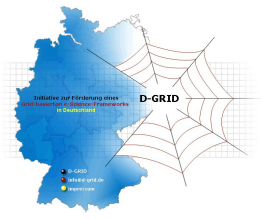
Bericht über eine Evaluation von Grid Middleware Standards und von Grid Software Paketen

Version August 2006
Arbeitspaket AP 3, Report 3.1
verantwortlicher Partner: SUB Göttingen

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: 07TG01-A

Laufzeit: Februar 2006 - Januar 2009

Dokumentstatus: Endversion

Verfügbarkeit: öffentlich

Autoren: Andreas Aschenbrenner, SUB

Peter Gietz, DAASI

Martin Haase, DAASI

Frank Knoll, DAASI

Marc W. Küster, FH Worms

Christoph Ludwig, FH Worms

Anhang A: Bibliographie 37

1. Einleitung

TextGrid (www.textgrid.de) ist das einzige geisteswissenschaftliche Projekt im Rahmen der deutschen Grid Initiative D-Grid (www.d-grid.de). Als Teil von D-Grid strebt TextGrid nach einer Integration mit der D-Grid Infrastruktur und Zusammenarbeit mit den Partnern aus anderen Communities. Die vorhandene Infrastruktur und die Anforderungen aus der textwissenschaftlichen Community sind dabei taktgebend. Dieser Bericht fokussiert auf Grid Standards und Tools, die für die Realisierung von TextGrid wichtig sein können.

1.1Das "Grid" in TextGrid

Oft werden verschiedene Ausrichtungen von Grid Umgebungen hervorgehoben: Computational Grid, Service Grid, oder Data Grid. Systeme in der Praxis kommen selten in einer dieser drei Reinformen vor, aber Schwerpunkte bildensich meist heraus.

Ein Computational Grid wird errichtet um rechenintensive Aufgaben wie zB Simulationen verteilt und möglichst schnell durchführen zu können. Systeme unterscheiden sich stark in Aspekten wie den zugrunde liegenden Ressourcen - eine Handvoll Supercomputer oder eine grosse Menge durchschnittlicher Rechner -, der Verteilung der Aufgabe - nicht alle Berechnungen lassen sich (einfach) verteilen -, und wie verfügbare Ressourcen und Prozessorzyklen einer Berechnung zugeteilt werden.

Je nachdem welche Community den Term "Service Grid" verwendet, schwingen unterschiedliche Ausrichtungen mit. In der Wirtschaft werden Web Services verwendet um angebotene Dienste an den Kunden zu bringen. Obwohl auch zu diesem Kontext manchmal "Service Grid" gesagt wird, hat diese Umgebung eigentlich nur am Rande mit "Grid" Systemen zu tun. Allgemein versteht man unter "Service Grids" die Verwendung von Services und von Konzepten der Service Oriented Architecture zum Aufbau einer Grid Umgebung (siehe Kapitel Grid Middleware Standards), und dabei vor allem auch die Endnutzer-orientierten Services und Tools. Die Wissenschaft arbeitet hier vor allem an der dynamischen Auswahl der 'richtigen' Services für den passenden Kontext [asg], und damit auch an die Anbindung der Semantic Web Community zum Semantic Service Grid [gsmo].

In einem Data Grid werden Daten verwaltet. Verteilte Ressourcen werden zu einem virtuellen Speichersystem zusammengeschlossen, in dem Daten zuverlässig, effizient und sicher abgelegt werden können. Zu den Themen zählen redundante Speicherung, Registration und Retrieval, und andere, die im entsprechenden Kapitel unten besprochen werden.

Wie jedes System in der Praxis ist auch TextGrid eine Mischform aus diesen drei Grid Reinformen. Ein zentraler Aspekt ist ganz klar der des Data Grid. In dem Zusammenschluss und der Virtualisierung der Daten aus verteilten, heterogenen und unabhängig verwalteten Text-Archiven werden Data Grid Funktionalitäten mit konventionellen Schnittstellen und Tools aus der textwissenschaftlichen Community verknüpft. In TextGrid wird auch die inhaltliche Verknüpfung der Daten eine wesentliche Rolle spielen. In diesem Sinne gehen die Funktionalitäten über die eines "Data Grid" im gängigen Sinn der Grid Community hinaus.

