Management von Workflow, Access, Kommunikation und Nutzer

Version 2008-01-10/0
Arbeitspaket AP1
verantwortlicher Partner: FH Worms

TextGrid

Modulare Plattform für verteilte und kooperative 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: final

Verfügbarkeit: öffentlich

Autoren:

C. Ludwig, FH Worms

M. Haase, DAASI

Inhaltsverzeichnis

1 Einleitung

In jedem nichttrivialen textwissenschaftlichen Projekt, z. B. einer kritischen Edition, durchlaufen die Ausgangsdaten neben der manuellen Anreicherung und Kommentierung durch den Editor sehr viele automatisierte Transfomationen, ehe daraus die fertige gedruckte oder elektronische Edition wird – erfasste Texte werden in Tokens zerlegt, Varianten eines Textes werden kollationiert, extrahierte Register werden sortiert usw. Diese Arbeitsschritte müssen typischerweise sehr häufig und immer wieder in der gleichen Reihenfolge durchlaufen werden, etwa nach Korrekturen in den Ausgangsdaten oder wenn die Parameter eines der zuvor genutzen Werkzeuge variiert werden sollen. Es bilden sich also festgelegte Arbeitsprozesse heraus, die sich nur in den Daten, auf die sie angewandt werden, und der Konfiguration der beteiligten Werkzeuge unterscheiden. Dieser Report untersucht in Abschnitt 2, welche Unterstützung potentielle TextGrid-Anwender erwarten, um diese repetitiven Arbeiten ergonomisch, nachvollziehbar und weniger fehleranfällig zu machen.

Nicht zuletzt, weil TextGrid angetreten ist, die Kollaboration räumlich entfernter Forscher zu fördern, werden solche Arbeitsabläufe häufig Daten betreffen, die von verschiedenen Personen erstellt wurden. In diesen Fällen ist die Kommunikation aller Mitarbeitenden essentiell, damit sich die Beteiligten nicht gegenseitig behindern oder gar Daten zerstören. Abschnitt 3 betrachtet deshalb, wie der notwendige Informationsfluss in einem in TextGrid umgesetzten Projekt realisiert werden kann.

Im Falle von in TextGrid erstellten Editionen greifen diese Werkzeuge üblicherweise auf Daten zu, die im TextGrid-Datenrepository hinterlegt sind. Weil die Rohdaten die Grundlage für die Arbeit der Fachwissenschaftler darstellen und in ihrer Aufbereitung, Auszeichnung und Kommentierung die intellektuelle Leistung der Editoren steckt, muss der Zugriff auf die Daten zwangsläufig reglementiert werden. Es muss in jedem Einzelfall entschieden werden, wer eine Datei lesen, modifizieren oder gar endgültig löschen darf. Tatsächlich betrifft diese Problematik nicht nur Dateien, sondern beliebige Ressourcen im Grid – wer darf wieviel Speicherplatz, wer darf welche Dienste nutzen etc. Hinzu kommt die rechtliche Notwendigkeit, etwa bei Urheberrechtsverletzungen nachvollziehen zu können, wer welche Daten im TextGrid-Datenrepository hinterlegt hat.

Diese Beispiele zeigen, dass es in TextGrid regelmäßig notwendig ist, zu entscheiden, wer auf eine Ressource zugreifen will und ob ihm dies erlaubt ist. Diese beiden Problemstellungen – *Authentication* und *Authorization* der Nutzer – müssen tatsächlich strikt unterschieden werden; weil sinnvolle Autorisierungsentscheidungen i.d.R. aber eine vorherige Authentifizierung des Anwenders bedingen, werden die eingesetzten Techniken meist unter dem dem Begriff *Authentication and Authorization Infrastructure (AAI)* zusammengefasst.

AAI kann nicht nur unter technischen Gesichtspunkten betrachtet werden, es kommen auch sehr viele organisatorische und ergonomische Aspekte hinzu: Viele der üblichen Sicherheitstechnologien kranken nach wie vor an einer mangelnden Benutzerfreundlichkeit. Und gerade in einer verteilten, „virtuellen“ Organisation wie der TextGrid Community müssen klare, einfache Abläufe definiert sein, wie Anwender ihren „Ausweis“ erhalten und wer ihnen welche Rehte einräumen darf; es bedarf also einer umfassenden Nutzerverwaltung. Wir werden im zweiten Teil dieses Reports die Anforderungen an die Nutzerverwaltung und AAI diskutieren, die sich in einer Grid-Community und insbesondere in TextGrid stellen, sowie einen Überblick über die verfügbaren Lösungen geben.

In beiden Teilen beziehen wir uns ausdrücklich auf bereits existierende Veröffentlichungen des TextGrid-Projekts, in denen Workflows und Sicherheitsarchitektur teils aus fachwissenschaftlicher,

teils aus technischer Sicht behandelt wurden. Hier sind in erster Linie das „Szenarienpapier“ [1] und die AP3-Reports zu Grid Middleware Standards [2] und zur TextGrid **Architektur [3] zu nennen.**

2 Workflow-Unterstützung in TextGrid

Viele der Arbeitsschritte beim Erstellen einer Edition sind inhaltlich vorgegeben – eine Lemmatisierung setzt z. B. voraus, dass der Text zuvor in Tokens zerlegt wird. Selbst dann, wenn ein Anwender die TextGrid-Werkzeuge jedesmal einzeln explizit aufruft, wird er sich also an Arbeitsabläufe halten, die für Dritte mehr oder weniger offensichtlich sind.

Solche ad hoc Workflows sind aber fehleranfällig: Wird ein Zwischenschritt vergessen, so wird das erst durch Fehlermeldungen in nachfolgenden Arbeitsschritten oder gar erst bei der Inspektion des Endergebnisses offensichtlich; in jedem Fall verliert der Anwender Arbeitszeit, weil er den Workflow oder Teile davon wiederholen muss. Noch kritischer ist, wenn der Anwender bei einem Zwischenschritt falsche Parameter gewählt hat. Dies ist im Nachhinein kaum noch nachzuvollziehen. Letzteres ist gerade bei wissenschaftlichen Arbeiten ein Problem: Hier ist es wichtig, dass der Forscher dokumentiert, wie er zu seinen Ergebnissen kam, so dass andere Wissenschaftler seine Methodik und Resultate einer kritisch Begutachten unterziehen können.

