuelle Forschungsumgebung für die Geisteswissenschaften

GEFÖRDERT VOM
Bundesministerium
für Bildung
und Forschung

Projekt: TextGrid - Vernetzte Forschungsumgebung in den eHumanities
BMBF Förderkennzeichen: 01UG1203A
Laufzeit: Juni 2012 bis Mai 2015

Dokumentstatus: final
Verfügbarkeit: öffentlich
Autoren:

Fatih Berber, GWDG
Stefan E. Funk, DAASI International GmbH
Ubbo Veentjer, SUB Göttingen
Martin Haase, DAASI International GmbH
Philipp Wieder, GWDG
Peter Gietz, DAASI International GmbH

Revisionsverlauf:

Datum	Autor	Kommentare
23.11.2012	Fatih Berber	Erste Version
30.01.2013	Stefan E. Funk	Formatierung und Struktur
31.01.2013	Stefan E. Funk	Ergänzungen TG-auth, TG-crud und TG-search
12.02.2013	Ubbo Veentjer	Überarbeitung TG-search
14.02.2013	Martin Haase	Überarbeitung TG-auth
14.02.2013	Stefan E. Funk	Allgemeine Änderungen und Überarbeitung
19.02.2013	Philipp Wieder	Einleitung; Anpassungen
21.02.2013	Peter Gietz	letzte Anpassungen, insbesondere bei TG-auth und TG-crud
22.02.2013	Stefan E. Funk	Abschlussformatierung

Inhalt

1 Einleitung

TextGrid stellt eine produktive Forschungsumgebung für die textorientierte Geisteswissenschaften und zunehmend auch für andere Disziplinen bereit. Ziel der aktuellen Förderphase ist es, diese Umgebung zu verstetigen und nachhaltige Konzepte und Strukturen zu erarbeiten, die Forscher auch in der Zukunft bei ihrer Arbeit unterstützt.

Eine wichtige Säule bei der Planung der Verstetigung von TextGrid ist die Softwareumgebung, welche die Basis für die Forschungsumgebung darstellt. Hier ist es notwendig, die existierende Architektur und Umsetzung in regelmäßigen Abständen mit den Kundenwünschen, Nutzerzahlen, Dienstgütedaten und aktuellen technischen Entwicklungen abzugleichen und zu bewerten. Daraus ergeben sich in der Regel Lücken (Stichwort *gap analysis*). Die Kosten und Zeitaufwände für das Schließen dieser Lücken sind anschließend zu erheben und eine Risikoabschätzung einzelner Lücken bezüglich deren Umsetzung zu machen. Ausgehend davon lässt sich dann eine Roadmap für die Umsetzung formulieren. Grundsätzlich geht es hier eher um eine Verstetigung der bisherigen Funktionalität als um die Entwicklung ganz neuer Funktionalitäten.

Der vorliegende Bericht setzt die oben genannten Punkte für die TextGrid Middleware um und gibt eine konsistente Analyse existierender Lücken mit Fokus auf die Optimierung von Performanz, Skalierbarkeit und Ausfallsicherheit. Der Bericht wird kontinuierlich aktualisiert und der Fortschritt der Arbeiten dokumentiert. Eine technische Roadmap über die gesamte Projektlaufzeit steht noch aus, da die Umsetzung einiger Punkte noch im Konsortium diskutiert werden muss.

Die vorliegende Version stellt aber einen stabilen Zwischenstand dar, der für die Planung der weiteren Projektphasen wichtigen Input gibt.

2 Analyse der TextGrid Middleware

Die TextGrid-Middleware besteht aus den Hautkomponenten TG-auth, TG-crud und TG-search. Diese stellen die grundlegenden Funktionen bereit, die für das TextGrid Repository benötigt werden, und bilden somit das Rückgrat der virtuellen Forschungsumgebung für die Geisteswissenschaften. Daher ist es essentiell, dass diese drei Komponenten nicht nur zuverlässig arbeiten, sondern dass sie auch bei steigenden Nutzerzahlen gleichbleibende Performanz garantieren. Zudem muss sichergestellt werden, dass die TextGrid-Middleware technologisch den Anschluss behält und keine teuer zu pflegende oder bereits abgekündigte Software Bestandteil der Middleware bleibt.

Aus diesem Grund wurden die in der Version 2.0 umgesetzte Architektur und die entsprechenden Technologien auf den Prüfstand gestellt, Lücken identifiziert und Lösungsvorschläge für das Schließen solcher Lücken erarbeitet. Die Ergebnisse werden in diesem Abschnitt vorgestellt, wobei folgende Struktur für gewählt wurde: (1) Beschreibung der Komponente, (2) Identifizierte Lücke(n), (3)

Optimierungsmöglichkeiten und (4) erste Abschätzung der Machbarkeit. Der Vollständigkeit halber ist das Zusammenspiel der drei Komponenten mittels einer Architekturskizze im Anhang dokumentiert (siehe Abschnitt 3.1).