Jenseits der digitalen Objekte ist die zu konstruierende Workbench ein zentraler Aspekt von TextGrid. Die Grafik (TODO Architekturgrafik einbinden) zur ersten Version der TextGrid Architektur zeigt eine eigene Sevice Layer als Brücke zwischen der Grid Middleware und den Tools. Eigenschaften wie die

Strukturierung und Vermittlung von passenden Services und deren Anordnung in Workflows sind die Kernaufgaben in der TextGrid Service Layer.

1.2Dieser Bericht

Dieser Bericht gliedert sich in zwei Bereiche: Grid Standards, die als Grundlage für eine Realisierung von TextGrid dienen könnten, und vorhandene Grid Software Pakete, die diese Standards implementieren und von TextGrid verwendet werden könnten. Konventionelle Infrastrukturkomponenten (i.e. nicht Grid), oder solche, die in der geisteswissenschaftlichen Community verbreitet sind - zB das Metadaten Protokoll OAI [oai], oder die Schnittstellendefinition ZING-SRW [zing-srw] - werden hier nicht behandelt. Szenarien und Anforderungsanalysen werden parallel erarbeitet, liegen aber der Auswahl der Standards und Software Pakete und den in diesem Dokument eingebetteten Bewertungen zugrunde. Breitere, tiefergehende, und vergleichende Evaluationen werden vom D-Grid Infrastrukturprojekt (dgi.d-grid.de) durchgeführt und von TextGrid verfolgt.

In beiden Bereichen, der Beschreibung einiger relevanter Grid Standards und die vorhandener Software, werden sowohl grundlegende Grid Infrastrukturkomponenten, als auch spezialisierte Komponenten zur Errichtung eines Daten- bzw Service-Grids betrachtet.

2.Grid Middleware Standards

Ein zentrales Paradigma für Grid Umgebungen ist die "Service Oriented Architecture" (SOA) [erl04]. Aktuelle Grid Standards basieren auf einer solchen, "loosely-coupled" Architektur. Die in [foster_phys] erstmals beschriebene "Open Grid Services Architecture" (OGSA) [ogsa_v1] beinhaltet auch heute noch SOA-basierte grundlegende Konzepte für Grid Architekturen. Eine erste detailliertere Spezifizierung der Konzepte von OGSA in der Form von Open Grid Services Infrastructure (OGSI) ist inzwischen allerdings vom Web Service Resource Framework (WSRF) abgelöst worden, und hat sich damit relevanten Standards aus der Web Community stark angenähert. [reinefeld-schintke]

[DINI-reinefeld]

Die Standardisierungsinstanz in der Grid Community ist das Global Grid Forum (GGF) (http://www.ggf.org/), das sich nunmehr mit der Enterprise Grid Alliance (EGA) zum Open Grid Forum (OGF) zusammengeschlossen hat und damit noch an Einfluss gewonnen hat. Durch das Zusammenwachsen mit Web Standards haben in der Grid Community natürlich auch das W3C (http://www.w3.org/), OASIS (http://www.oasis-open.org/), IETF (http://www.ietf.org/) eine wichtige Position. Diverse ausgewählte Standards von diesen Organisationen werden in diesem Kapitel betrachtet. Dazu zählen Web Services, WSRF und andere aktuelle Standards in Grid Umgebungen wie die "Grid Security Infrastucture" (GSI). Neben allgemeinen Standards wird dabei vor allem auch auf die Service- und die Datenmanagement-Aspekte für TextGrid geachtet.

2.1Web-Services

Ein Web-Service ist ein internetbasierter serverseitiger Dienst, der Clients eine öffentliche Schnittstelle zur Verfügung stellt, über die der Client den Web-Service ansprechen kann. Ein Web-Service wird nicht direkt von einem Menschen bedient, sondern indirekt von einer Applikation, die die öffentliche Schnittstelle des Web-Service anspricht, und die vom Web-Service zurückerhaltenen Ergebnisse für den Benutzer aufbereitet (z.B. in einem Browser). Kurz gesagt gilt: Was Internetseiten für Menschen sind,

sind Web-Services für Applikationen. Da in TextGrid Werkzeuge definiert werden, die im Grid verteilt werden sollen, bieten sich Web-Services als Technologie für diese Werkzeuge an.