Diese Probleme werden durch den Einsatz technisch implementierter Workflows vermieden. Einmal definiert, stellt das System sicher, dass die einzelnen Schritte immer wieder in der gleichen Reihenfolge durchlaufen werden. Statt der Parameter vieler Werkzeuge muss nur noch dieser einzelne Workflow konfiguriert werden, was Fehler vermeiden hilft. Zugleich dient die Workflowdefinition auch der Dokumentation.

2.1 Textwissenschaftliche Anforderungen

Das Szenarienpapier [1] beschreibt typische Vorgehensweise bei verschiedenen, teils überlappenden Typen von textwissenschaftlichen Projekten. Bei einer literaturwissenschaftlichen Edition sind z. B. häufig mehrere Personen mit Texterfassung, manueller oder automatisierterAuszeichnung, Kommentierung, Kollationierung, Vorbereiten von Registern, Satz einer Druckausgabe etc. befasst, die auch noch oftmals parallel arbeiten. Es sollte nicht notwendig sein, dass eine mit dem Satz eines Zwischenstandes der Edition beauftragte Hilfskraft sich in alle für die Edition genutzten Werkzeuge einarbeitet, nur um nach einer Änderung der Auszeichnung durch den Projektleiter die Datenbasis des Satzes aktualisieren zu können. Vielmehr muss der Hilfskraft ein vordefinierter Workflow inkl. Konfiguration zur Verfügung stehen, der all die notwendigen Schritte automatisiert.

Damit dies funktionieren kann, müssen offensichtlich alle Beteiligten von manuellen Korrekturen der Ausgabe eines Werkzeuges im Workflow absehen; andernfalls würde diese Korrektur beim nächsten Durchlauf überschrieben. Im einfachsten Fall kann der Fehler in den Rohdaten korrigiert werden, wenn es sich um ein falsch transkribiertes Zeichen handelt. Hat dagegen ein Tool, das Namen mit Verweisen auf eine einschlägige Normdatenbank anreichert, den falschen Datenbankeintrag ausgewählt, so muss die Konfiguration dieses Werkzeuges angepasst werden, evtl. indem der betreffende Name einer separat geführten Ausnahmeliste hinzugefügt wird. Falls es in Einzelfällen nicht möglich sein sollte, den Fehler durch Parametrisierung der Werkzeuge zu vermeiden, so muss die Änderung als „Patch“ hinterlegt und in einem eigenen Arbeitsschritt, der Teil des Workflows wird,

in die Ausgabe der übrigen Werkzeuge eingearbeitet werden. (In TextGrid würde man in so einem Fall typischerweise auf den Streaming Editor zurückgreifen.)

Daraus ergibt sich, dass textwissenschaftliche Workflows regelmäßig zahlreiche getrennte Dateien zusammenführen: Evtl. über mehrere Dateien verteilten Rohdaten, Ausnahmelisten, Datenbanken, Dienstekonfigurationen und ggf. Patches. Auch die umgekehrte Situation ist häufig, dass ein Zwischenschritt mehrere Ausgabeströme erzeugt; denn bevor Registereinträge sortiert werden, ist es beispielsweise notwendig, dass alle übrigen Daten ausgefiltert werden. Erst wenn die Register zusammengestellt sind, werden sie den Gesamtdaten wieder hinzugefügt.

Insgesamt ergeben sich dadurch mäßig komplexe Workflows. Aus Sicht der Anwender ist besonders wichtig, dass die Workflows intuitiv zusammengestellt und konfiguriert werden können. Die Fachwissenschaftler möchten sich mit ihrem Forschungsgegenstand auseinandersetzen und nicht noch eine weitere Konfigurations- oder Programmiersprache erlernen. Idealerweise können Sie den Workflow über eine graphische Nutzerschnittstelle (GUI) definieren, die visuell vermittelt, welche Daten an welches Tool übergeben werden, wie die Abhängigkeiten zwischen den einzelnen Arbeitsschritten sind und welche Optionen beim Aufruf des Workflows noch explizit von Anwender festgelegt werden müssen. Insbesondere sollte sich die Schnittstelle in TextGrids Eclipse-basierte Client-Anwendung integrieren lassen.

Manche Workflows sind sehr zeitaufwändig, etwa wenn Bedeutungsfelder über mehrere Wörterbücher hinweg miteinander verlinkt werden. Solche Workflows muss das System ausführen können, ohne dass der Anwender ständig online ist; der *Workflow Enactor* muss folglich als eigenständiger Dienst in TextGrid zur Verfügung stehen. Speziell während der Spezifikation der Workflows muss der Anwender aber auch deren Ablauf im GUI überwachen können, so dass sich im Falle von Fehlern das Problem auch ohne mühsames Studium von Logdateien diagnostizieren lässt.

2.2 Existierende Software zum Workflow-Management

Im Folgenden werden einige Systeme für Workflow-Management in Grid-Umgebungen vorgestellt.

- **Intalio|BPMS** (http://bpms.intalio.com): Die Software ist zwar quelloffen, aber mit erheblichen Beschränkungen lizensiert. In der Architektur eine strikte Trennung zwischen Editor ("BPMN Designer") und Enactor ("BPEL Server"). Im Designer wird die eher intuitive BPMN-Notation verwendet, mit der eine Workflow-Spezifikation nach WS-BPEL 2.0 exportiert werden kann. Die BPEL-Datei wiederum kann im Server abgesetzt werden. Letzterer läuft unter einem J2EE Server (Apache Geronimo) und hat eine eigene Web-Console zum Administrieren der übergebenen Prozesse. Weitere Merkmale:
  - Information Retrieval: UDDI und MDS4 werden nicht unterstützt
  - Einbindung: Designer ist Eclipse-Plugin
  - Web Services: können gestartet werden da WS-BPEL; allerdings keine RPC-Style
- **OMII-BPEL** (http://sse.cs.ucl.ac.uk/projects/omii_bpel/): OMII-BPEL ab Version 2.0 basiert auf dem *Eclipse BPEL Designer*, der als Open-Source-Projekt u. a. von IBM, Oracle und eben OMII mitentwickelt wird. Im Installationspaket wird dieser Editor zusammen mit der ebenfalls Open Source ActiveBPEL Engine ausgeliefert. Beide Teile sind von OMII gegenüber den Originalprojekten (http://www.eclipse.org/bpel/ bzw. http://www.activebpel.org/) für Grid-Anwendungen speziell modifiziert worden.