2.1 TG-auth

TG-auth ist die zentrale infrastruktur, die die notwendigen Funktionalitäten für (föderierte) Authentifizierung und Autorisierung, sowie Zugriffskontrolle zur Verfügung stellt. TG-auth besteht aus folgenden Bestandteilen:

- OpenLDAP[1] (eine Implementierung des Standards LDAP) basierter Authentifizierungsserver, in dem TextGrid-Benutzer-Acounts angelegt und verwaltet werden können, für die Benutzer, die sich (noch) nicht über die DFN-AAI-Föderation über ihren lokalen Campus-Account anmelden können.

- OpenRBAC[2] eine Implementierung des Standards RBAC (Role Based Access Control), über die die Zugriffskontrolle auf alle TextGrid-Objekte festgelegt werden kann. Diese Komponente verwendet wiederum einen OpenLDAP-Server als Datenbackend für Rollen, Ressourcen Zugriffsrechte, etc. Standard RBAC-Funktionen sind u.a. über Web-Services zugreifbar.

- Tgextra, TextGrid-spezifische Webservices zur TextGrid-spezifischen Erweiterungen der OpenRBAC-Funktionalität

- Shibboleth[3], eine Implementierung des Standards SAML (Security Assertions Markup Language), wodurch sich sowohl Benutzer im oben erwähnten LDAP Server als auch Benutzer, deren Heimatorganisation im Rahmen der DFN-AAI an TextGrid-Dienste einen eindeutige ID schicken, an TextGrid anmelden können

- WebAuthN, eine Web-Anwendung für Browser-basierte Authentifizierung, welche

- LDAP- und SAML-basierte Authentifizierung ermöglicht

2.1.1 Identifikation von Lücken

Der Dienst TG-auth hat Probleme:

- bei komplexen Operationen und beim gleichzeitigen Zugriff vieler Nutzer und dem daraus resultierenden lesenden und schreibenden Zugriff auf das Rechtesystem (über OpenRBAC), insbesondere bei der Methode `createProject`

- bei filternden Methoden, über welche sehr viele URIs (> 100.000) angefragt werden und diejenigen mit entsprechenden Rechten zurückgegeben werden.

---

1 OpenLDAP: http://www.openldap.org. Letzer Zugriff 21.02.2013
2 openRBAC: http://www.openrbac.de/en_startup.xml. Letzter Zugriff 19.02.2013.
3 Shibboleth: http://shibboleth.net/. Letzter Zugriff 20.2.2013.

Die oben beschriebenen Komponenten sind zwar auf häufig vorkommende Operationen (insbesondere `checkAccess` und beliebige andere Leseoperationen) optimiert, aber eine hohe Anzahl vieler gleichzeitig auftretender komplexerer schreibender Operationen kann sich als problematisch herausstellen. Auch ist TG-auth Stand heute nicht redundant ausgelegt.

2.1.2 Optimierungsmöglichkeiten

Aus der Analyse des Status Quo ergeben sich drei Empfehlungen für die Verbesserung von TG-auth[4]:

Skalierbarkeit und Ausfallsicherheit OpenRBAC (F001)
Redundanz der Middleware durch weitere TG-auth Service-Knoten (also jeweils OpenRBAC+OPenLDAP+*tgextra*), wobei jeweils Knoten für verschiedenen Aufgaben, die jeweils wiederum mehrfach vorhanden sein können, sinnvoll sind. Folgende tgextra-Funktionalitäten lassen sich hierbei trennen:

- *tgextra-write* (nur schreibender Zugriff für allgemeine Funktionen)
- *tgextra-crud* (auch schreibender Zugriff für Funktionen, die nur TG-crud nutzen darf)
- *tgextra-readonly* (nur lesender Zugriff auf allgemeine Funktionen), parallel nutzbar, da die hier genutzten LDAP-Knoten als „Slave“ nur lesend genutzt werden

Weiterhin besteht die Möglichkeit, weitere allerding eher marginale Optimierungsmöglichkeiten auch im Code von OpenRBAC zu untersuchen. Insbesondere liegt im Hinblick auf Transaktionssicherheit noch weiteres Potential.

Skalierbarkeit und Ausfallsicherheit LDAP (F002)
Lesender Zugriff auf das LDAP ist durch weitere LDAP-Knoten (Master, Slave) skalierbar, wobei allerdings nur auf dem LDAP Master geschrieben werden darf. Eine weitere Möglichkeit ist die Vorhaltung von mehreren LDAP-Servern im Multi-Master-Modus, von denen auf jeden geschrieben werden kann. Weitere Lese-Replikate (Slaves) sind quasi unbegrenzt hinzufügbar. Außerdem ist durch entsprechende Datenbank Back-Ends und Aufrüstung der unterliegenden Hardware (beispielsweise durch SSD Platten) eine sowohl hohe Schreib- und als auch Lese-Performanz erreichbar. Insgesamt ergibt die Auswertung der aktuellen Situation dass der LDAP-Server sehr gut für auch wesentlich höhere Auslastung gerüstet ist.

Skalierbarkeit und Ausfallsicherheit WebAuthN (F003)
Die Webseite kann relativ einfach geclustert werden, da sie mit Ausnahme der folgenden Punkte keinen Status halten und/oder teilen muss.

[4] Alle Verbesserungsmöglichkeiten werden fortlaufend mit *Fxxx* gekennzeichnet (*xxx* bei 001 beginnend). Dies dient der eindeutigen Identifikation einer Verbesserung (in der Softwareentwicklung *feature request* genannt) in der Kommunikation innerhalb des Projektes.

- Zugrundeliegender Community-OpenLDAP-Server: hier gilt dasselbe wie beim OpenRBAC-LDAP-Server

- Clustering des Service Providers: Aktuell wird der SP, verglichen mit den Community-Anwendungen, kaum genutzt. Man kann den SP gleichwohl clustern, indem entweder die Sessions (diese sind statusbehaftet) in eine gemeinsame über ODBC erreichbare Datenbank geschrieben werden oder indem ein gemeinsamer Shibboleth Demon für alle Instanzen vorgehalten wird. Die Option, die Sessions gar nicht zu teilen, ist auch möglich, wenn der Load-Balancer eine entsprechende Cookie- oder IP-Stickiness für ca. 5 Minuten (die maximale Dauer eines typischen Login-Vorgangs) implementiert.

Hier wäre ein normaler Load-Balancer mit dem vorhandenen Proxy ausreichend.

2.1.3 Abschätzung der Machbarkeit

Die Optimierungsmöglichkeiten F001 und F002 sind primär durch Konfiguration der Middleware zu realisieren, wobei die Untersuchung des Codes von openRBAC zeitlich noch nicht abgeschätzt werden kann. Dazu sind weitere Auswertungen notwendig. In F002 und F003 integriert sind Anforderungen an die betriebliche Infrastruktur (Hardware-Upgrade für LDAP, Load-Balancer). Beide Anforderungen sind zu erfüllen, allerdings sind in Absprache mit dem Betreiber der Ressourcen Kosten und Zeitrahmen der Umsetzung noch zu prüfen.

2.2 TG-crud

TG-crud nimmt die Anfragen der Klienten an, bekommt Daten und Metadaten geliefert und verteilt diese Daten dann – je nach Methode – auf die unterschiedliche Datensenken. Dabei existieren in der Middleware folgende Instanzen: eXist[5] (XML-Datenbank), Sesame[6] (RDF-Datenbank, Triple-Store), TG-auth (tgextra-crud) und der StorNext-Knoten, wodurch über JavaGAT[7] auf das Speicher Back-End zugegriffen wird (anstelle von JavaGAT kann augenblicklich auch Fedora als Datenbackend angesprochen werden). Es wird erst eine Antwort zurückgesendet, wenn alle diese Zugriffe erfolgreich waren (synchrone Operation), falls nicht, wird ein Fehler zurückgemeldet.

2.2.1 Identifikation von Lücken

Der Dienst TG-crud selbst ist bereits für einen parallelen Betrieb ausgelegt, und wird schon in zwei Versionen benutzt, eine öffentliche und eine nicht-öffentliche. Einziger gemeinsamer Knoten für verschiedene TG-crud-Instanzen ist der Dienst TG-noid, der die TextGrid-URIs generiert (diese müssen im gesamten TextGrid-Namespace eindeutig sein), und weiterhin werden damit das TG-crud-interne und das externe Locking von Objekten über deren TextGrid URIs kontrolliert.

---

5 eXist-db: http://exist-db.org/exist/apps/homepage/index.html. Letzter Zugriff 20.2.2013.
6 openrdf.org – ... home of Sesame: http://www.openrdf.org/. Letzter Zugriff 20.2.2013.
7 JavaGAT: http://www.cs.vu.nl/ibis/javagat.html. Letzter Zugriff 20.2.2013.

Die TextGrid-Objekte, die aus Daten und Metadaten bestehen, werden durch TG-crud momentan über eine JavaGAT-Implementation der Storage-Schnittstelle im Speicher Back-End gespeichert. Da JavaGAT z.B. kein iRODS-Modul anbietet, und der lokale JavaGAT File-Adapter nur über gemountete Dateisysteme genutzt werden kann, bietet sich hier ein Wechsel der Interface-Implementation an.[8]

Dies würde zwar sehr wahrscheinlich auch Performanzvorteile bieten, die Flexibilität, die durch JavaGAT geboten wird, geht jedoch verloren.

2.2.2 Optimierungsmöglichkeiten

Parallelisierung (F004)
Ein Test von mehreren TG-crud Instanzen, die parallel genutzt werden (schreibend wie lesend) ist erforderlich. Die Implementierung ist auf parallelen Betrieb ausgelegt, wurde jedoch noch nicht eingehend daraufhin getestet, ebenso nicht die Skalierbarkeit.