Voraussetzung für den Betrieb der so modifizierten Pakete ist eine Installation des OMII-Programms, d.h. für den Editor muss der OMII-Client (Größe des gepackten Archivs ca. 200MB) und für die Engine der OMII-Server (400MB) installiert sein. WS-BPEL 2.0 wird weitgehend unterstützt. ActiveBPEL hat ein Web-Frontend zur Administration der eingesetzten Prozesse. Hauptnachteil ist die Bedienbarkeit von OMII-BPEL: die graphische Oberfäche nimmt nichts von der Komplexität von BPEL und ist für einen nicht-BPEL-Experten kaum intuitiv. Da das BPEL-Designer-Projekt von Eclipse noch keinen stabilen Status erreicht hat, fehlt noch die eine oder andere Funktionalität. Insbesondere ist das automatische Erstellen des BPR-Archivs (was ActiveBPEL erwartet) und das Deployment an die Engine noch nicht hinreichend unterstützt. Weitere Merkmale:

  - Information Retrieval: Es gibt einen noch nicht funktionalen UDDI-Browser
  - Einbindung: der Editor ist Eclipse-Plugin; zur Administration der Engine muss auf ein Web-Interface zurückgegriffen werden.
  - Web Services: können gestartet werden, da BPEL

- **Triana** (http://www.trianacode.org) Triana ist ein integriertes Workflow-Tool, d.h. Editor und Enactor sind eng verzahnt, die Ausführung und Überwachung des Workflows kann vom Editor aus gesteuert werden. Die Bedienung des Editors ist sehr intuitiv per Drag&Drop , Komponenten werden verbunden durch (Datenfluss-)Kabel. GT(4)-Unterstützung durch JavaGAT. Inkompatibilität von Datentypen wird noch bei der Komposition des Workflows abgefangen.
  - Open Source: ja
  - Einbindung: Eclipse-Plugin fraglich (zu groß und zu viele Abhängikeiten von diversen Bibliotheken)
  - Format der Workflow-Spezifikation: proprietäres XML (Triana "TaskGraph")
  - Web Services: können ausgeführt werden, sowie GT4 WSRF Services
  - Information Retrieval: UDDI, kein MDS
- **Taverna** (http://taverna.sourceforge.net) Taverna ist ein integriertes Workflow-Tool. Die graphische Komposition eines Workflows ist nicht möglich, dafür aber Menü-basierte Zusammenstellung mit sofortiger graphischer Darstellung, also etwas weniger intuitiv. Der Workflow kann direkt in Taverna ausgeführt und überwacht werden. Im Hintergrund läuft die Freefluo Engine (Enactor), was dem normalen Anwender aber verborgen bleibt. Der Enactor kann alternativ auch auf einem speziellen Rechner als Service laufen. Fault Handling kann explizit spezifiziert werden. Globus Toolkit Jobs werden nicht unterstützt
  - Open Source: ja
  - Einbindung: Eclipse-Plugin fraglich (zu groß und zu viele Abhängikeiten von diversen Bibliotheken)
  - Format der Workflow-Spezifikation: proprietäres DAG-basiertes XML (SCUFL)
  - Web Services: können ausgeführt werden
  - Information Retrieval: WSDL, sowie Bioinformatik-spezifische Protokolle (biomart, seqhound, biomoby, soaplab)
- **Karajan bzw. cog-workflow-gui** (http://wiki.cogkit.org) Karajan ist ein integriertes Workflow-Tool. Eine Graphische Komposition des Workflows ist nicht möglich (die Autoren bevorzugen das direkte Editieren von XML-Code), wohl aber Visualisierung, Starten und

Überwachung eines Workflows in der GUI. Als Teil des Java CoG Kits wird Globus unterstützt.

- Open Source: ja
- Einbindung: Eclipse-Plugin wahrscheinlich realisierbar, das Tool ist relativ klein und kompakt.
- Format der Workflow-Spezifikation: proprietäres textbasiertes und XML-Format
- Web Services: benutzt CoG Kit Abstraction Classes für Services für Job Submission und File Operations, d.h. der Service hängt vom *Provider* ab; das Abstraction Interface JobSpecification geht von Executables und nicht Web Services aus
- Information Retrieval: Abstraction Interface "InformationSpecification" für die Abfrage von Informationen ist noch nicht implementiert.

- **GridAnt** (http://wiki.cogkit.org) GridAnt ist ein nicht-graphisches Tool, das Apache Ant erweitert und Vorgänger von Karajan.
  - Open Source: ja
  - Einbindung: vgl. Karajan
  - Format der Workflow-Spezifikation: vgl. Karajan; weniger mächtig in Bezug auf Programmablaufkontrollstrukturen
  - Web Services: vgl. Karajan
  - Information Retrieval: vgl. Karajan
- **Virtual Data System** (http://vds.uchicago.edu/twiki/bin/view/VDSWeb/WebMain) VDS ist das Workflow-System in GriPhyN (hervorgegangen aus "Chimera" und beinhaltet Pegasus) für Daten-intensive Grids. VDS benötigt neben Globus2/3/4 auch Condor. Weder graphischer Editor noch Visualisierungs-Tool stehen zur Verfügung. Als Workflow Engine kommt Condor DAGman zum Einsatz.
  - Open Source: ja
  - Einbindung: vermutlich schwierig, System relativ groß, viele Bibliotheken
  - Format der Workflow-Spezifikation: DAG in textuellem bzw. XML-Format
  - Web Services: Jobs werden über GRAM gestartet (also Executables, keine Web Services)
  - Information Retrieval: Abstrakte Workflows werden in konkrete (d.h. voll in Bezug auf bestimmte Ressourcen und physikalische Dateien spezifizierte) Workflows mit Information aus dem *Site Catalog* einer jeden Ressource umgewandelt. Der Site Catalog kann u.A. Information aus MDS4 beinhalten.
- **Kepler** (http://www.kepler-project.org) Kepler ist ein integriertes Tool, d.h. vom Workflow-Editor wird auch die Ausführung des Workflows gesteuert. Ein Workflow ist aus Komponenten ("Actors") aufgebaut, verbunden durch Data Links. Verschiedene Scheduler ("Directors") sind verfügbar.
  - Open Source: ja
  - Einbindung: vermutlich schwierig, System relativ groß, viele Bibliotheken
  - Format der Workflow-Spezifikation: proprietäres XML-Format (MoML)
  - Web Services: entsprechende Actors können über ihre WSDL eingebunden werden; es gibt auch Actors für Grid Services und Globus Grid Jobs (GRAM)