Die Ausfallsicherheit von TG-noid ist noch nicht implementiert, dies soll im Zuge der High Availability-Implementierung bedacht werden. Eine Skalierbarkeit von TG-noid ist gewährleistet.

Storage-Interface (F005)
Als Alternative für den Zugriff auf den Storage bietet sich eine der Java-Implementationen von SAGA[9] an. Dazu ist vorerst eine Evaluierung von SAGA nötig, es muss geprüft werden, ob es einen iRODS-Adapter für SAGA gibt, und ob dieser für TextGrid dienlich ist. Außerdem muss die Frage beantwortet werden, ob man einen Layer wie SAGA zwischen TG-crud und dem Storage überhaupt benötigt. Die Entwicklung einer standardisierten Schnittstelle stellt eine Notwendigkeit für die Wartbarkeit des System dar. Jedoch sind diese Evaluierungen und Entwicklungen eher mittel- bis langfristige Lösungen.

Kurzfristig bietet sich der direkte Zugriff auf das Storage-Backend an. In TG-Crud ist bereits jetzt ein Interface als Kapselung der Backendzugriffe realsiert (worunter jetzt Fedora oder JavaGAT angesprochen werden), sodass über dieses Interface weitere Backend-Module, z.B. für iRODS realisierbar sind. Die Aufwandsabschätzung für einen Einsatz von iRODS als Backend beträgt drei Monate.

Die GWDG entwickelt derzeit ein Frontend inclusive API für iRODS, das einen Zugriff ermöglicht, ohne iRODS auf dem TextGrid-Server zu installieren. Jeder Zugriff auf iRODS wäre dann über die iRODS-Implementation des TG-crud Storage-Interfaces möglich.

Storage Backend (F006)
Die aktuelle Anbindung an StorNext der GWDG limitiert die Nutzung anderer Backends. Einen Ausweg aus dieser Problematik stellt iRODS dar. iRODS

[8] Siehe TextGrid Middleware-Dokumentation. https://dev2.dariah.eu/wiki/display/TextGrid/Main+Page (14. Februar 2013).
[9] Siehe SAGA Implementations-Homepages: JSAGA. http://grid.in2p3.fr/jsaga/ (14. Februar 2013) und JavaSAGA (Status unbekannt).

(integrated Rule Oriented Data System) ist ein Open-Source Daten-Grid. Es ermöglicht die Verwaltung von großen und verteilten Daten. Abbildung 1 zeigt die Komponenten des iRODS Systems.

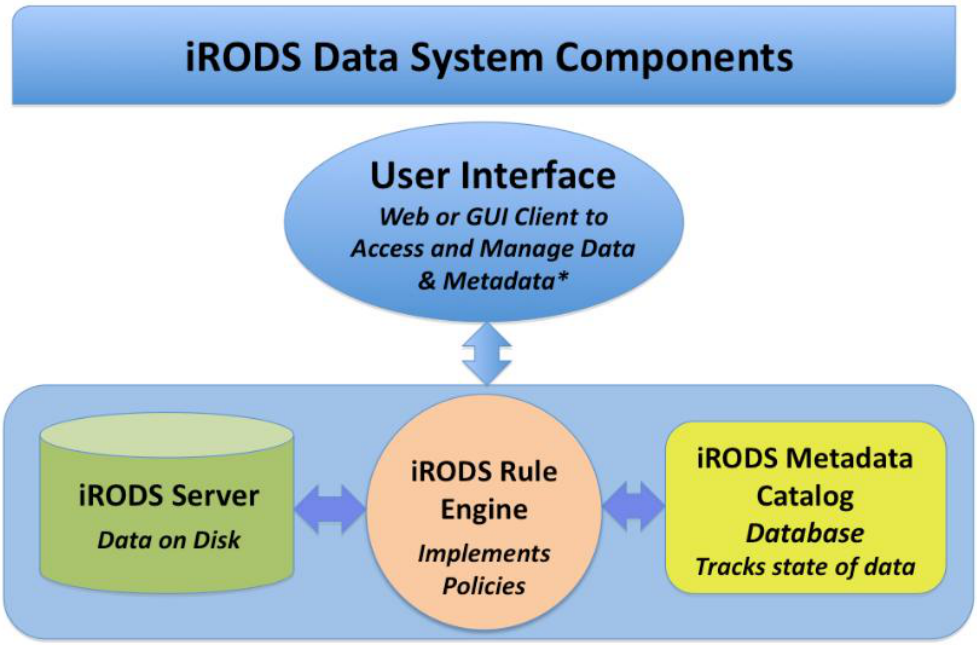


Abbildung 1: Komponenten des iRODS Systems

Der iRODS Metadata Catalog (ICAT) ist eine Datenbank, die Zustandsinformationen zu den Daten speichert. Da die Daten über verschiedene Ressourcen verteilt sein können, werden bspw. die physikalischen Speicheradressen im ICAT festgehalten.