    - Information Retrieval: *Web Service Harvester* findet verfügbare WS in einem Repository (z.B. UDDI), über Storage Resource Broker kann auf Metadaten zugegriffen werden (sowie SRB Dateioperationen).
- **g-Eclipse** (http://www.geclipse.org) Das Eclipse-basierte Projekt hat auch eine Workflow-Komponente, die momentan (Nov 07) aber noch sehr in Entwicklung befindlich ist. Diese ist, wie auch sonst durchgängig in g-Eclipse, zunächst abstrakt und besteht aus einem GMF-basierten Editor mit Jobs, In- und Output Ports und Links, der eine Workflowbeschreibung im XMI-Format erstellt. Die Abbildung des Workflows auf eine konkrete Grid-Umgebung geschieht dann über entsprechende Konverter. Im Augenblick sind diese für zunächst gLite geplant, später GRIA. (In gLite gibt es ein Grid-weites Workload Management System, WMS, an das DAG-basierte in JDL beschriebene Jobs übermittelt werden können; einzelne Workflow-Engines a la Taverna sind nicht vorgesehen.)
    - Open Source: ja
    - Einbindung: Editor läuft unter Eclipse; Programmierung der Konversion in konkrete Beschreibungssprache (z.B. in Taverna's XSCUFL) muss noch vorgenommen werden, mitsamt der Auswahl einer konkreten Engine
    - Web Services: abhängig von der gewählten Beschreibungssprache und Engine
    - Information Retrieval: noch nicht vorgesehen

3 Kommunikation kollaborierender Nutzer

Trotz aller automatisierten Workflows ist eine funktionierende Kommunikation Grundvoraussetzung in jedem Projekt, an dem mehrere TextGrid-Nutzer beteiligt sind. Dies beginnt bei der Frage, wie die innerhalb eines Projektes einzuhaltenden Konventionen aussehen, geht über fachwissenschaftliche Diskussionen zum Forschungsgegegenstand bis hin zu der Entscheidung, wann welche Daten in welcher Form veröffentlicht werden.

In der Anfangsphase eines Projektes sind vor allem zahlreiche Entscheidungen bzgl. der im Projekt verwendeten Richtlinien notwendig: Welche Quellen werden in die Arbeit einbezogen? Welche Textphänomene sollen erfasst werden? Gemäß welchem Schema soll die Auszeichnung erfolgen? Weil Änderungen bei diesen Punkten später im Projekt meist nur mit sehr großem Aufwand umzusetzen sind, müssen die Projektverantwortlichen diese Richtlinien vorab diskutieren. Im Rahmen dieser Diskussion kann es sinnvoll sein, kleinere Testläufe mit den TextGrid-Werkzeugen durchzuführen und allen Beteiligten Zugriff auf die Testdaten und Resultate zu geben; doch darüber hinaus kann dieser Prozess nicht wesentlich mit spezifischen Werkzeugen unterstützt werden. Diese Entscheidungsfindung wird typischerweise in persönlichen Treffen oder über die „klassischen“ Telekommunikationswege Telefon und Email erfolgen.

Das gleiche gilt auch für die inhaltlichen Entscheidungen während der Arbeitsphase eines Projektes. Sofern sie gemeinsam getroffen werden, so geht ihnen i.d.R. eine fachwissenschaftliche Diskussion voraus. Für diese sind Email und Telefon das Medium der Wahl.

Ein wenig anders ist die Situation bzgl. organisatorischer Fragen. Beispielsweise muss vermieden werden, dass mehrere Anwender gleichzeitig an einer Datei arbeiten und sich möglicherweise gegenseitig die Änderungen überschreiben – auch dann, wenn es sich bei den Anwendern um Hilfskräfte an verschiedenen Standorten handelt, die vielleicht gar nicht voneinander wissen.

Idealerweise wird ein solches Szenario von vornherein verhindert, indem jeder Anwender einen klar definierten Zuständigkeitsbereich hat und nur auf „eigenen“ Dateien arbeitet. Wenn jeder Mitarbeiter ausschließlich auf eigenen Dateien Schreibrechte hat, besteht keine Gefahr, dass sich die Beteiligten gegenseitig Daten zerstören. Bei Bedarf können die Dateien der einzelnen Projektangehörigen durch einen geeigneten Workflow zusammengeführt werden.

Dies wird sich nicht immer für alle Dateien in einem Projekt durchhalten lassen. Speziell Workflow-Konfigurationen u.ä. werden häufig von mehreren Mitarbeitern in einem Team angepasst werden. In diesem Fall sind individuelle Absprachen untereinander und instantane Kommunikationsmittel (Telefon, Chat) notwendig. Es wäre an dieser Stelle hilfreich, wenn ein Anwender durch ein Lock auf eine Datei (oder noch besser einen Dateiausschnitt) signalisieren könnte, dass er beabsichtigt, hier demnächst Änderungen vorzunehmen. Dies wird derzeit von TextGrids File-Schnittstelle nicht unterstützt; erst wenn Erfahrungsberichte aus den ersten Anwenderprojekten vorliegen, wird es möglich sein, zu entscheiden, ob der Bedarf an dieser Stelle groß genug ist, um den Aufwand zu rechtfertigen, die File-Schnittstelle um ein derartiges Feature zu erweitern.

Abgesehen von solchen Hilfsmitteln, die ggf. eine Kommunikation erzwingen, ehe der Schreibzugriff auf eine Ressource erlaubt wird, sind keine dedizierten Kommunikationswerkzeuge in TextGrid selbst vorgesehen. Wir gehen davon aus, dass die anderweitig verfügbaren Kommunikationsmittel hinreichend sind.

4 Nutzerverwaltung

Wenn ein Wissenschaftler seine Daten online veröffentlicht, dann ist typischerweise das Ziel, dass jeder Interessierte die betreffenden Dateien z. B. mit dem Webbrowser herunterladen kann. Der Nutzerkreis soll nicht auf Personen beschränkt werden, die in irgendeiner Weise mit einem Projekt oder einer Einrichtung affiliiert sind.

Vor der Veröffentlichung der Daten wünschen die Forscher aber sehr wohl eine genaue Kontrolle, wer Zugriff darauf bekommt. Wenn die Daten, wie im Fall von TextGrid, in einem gemeinsamen Repository liegen, dann muss das System folglich Benutzer verwalten, denen es dann jeweils Zugriffsrechte zuordnen kann.

4.1 Aufgaben und Randbedingungen der Nutzerverwaltung

Aus Sicht eines Computersystems ist ein Nutzer immer nur eine Nutzer-ID. Bei dieser ID kann es sich um eine Zahl, einen lokalen Accountnamen, einen *Distinguished Name* nach RFC 4514 oder anderes handeln. Auch wenn im Einzelfall der Name einer Person, die das Computersystem nutzt, unmittelbar aus der Nutzer-ID hervorgehen kann, handelt es sich dabei im Allgemeinen um ein Pseudonym. Während jede Person nur eine Identität hat, kann sie einem System wie TextGrid gegenüber mit mehreren Pseudonymen auftreten. Diese sind nicht zu verwechseln mit den Funktionen oder *Rollen*, die ein Nutzer in einem System übernehmen kann. Jedem Nutzer (genauer: jeder Nutzer-ID) können beliebig viele Rollen zugewiesen werden; umgekehrt können sich aber auch mehrere Nutzer eine Rolle teilen.

Die Nutzerverwaltung entscheidet, welche Nutzer-IDs vom System akzeptiert werden, wie sich die Nutzer gegenüber dem System authentisieren können und welche Rollen ein Nutzer übernehmen kann. Bei der *Nutzerregistrierung* wird ein Pseudonym mit der Identität eines Anwenders verknüpft. Im

einfachsten Fall sind alle zulässigen Nutzer-IDs in einer Passwortdatenbank hinterlegt und die Authentifikation der Anwender geschieht durch die Eingabe des Passworts. Die Registrierung erfolgt, indem der Anwender, für den die Nutzer-ID angelegt wurde, von einem Administrator persönlich das initiale Passwort ausgehändigt bekommt. Allerdings ist diese Methode bei einem verteilten, institutionenübergreifenden System wie TextGrid nicht mehr handhabbar. Hier werden Mechanismen benötigt, die es erlauben, die Nutzerregistrierung zu delegieren – üblicherweise an die lokalen Institutionen der Anwender, die sich im Gegenzug dazu verpflichten müssen, bei der Nutzerregistrierung die Reglements der jeweiligen *Virtuellen Organisation*, im konkreten Fall der TextGrid-VO, zu beachten.

Außerhalb des Grid-Kontextes ist eine Vielzahl von Systemen zur Nutzerregistrierung im Einsatz. Nach der bereits oben genannten Registrierung durch den Admin, der auch die Nutzer-ID im System anlegt, ist die Verifikation eines aktiven Mailaccounts das nächsteinfachste (aber auch das unsicherste) Verfahren. Weil hierbei die Nutzer-ID nicht direkt an die Identität der Person, sondern nur an den Zugriff auf ein Email-Postfach geknüpft wird und aus letzterem nicht zwangsläufig auf eine Person geschlossen werden kann, ist dieses Verfahren für die Nutzerregistrierung in TextGrid ungeeignet.

Besorgt sich ein Anwender ein X.509 „Grid-Zertifikat“, so wird seine Identität durch eine von der zertifikatsausgebenden Stelle beauftragte *Registration Authority* geprüft. Indem die CA anschließend das Zertifikat des Anwenders signiert, wird der darin enthaltene Distinguished Name (DN) mit dem darin enthaltenen privaten Schlüssel verknüpft, auf den ausschließlich der Anwender Zugriff haben darf. Hierbei ist zu beachten, dass damit noch keine Registrierung bei einer Grid-VO verbunden ist. Es sind lediglich die Voraussetzungen geschaffen, dass eine VO den entsprechenden DN später ohne weitere Identitätsprüfung registrieren kann, weil das Zertifikat auch für eine Public Key Authentifizierung genutzt werden kann. Es bleibt zunächst offen, ob die VO bereit ist, jeden Zertifikatsinhaber zu registrieren, oder ob sie dies an Bedingungen knüpft wie etwa die Einwilligung bestimmter Personen.

In wieder anderen Szenarien erfolgt die Authentifizierung nicht über eine Public Key Infrastructure ohne Einschaltung von Dritten, sondern bei Bedarf unter Zuhilfenahme eines *Identity* Providers (vgl. dazu auch die Ausführungen zu Shibboleth im folgenden Abschnitt). Dieser kann, muss aber nicht von der Heimatinstitution des Nutzers bereitgestellt werden.

Welcher Aufwand für eine Registrierung bei der TextGrid-VO akzeptabel ist, hängt u.a. vom jeweiligen Nutzungsszenario ab. Wer ein großes, mehrjähriges Editionsprojekt mit einer relativ stabilen Gruppe von Mitarbeitern plant, kann die Umstände in Kauf nehmen, die damit verbunden sind, für die Mitarbeitenden Grid-Zertifikate zu besorgen, auch falls an der jeweiligen Einrichtung noch keine Registration Authority existiert. Wer andererseits TextGrid in der Lehre einsetzen will, wird diesen Aufwand für möglicherweise Dutzende oder gar hunderte von Studierenden im Jahr scheuen, weil die Anwender dann auch noch in der Nutzung dieser Zertifikate geschult werden müssen.

In D-Grid wird die Authentifizierung von Benutzern offiziell bislang ausschließlich über Nutzerzertifikate unterstützt. Als Top-Level CAs fungieren hier zentral für Deutschland das Forschungszentrum Karlsruhe und der DFN-Verein. Allerdings wird die Notwendigkeit von dezentralen Authentifikationsmechanismen gesehen, wie etwa Shibboleth, das derzeit verstärkt an deutschen Hochschulen eingeführt wird.

4.2 Verfügbare Systeme zur Nutzerverwaltung

Bei den Systemen zur Benutzerverwaltung ist grundsätzlich in eher zentrale und eher föderale Ansätze zu unterscheiden. Systeme für beide Ansätze werden im Folgenden kurz beschrieben.