Die iRODS Rule Engine sorgt dafür, dass beim Ingest bestimmte Aktionen für Dateien ausgeführt werden. Eine mögliche Aktion wäre beispielsweise die Replikation auf andere Ressourcen.

Die Einrichtung und Konfiguration von iRODS als Backend mit den Basisfunktionen dürfte einen Monat in Anspruch nehmen.

Denkbar wäre weiterhin die Nutzung von Fedora Commons als unterliegendes Storage-System auszubauen. Da Fedora jedoch als System sehr komplex ist, und die TextGrid-Middleware bereits viele der in Fedora vorhandenen Funktionen (zum Beispiel Versionierung, einen Triple-Store, Suchfunktionalität), wäre der Ansatz, alle diese bereits vorhandenen Funktionen von der TextGrid-Middleware in Fedora zu schieben. Dies würde einen sehr hohen Aufwand mit sich bringen. Um die Stärken von Fedora voll ausnutzen zu können, müssten weitreichende Änderungen an der TextGrid-Middleware vorgenommen werden, die in keinem Verhältnis zum Nutzen stehen würden.

Integration in die DARIAH Bit Preservation (F007)

Die Zusammenführung von TextGrid und DARIAH im Bereich Storage ist von beiden Seiten gewünscht; um Ressourcen zu bündeln, die eHumanities in den Bereichen Storage zu stärken und an dieser Stelle existierende Synergien zu nutzen. Die Integration der DARIAH Bit Preservation[10] (DBP) in die TextGrid Middleware ist eine der Möglichkeiten, dies zu fördern. Grundsätzlich wäre hier die Idee, die DPB als unteren Dienst einer LZA-STrategie zu verwenden, auf die höhere LZA-Dienste von der TextGrid-Middleware (Formatvalidierung etc.) aufbauen. Für eine Kooperation von TextGrid und DARIAH ist es notwendig, solche Nutzungsszenarien und Anforderungen aufzuzeigen und schließlich eine Machbarkeitsstudie für den Einsatz in TextGrid durchzuführen. Eine Implementation eines Storage-Interfaces innerhalb TG-crud, das über die DARIAH Storage-API auf Storage zugreift, ist ein nächster Schritt.

2.2.3 Abschätzung der Machbarkeit

Die Machbarkeit aller oben genannten Punkte wurde analysiert und generell ist die Umsetzung aller Punkte im Rahmen des Vorhabens machbar. F004 bedarf noch weiterer Analysen, um hier den benötigten Zeitaufwand zu bestimmen, allerdings ist dieser vermutlich im Rahmen eines Monats. Für F005 ist ein Aufwand von drei Monaten veranschlagt; die Umsetzung wird als essentiell angesehen und in Kürze angegangen. F006 ist aktuell bereits umgesetzt worden, wobei initial ein Monat veranschlagt wurde. Auch für F007 wurde ein Monat veranschlagt. Dieser Punkt kann angegangen werden sobald F006 vollständig verfügbar ist.

2.3 TG-search

TG-search ist ein Kerndienst in TextGrid, der die Suche über alle Daten ermöglicht. Er ermöglicht die Volltextsuche über alle (vom Nutzer lesbaren) TEI-Daten oder die Suche nach Metadaten. Desweiteren stellt TG-Search Navigationsinformationen (z.B. für den Navigator im TextGridLab) und Revisionsinformationen zur Verfügung.

TG-search greift für die Suche nach Meta- und Struktur- (Volltext-) Daten auf die eXist XML-Datenbank zu. Ein Teil der Funktionalität liegt in Form von XQuery-Modulen in eXist vor.

Für die Navigations- und die Revisionsinformationen, sowie das Filtern nach den letzten Revisionen eines Objektes, nutzt TG-Search Sesame als RDF-Speicher. Anfragen werden in SPARQL gestellt und die Ergebnisse werden für die Suche ausgewertet.

Für den nicht öffentlichen Bereich des Repositories werden TG-Search Treffer mit TG-auth abgeglichen um festzustellen, ob der Nutzer die Suchtreffer angezeigt bekommt. Im öffentlichen Bereich des Repositories wird TG-auth nicht angefragt.

[10] DARIAH Storage API – A Basic Storage Service API on Bit Preservation Level. http://hdl.handle.net/11858/00-1734-0000-0009-FEA1-D (14. Februar 2013)

2.3.1 Identifikation von Lücken

Beim bisherigen Betrieb des TextGrid Repositories gab es immer wieder Probleme mit der TG-search zugrunde liegenden eXist XML Datenbank. Es wurden verschiedene Strategien zur Lösung angewandt die, auch immer wieder zum Ziel geführt haben, allerdings ist feststellbar, dass mit wachsendem Datenbestand ein bei einem Ausfall nötiger Re-Index (oder gar eine Restaurierung) des Datenbestandes immer zeitaufwändiger wird. Bei einem Ausfall der eXist-Datenbank ist derzeit das gesamte Repository unbenutzbar, da TG-search auch die Daten für die Navigation und das Browsing im TG-lab liefert. Außerdem ist die eXist-Datenbank zwingend nötig für das Einspielen neuer Objekte (siehe TG-crud). Die Downtime bei einem eXist Ausfall ist dann mindestens die Zeit zur Wiederherstellung des Indexes[11], und möglicherweise die für weitere Fehlersuche. Besonders schwer wiegt hierbei, dass eXist derzeit nicht verteilt aufgesetzt ist, das heißt, es gibt nur eine Instanz und somit einen Single-Point-of-Failure.