4.2.1 Zentrale Ansätze

- **Virtual Organization Membership Service (VOMS)**: Verwalten von Benutzern innerhalb einer Virtuellen Organisation. Für das D-Grid gibt es einen zentralen VOMS-Server beim Forschungszentrum Karlsruhe. Es werden Eigenschaften des Benutzers als Attribute in einer Datenbank gespeichert. Übermittelt werden sie in Attribut-Zertifikaten, die wiederum als Erweiterungen in Proxy-Zertifikate eingebettet werden. Als Attribute können enthalten sein: E-Mailadresse, Common Name, sowie weitere Gruppen- bzw. Rollenattribute, die für die Autorisierung auf D-Grid-Ressourcen verwendet werden können. Per Web-Frontend können Nutzer sich registrieren und Administratoren diese verwalten. Die Daten können dann per Web Service abgefragt werden.
- **Virtual Organization Membership Registration Service (VOMRS):** Der VOMRS-Server für das D-Grid ist beim Forschungszentrum Jülich installiert. VOMRS hat einige Gemeinsamkeiten mit VOMS, aber auch eigene Merkmale, weshalb sich diese beiden Systeme gut ergänzen, auch wenn sie im Prinzip auch getrennt eingesetzt werden können. Gemeinsam mit VOMS hat VOMRS: Das Datenbank-Backend, die Speicherung von Attributen des Benutzers, sowie ein Web-Frontend für Benutzer und Administratoren. Alleinstellungsmerkmale im Gegensatz zu VOMS sind: einfachere Adminstration, Verwendung von Acceptable Usage Policies (AUPs), Speicherung beliebiger Attribut-Wert-Paare. Die eingegebenen Daten können dann an einen VOMS-Server weitergeleitet werden.

4.2.2 Föderale Ansätze

- **Shibboleth** bietet – basierend auf der Security Assertion Markup Language (SAML) – eine föderierte Authentifizierungs- und Autorisierungs-Infrastruktur an. Es wird *Single Sign On* (SSO) über Organistationsgrenzen hinweg realisiert. Die Funktionsweise von Shibboleth wird von drei Komponenten bestimmt: *Identity Provider* (IdPs), *Service Provider* (SPs) und einen Lokalisierungsdienst (*Where-Are-You-From*, WAYF). IdPs gibt es pro Organisation jeweils nur einmal. Sie sind an die Benutzerverwaltung der jeweiligen Organisation angeschlossen und geben Attribute von dort über Nutzer weiter. SPs verwalten Web-basierte Ressourcen. Den WAYF gibt es für die ganze Föderation nur einmal. Hier kann der Benutzer auswählen, an welchen IdP er zur Authentifizierung geleitet wird.

  Hier der grobe Workflow einer Shibboleth-Sitzung: Ein nicht authentifizierter Nutzer will mit seinem Browser auf eine Ressource zugreifen. Der dazugehörige SP leitet die Anfrage zum WAYF um, wo der Nutzer die Organisation auswählt, bei der er einen Account hat. Der WAYF leitet den Browser nun zum IdP dieser Organisation weiter, wo er seinen Benutzernamen und Passwort eingibt. Nach Authentifizierung schickt der IdP an den SP, auf den der Benutzer zugreifen wollte, ein Handle, über das der SP noch weitere Attribute vom IdP erfragen kann. Ist dieser (optionale) Austausch von Informationen erfolgt, gibt der SP die Ressource frei.

Zu beachten ist, dass bei Shibboleth keine zentrale Nutzerverwaltung existiert, sondern diese jeweils bei den einzelnen Organisationen, die an der Föderation teilnehmen, verbleibt.

Problematisch bei Shibboleth-basierten Systemen ist die Verwaltung der VO-Attribute. Die Verwaltung der Campus-Attribute, also der originär Hochschul- bzw. Organisations-spezifischen Attribute wird von der jeweiligen Hochschule betrieben. Im Gegensatz dazu stehen aber die VO-Attribute, also Grid-spezifische Attribute, die ebenfalls irgendwo hinterlegt sein müssen. Diese Verwaltung kann schwerlich die Hochschule für jede VO oder Grid-Community durchführen, da hier gänzlich andere Anforderungen an die Attribute gestellt werden.

- Als Lösung des Problems der VO-Attribute ist am ehesten das Konzept des IdP Proxy anzuführen, das z.B. in **myVocs** verfolgt wird. myVocs tritt gegenüber den Campus IdPs als SP auf und erfragt dort die Campus-Attribute. Gegenüber den Grid SPs fungiert es als IdP, der VO-Attribute vorhält, pflegt und an die SPs ausgibt. Via Account Linking werden die Campus- und VO-Attribute dann verbunden und an die SPs übergeben. myVocs fungiert also als Brücke zwischen einem Verbund von Heimat-IdPs und VO SPs.
- Das **GridShib**-Projekt (GS) ist der Versuch, die Shibboleth-Technologie auch für Grid-Umgebungen (momentan Globus) zu nutzen. Die hier entwickelte Software besteht aus verschiedenen Teilen: *GS for Globus*: implementiert einen *Policy Decision Point* (PDP), der Autorisierungsentscheidungen (ja/nein) aufgrund von Shibboleth-Attributen macht. *GS for Shibboleth* ist ein Shibboleth-Plugin, das die Abbildung von Zertifikats-DNs auf lokale Benutzernamen ermöglicht. Die *GS CA* ist vor allem für neue oder gelegentliche Grid-Nutzer gedacht, die kein eigenes X.509-Zertifikat besitzen. Der Benutzer kann dadurch kurzlebige Zertifikate nach der Authentifizierung bei seinem Heimat-IdP über Shibboleth erzeugen. Für den Programmierer stehen noch die *GS SAML Tools* zur Verfügung, die für die Verarbeitung von SAML-Assertions eingesetzt werden können.

5 Rechteverwaltung und Zugriffskontrolle

Die Nutzerverwaltung entscheidet lediglich, welche Nutzer-IDs akzeptiert werden und wie die Authentisierung der Anwender erfolgt. Im Regelbetrieb muss das System aber regelmäßig anhand der Nutzer-ID, der vom Anwender (aktuell) wahrgenommenen Rollen, den Attributen der angeforderten Ressourcen und möglicherweise weiterer Informationen auch Autorisierungsentscheidungen treffen. Dies ist die Domäne der Systeme zur Rechteverwaltung und Zugriffskontrolle. Wir betrachten im Folgenden die Anforderungen an Autorisierungsentscheidungen und mit welchen Mechanismen diese getroffen werden können. Wie solche Entscheidungen vom Gesamtsystem durchgesetzt werden können, ist eine Architekturfrage, die z. B. im TextGrid Architektur Report [3] diskutiert wird..