Es gibt weiterhin Perfomanzprobleme mit der jetzigen Suche im TG-Search, die vermutlich durch eine Änderung der Suchstrategie und Optimierung von Indizes behoben werden könnten.

Insbesondere problematisch ist auch die Nutzer / Projekt-basierte Authentifizierung im nicht öffentlichen Repository, da hier in Theorie jeder einzelne Volltexttreffer mit einer Liste der dem Nutzer einsehbaren Ergebnisse abgeglichen werden muss. Auch an dieser Stelle wurde noch keine optimale Lösung gefunden. Derzeit wird bei TG-auth eine Liste der dem Nutzer einsehbaren Projekte eingesammelt, und diese wird als Grundlage genommen, um Sesame nach einer Liste aller dem Nutzer zugänglichen URIs zu fragen, diese werden dann im XQuery im eXist mit einer Liste aller Treffer abgeglichen. Auch dieses Vorgehen birgt bei wachsendem Datenbestand, auf den ein Nutzer Zugriff haben könnte, die Gefahr zukünftiger Performanzprobleme. Beim öffentlichen Repository kann bei der Suche komplett auf Authentifizierung verzichtet werden, daher stellt sich das Problem dort nicht.

2.3.2 Optimierungsmöglichkeiten

Da TG-search und eXist eine Kernfunktionalität des Repositories bieten, ist es angeraten, auch hier auf eine stabilere Lösung zu setzen.

Ersatz für eXist (F008)
Als Ersatz-Produkt für eXist ist Apache Solr empfehlenswert. Apache Solr ist eine hochperformante Suchmaschine, vor allem die Skalierbarkeit von Solr bringt dieses System in den Fokus. Der Suchindex kann sehr einfach auf andere Server repliziert werden.

Solr benutzt die Suchbibliotheken von Lucene, welche auch in eXist zum Einsatz kommen. Dass mit Solr nicht wirklich XML-Strukturen gezielt durchsucht werden

[11] eXist war mit dem wiederherstellen eines Backups des öffentlichen Repositories über 24 Stunden beschäftigt, allerdings kam hier Pairtree zum Einsatz, weswegen mehr als doppelt so viele Collections angelegt werden mussten.

können, ist zunächst ein Nachteil, ein Ausweg aus dieser Problematik könnte jedoch eine gleichzeitiger Einsatz von Solr und eXist sein. Hier übernimmt Solr dann die skalierbare und ausfallsichere Seite von TG-search, kombiniert mit Sesame. Der Einsatz von eXist ermöglicht dann die strukturierte Suche über XML-Daten, von der dann nicht mehr das gesamte System abhängig ist.

Der erste Schritt ist die Installation eines Solr Servers. Dann muss TG-search an die neue Solr-Komponente angepasst werden. Da die Komponente TG-crud momentan ebenfalls auf die eXist-Datenbank zugreift, muss TG-crud ebenfalls an diese Änderung angepasst werden. Um einen parallelen Einsatz von Solr und eXist zu realisieren, muss eine effektive Verteilungsstrategie konzipiert werden. Sesame sollte ebenfalls unter dem Aspekt der Verteilbarkeit untersucht werden.

Die neukonzipierte Suche sollte dabei insbesondere auch Features bieten, mit denen die Fachwissenschaftler die aufwendig annotierten Metadaten nutzen können: Dazu gehören etwa

- Suchen über Zeiträume
- Kombination der Werkinformation (z.B. Entstehungszeit, Genre) mit Editions- bzw. Itemdaten (z.B. »Alle Romane aus dem 18. Jahrhundert, in denen ›Liebe‹ vorkommt«)
- Refinement der Suchen über Facetten, die aus den Metadaten aller Ebenen gebildet werden
- Suche in Korpora / Kollektionen öffentlicher Daten, die vom Benutzer zusammengestellt (aber nicht notwendigerweise veröffentlicht) werden
- Unterstützung von Vokabularen in den Keywords, z.B. falls Erstdruckdaten über temporal-Keywords codiert werden
- Einbeziehung von externen Normdateien, z.B. bei PND-Link: Suche nach Romanen weiblicher Autoren, die im 19. Jh. entstanden sind (+ ggf. Volltexte)

Die Erfahrungen mit der Digitalen Bibliothek haben gezeigt, dass es insbesondere bei Textsorten wie Enzyklopädien nicht immer sinnvoll ist, die Texte bis auf die unterste mit Metadaten zu adressierende Ebene in einzelne XML-Dokumente zu zerlegen. Es muss deshalb möglich sein, bei der Suche (und z.B. bei der Breadcrumbanzeige in Suchergebnissen) auch Gliederungen innerhalb von TEI-Dokumenten, etwa per div/head oder entry/form[@type=lemma] zu berücksichtigen.