5.1 Rechtliche, technische und fachwissenschaftliche Anforderungen

Auch wenn vielleicht die Mehrzahl der potentiellen TextGrid-Nutzer in erster Linie wissenschaftliche und keine kommerziellen Ziele verfolgen, kann eine Plattform wie TextGrid nicht in einem rechtsfreien Raum operieren. So gibt es beispielsweise für Anwender kein technisches Hindernis, Daten unter Verletzung des Urheber- bzw. Verwertungsrechtes Dritter in TextGrids Datenrepository zu speichern und womöglich zu publizieren. Damit die Betreiber der Infrastruktur nicht ein

unkalkulierbares Risiko eingehen, für solche Verletzungen mitverantwortlich gemacht zu werden, benötigen sie die Möglichkeit, die betreffenden Ressourcen zunächst für jeden Zugriff zu sperren und ggf. auch zu löschen, sobald sie von einer Urheberrechtsverletzung Kenntnis erlangen. Ausgewählte „TextGrid-Administratoren“ müssen sich also jederzeit über die für die übrigen Anwender geltenden Zugriffsbeschränkungen hinwegsetzen können.

Zugleich müssen natürlich auch die Anwender sicher sein, dass Ihnen bzw. Ihren Projekten niemand Daten unterschieben kann, derentwegen sie später vielleicht in juristische Auseinandersetzungen verwickelt werden. Sie benötigen deshalb die Möglichkeit, im Voraus zu bestimmen, wer „ihrem“ Projekt zugeordnete Daten schreiben und womöglich im Namen des Projektes veröffentlichen darf. Außerdem wollen die Forscher selbstverständlich auch ihr eigenes Urheberrecht an den von ihnen selbst erstellten Daten durchsetzen können und z. B. Lesezugriffe durch nicht ausdrücklich Befugte unterbinden.

Nach der Authentifizierung, also der Feststellung der Identität des Benutzers, muss eine Autorisierung erfolgen, also letztlich eine Ja/Nein-Antwort auf die Frage: „darf der Benutzer auf Ressource X zugreifen?“ Eine Ressource kann dabei eine Datei, ein Dienst, ein zugangsbeschränkter Raum oder Ähnliches sein. Die Entscheidung, ob der Benutzer mit seiner ID auf die Ressource zugreifen kann, kann auf vielerlei Informationsquellen basieren: expliziten Faktoren wie der ID selbst, der Eintragung der ID in eine spezielle Gruppe/Rolle, der Operation, die auf der Ressource ausgeführt werden soll;der Inhalt oder Eigenschaften der Ressource selbst können herangezogen werden; oder äußere Faktoren können eine Rolle spielen, z.B. die Uhrzeit des Zugriffs oder die Abhängigkeit von einer weiteren externen Informationsquelle, z.B. einer Alarmanlage. Alle diese Faktoren und mögliche Kombinationen davon oder Vergleichsoperationen auf ihnen können implizit im System ausgedrückt sein; bei einer expliziten Formulierung der Zugriffsregeln spricht man dagegen üblicherweise von einer Richtlinie oder *Policy*.

Eine Policy kann separat formuliert werden, z.B. spezifiziert der Standard XACML solche Sprachelemente. Ferner kann es abstrakte Policies geben, die Klassen von Ressourcen beschreiben und die dann auf verschiedene konkrete Umgebungen angewandt werden können, bzw. die Instanzen von konkreten Policies bilden können.

In TextGrid sind die Objekte, die die Ressourcen bilden, im Regelfall Dateien in verschiedenen Formaten, aber theoretisch auch – wie in anderen Communities – Services. Diese können einerseits Benutzern, andererseits bestimmten Gruppen (bzw. Rollen) oder Projekten zugeordnet sein. Es gibt publizierte Ressourcen, bei denen keinerlei Beschränkung von Lesezugriffen vorliegt.

Einige Szenarien, wie sie in TextGrid vorkommen werden und die durch die Zugriffskontrolle abgedeckt werden sollen, sind:

- Professor X will neues Projekt Y anlegen mit sich als Projektleiter.
- Projektleiter in Projekt Y will Bearbeiter in das Projekt aufnehmen.
- Projektleiter in Projekt Y möchte Administration an einen Mitarbeiter delegieren
- Einige Dokumente aus Projekt Y1 sollen dem Projektleiter oder anderen Personen oder Rollen aus Projekt Y2 zum Lesen zur Verfügung gestellt werden.

- Einige Dokumente aus Projekt Y1 sollen für die Allgemeinheit oder TextGrid-weit publiziert werden
- TextGrid-Nutzer möchte über ein Projekt / alle seine Projekte / TextGrid-weit auf den Metadaten oder im Volltext suchen
- Bearbeiter in Projekt Y will eine Ressource lesen/modifizieren/löschen
- Projektleiter/Bearbeiter in Projekt Y will eine Ressource in Projekt Y einbringen
- Projektleiter/Bearbeiter in Projekt Y will Ressource Z für alle Bearbeiter in Projekt Y freischalten (lesen, evtl. schreiben)

5.2 Systeme zur Autorisierung von Zugriffen

- Ein einfaches, aber relativ mächtiges und schon sehr ausgereiftes „Autorisierungssystem“ ist das Dateisystem unter **Unix**. Hier werden alle Dateien bzw. Verzeichnisse einem Nutzer und einer Gruppe zugeordnet. Entsprechend kann der Zugriff auf die Datei auf den Nutzer, die Gruppe oder aber die Allgemeinheit beschränkt werden, jeweils mit den zugelassenen Operationen *lesen*, *schreiben* und/oder *ausführen*. Ein Nutzer kann in beliebig vielen Gruppen Mitglied sein.