Schließlich muss die Suche einen sinnvollen Umgang mit Revisionen bieten: D.h. im Standardfalle nur in der neuesten bzw. neuesten publizierten Revision suchen, den Zugriff auf ältere Revisionen jedoch ermöglichen.

Der genaue Aufwand für diese Änderungen ist sehr schwer abzuschätzen. Dies liegt vor allem an der mangelnden Erfahrung mit Solr, wird aber mit Sicherheit mehrere Monate in Anspruch nehmen.

Auch am eXist gibt es Optimierungsmöglichkeiten, die zur Stabilität beitragen könnten. Auch falls in Zukunft Solr zum Einsatz kommt, wären diese sinnvoll einzusetzen für eine mögliche zukünftige XML-Suche. Evaluiert wird per Performanztests gemeinsam mit der FH Worms, ob eXist V2.0 ohne Pairtree besser funktioniert als die Version 1.4. eXist wurde generell von V1.4 auf die jetzt stabile Version 2.0 umgestellt und seit dieser Version verfügt eXist wieder über eine Replikationsfunktionalität, die ebenfalls getestet werden kann.[12]

2.3.3 Abschätzung der Machbarkeit

Aufgrund der Komplexität der Anforderungen und der Menge an Optionen bezüglich eines Ersatzes für eXist ist es dem Konsortium noch nicht möglich gewesen eine verlässliche Aufwandsabschätzung für die Umsetzung von F008 zu machen. Speziell zu diesem Punkt findet am 26. Februar 2013 ein ganztägiger Workshop in Göttingen statt. Die Ergebnisse und Empfehlungen werden in der nächsten Version dieses Berichts veröffentlicht.

[12] Siehe eXist-db Dokumentation. Replication. http://exist-db.org/exist/apps/doc/replication.xml (14. Februar 2013).

3 Anhang

3.1 TextGrid Architekturskizze

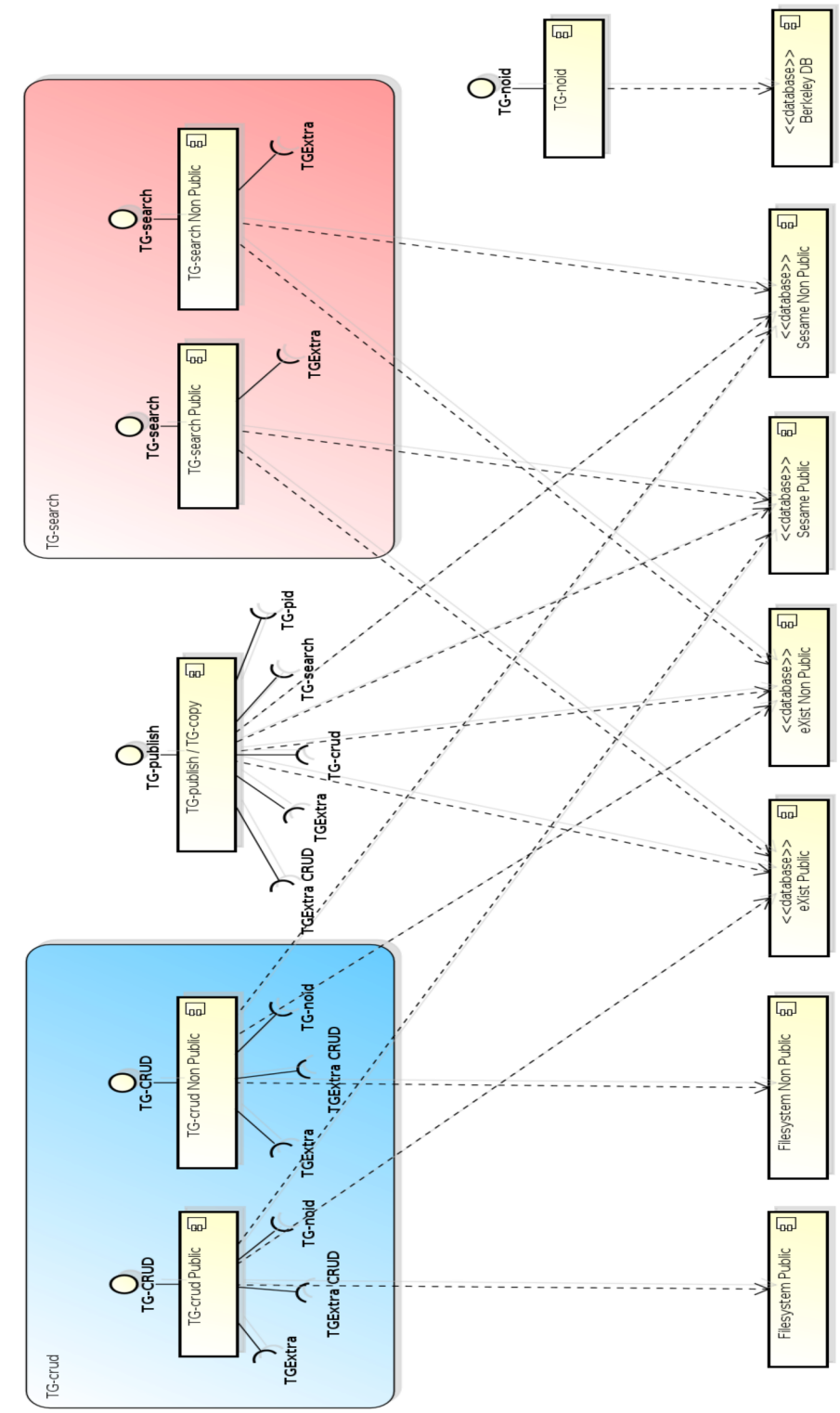
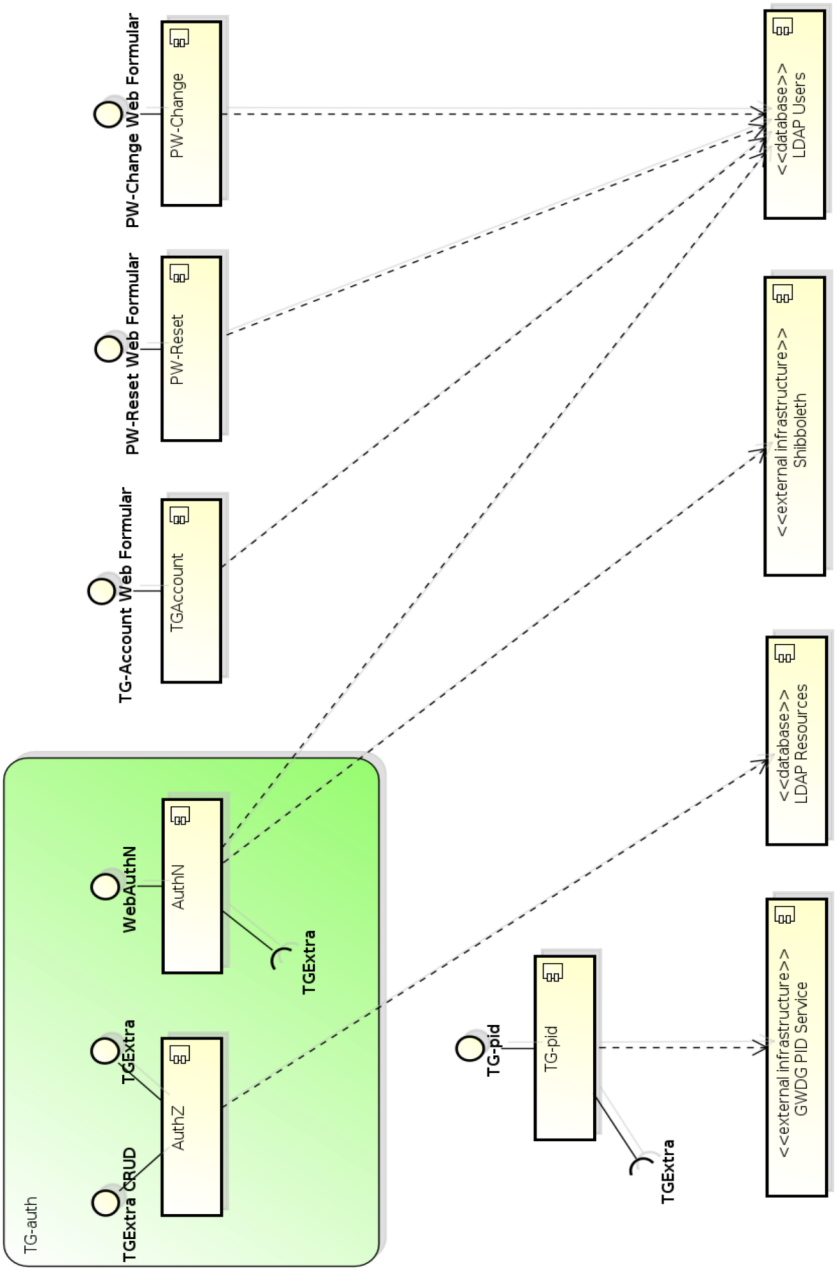


Abbildung 2: TextGrid Architekturskizze, Teil 1



bbildung 3: TextGrid Architekturskizze, Teil 2