  Es gibt Unix-Erweiterungen, bei denen eine Datei zu mehreren Gruppen/Benutzern zugeordnet werden kann, aber im Basis-Unix „gehört“ eine Datei genau einem Nutzer und einer Gruppe, was eine offensichtliche Beschränkung darstellt. Diese Art der Autorisierung hat außerdem die Beschränkung, dass sie nur in einem lokalen Netz von Unix-Rechnern gilt und nicht über die Grenzen dieses Netzes hinweg oder gar im Internet.

- Im Projekt **PERMIS** (PrivilEge and Role Management Infrastructure Standards validation)[1] der Universität Kent wurde ein mächtiges System entwickelt, dessen in unterschiedlichen Einzelprojekten entstandenen Module verschiedene Aspekte von Autorisierung in verteilten Umgebungen abdecken. Der Kern der PERMIS-Palette ist der *Credential Validation Service* (CVS) und die *Policy Engine* (PDP), wobei der CVS Attribute von Nutzern überprüft und an den PDP weiterreicht, welcher dann die eigentliche Autorisierungentscheidung basierend auf der Policy und den Attributen, aber auch weiteren Kontextvariablen, durchführt. Weitere eigenständige Komponenten sind der *Attribute Certificate Manager*, mit dem Benutzer bzw Administratoren Rechte (=Attribute) beschreiben und delegieren können, und der *Policy Editor*, mit dem Autorisierungsrichtlinien beschrieben und in einem XML-Format (und dann optional in einem Attributzertifikat gekapselt) gespeichert werden können. Hinzu kommt noch die Integration der PERMIS-Software in verschiedene Umgebungen, so Apache Webserver, RBAC, Shibboleth, GT4, GridShib, LDAP,

  PERMIS, so wie eben beschrieben, ist äußerst komplex und teilweise auch inkonsistent mit neueren Versionen der Software-Umgebungen, auf die es aufbaut. Der Code verwendet eine

[1] http://www.permis.org/en/index.html

Bibliothek mit einer für nicht-Forschungszwecke problematischen Lizenz. Die Software ist nach unserem Wissen nicht produktiv im Einsatz und eher ein Forschungsbaukasten, weshalb von einem konkreten Einsatz wohl eher abzuraten ist.

- Eine weniger komplexe, aber dennoch mächtige Alternative zu PERMIS ist eine **RBAC**-Implementierung, die gegenwärtig am Wilhelm-Schickard-Institut für Informatik der Universität Tübingen entsteht. RBAC (Role-Based Access Control) [4] ist eine Autorisierungslösung, die auf Rollen – im Gegensatz zu Gruppen, wie etwa unter Unix – basiert. Die Rollen sollen – vor allem in größeren Einrichtungen – entlang der Organisationsstruktur angelegt werden, was zu einer natürlicheren Einteilung führt als bei Gruppen, die normalerweise nur zum Zwecke der Autorisierung angelegt werden. Ein Benutzer kann beliebige Rollen einnehmen, wobei spezifiziert werden kann, welche und wieviel davon gleichzeitig aktiv sein dürfen. Zu jeder Ressource kann flexibel aufgelistet werden, für welche Rollen mit welchen Operationen Zugriff besteht.

  Die rein auf Open-Source-Tools basierende Implementierung besteht aus einer Suite von PHP-Web- Services mit einer LDAP-Datenbank zur Speicherung der Berechtigungen. Der Quellcode selbst ist ebenfalls Open Source.

6 Zusammenfassung

Wir haben in diesem Bericht einen Überblick über die Anforderungen an und Systeme zur Workflow-Unterstützung, Nutzerverwaltung, Rechteverwaltung sowie Zugriffskontrolle gegeben, die für TextGrid betrachtet wurden. BeimWorkflow-Management wurden mit Intalio|BPMS, OMII-BPEL, Triana, Taverna, Karajan, GridAnt, Virtual Data System, Kepler und g-Eclipse ausschließlich Systeme betrachtet, die explizit für die Verwendung im Grid-Kontext entwickelt wurden oder bereits in Grid-Umgebungen eingesetzt werden. Dabei zeigte sich, dass keines der Systeme perfekt zu TextGrids Anforderungsprofil passt, weshalb in jedem Fall für Anpassungen ein nicht zu vernachlässigender Entwicklungsaufwand notwendig ist.

Für die Nutzerverwaltung stehen mit VOMS bzw. VOMRS sowohl zentralisierte als auch mit den auf Shibboleth basierenden myVocs und GridShib föderale Ansätze zur Verfügung, wobei sich die Systeme innerhalb eines Ansatzes jeweils gegenseitig ergänzen. Pluspunkt von VOM(R)S ist, dass in D-Grid bereits entsprechende Dienste für den Regelbetrieb zur Verfügung stehen; die föderalen Systeme scheinen langfristig für TextGrid aber vorteilhafter, weil bei diesen die Nutzeridentifizierungen durch die Heimateinrichtungen, z. B. gebunden an den jeweiligen Account beim (Universitäts-)Rechenzentrum erfolgen kann.

Der aus den Unix-Filesystemen bekannte Zugriffsschutzmechanismus ist leicht verständlich, für TextGrid aber wohl zu simplistisch. PERMIS ist demgegenüber eine sehr elaborierte Lösung zum Treffen von Autorisierungsentscheidungen, die aber auch hochkomplex und nicht in allen Punkten kompatibel zu aktuellen Weiterentwicklungen der zu Grunde liegenden Softwarekomponenten ist. Weil die Lizenz von PERMIS im Falle evtl. kommerzieller Nutzungen von TextGrid Probleme aufwerfen würde, scheint eine Implementierung des RBAC-Standards am zielführendsten.

Anhang A: Bibliographie

[1] *TextGrid Szenarien*. TextGrid-Projektbericht, Dezember 2006. http://www.textgrid.info/fileadmin/TextGrid/TextGrid-Szenarien_061212.pdf

[2] *Bericht über eine Evaluation von Grid Middleware Standards und von Grid Software Paketen.* TextGrid-Projektbericht, August 2006. http://www.textgrid.info/fileadmin/TextGrid/reports/TextGrid_Report_3_1.pdf

[3] *TextGrid-Architektur*. TextGrid-Projektbericht, Januar 2007. http://www.textgrid.info/fileadmin/TextGrid/reports/TextGrid_Report_3_2.pdf

[4] ANSI/INCITS 359-2004: *Information Technology - Role Based Access Control.* International Standard, Februar 2004.