Die folgenden drei Abschnitte beschreiben einen Standard zur Definition der öffentlichen Schnittstelle eines Web-Services (WSDL), ein Protokoll, auf dessen Grundlage die Kommunikation der Web-Services untereinander stattfinden kann (SOAP), sowie einen Retrievalmechanismus, durch den automatisch Web-Services registriert und gefunden werden können (UDDI).

2.1.1 Web Services Description Language (WSDL)

Die öffentliche Schnittstelle eines Web-Service beschreibt ein so genanntes WSDL-Dokument. In ihm werden die zur öffentlichen Schnittstelle des Web-Service gehörenden Operationen und ihre Signaturen (d.h. Eingabe- und Ausgabe-Parameter) beschrieben. Die dabei verwendeten Datentypen werden in Form von XML-Schema-Definitionen angegeben. Damit ein Client einen Web-Service über seine WSDL-Schnittstelle ansprechen kann, müssen zwischen dem Client und dem Web-Service Informationen - wie z.B. die auszuführende Operation, aktuelle Eingabeparameter und das Ergebnis - hin- und hergeschickt werden. Als konkrete Realisierung für das Format dieser Nachrichten bietet sich das SOAP-Protokoll an und als konkretes Transportprotokoll zum Verschicken solcher SOAP-formatierten Nachrichten das HTTP-Protokoll.

2.1.2 SOAP

SOAP ist ein Protokoll, mit dem XML-basierte Informationen in einer dezentralisierten, verteilten Umgebung - wie einem Grid - ausgetauscht werden können. Im Rahmen von Web-Services sind diese XML-Informationen Aufrufe, Ergebnisse und Fehlermeldungen von Web-Service-Operationen. Der Informationsaustausch zwischen einem Sender und einem Empfänger kann dabei entlang einer Nachrichtenkette über dazwischengeschaltete Einheiten stattfinden, die die Informationen verarbeiten oder sogar nach genau zu beschreibenden Regeln verändern können.

Eine SOAP-Nachricht ist ein XML-Dokument, das aus einem XML-Element namens Envelope besteht, in dem sich zwei unmittelbare XML-Unterelemente namens Header und Body befinden. Das Header-Element ist optional und dafür gedacht, Kontrollinformationen von Applikationen zu transportieren. Das Body-Element soll die Hauptinformation des Nachrichtenaustausches zwischen zwei Parteien beinhalten. Die im Header untergebrachten Kontrollinformationen können von in der Nachrichtenkette dazwischengeschalteten Einheiten ausgewertet werden, wobei der Inhalt des Body-Elementes nur vom endgültigen Nachrichtenempfänger ausgewertet werden darf. Um den Adressaten einer im Header-Element befindlichen Information entlang der Nachrichtenkette zu identifizieren, tragen diese Informationen ein Role-Attribut, mit dem sie festlegen, wer entlang der Nachrichtenkette diese Information verarbeiten darf oder sogar muss. Als Werte für das Role-Attribut sind unter anderem "next" für die nächste Einheit in der Nachrichtenkette und "ultimateReceiver" für den Endadressaten der Nachricht möglich.

Um eine SOAP-Nachricht zwischen zwei Parteien zu verschicken, muß ein konkretes Transport-protokoll, mit dessen Hilfe die Nachricht verschickt werden soll, angegeben werden. . Viele Pro-

tokolle sind dabei prinzipiell möglich. Das am häufigsten verwendete Protokoll ist das Hypertext-Transfer-Protocol (HTTP) - sprich: SOAP über HTTP, deutlich seltener das Simple-Mail-Transfer-Protocol - sprich SOAP über E-Mail.

Im Rahmen von TextGrid bietet es sich an, für den dazugehörenden Nachrichtenaustausch SOAP über HTTP zu verwenden.

2.1.3 Registry-Normen

Im „klassischen“ Web-Service-Stack war der Registry-Norm Universal Description, Discovery and Integration (UDDI) die Funktion des universellen Registry-Standards zugedacht. UDDI, ein OASIS-Standard, der aktuell in der Version 3.0 vorliegt, ist im Kern eine Menge von SOAP-Schnittstellen und dazugehörigen Informationsmodellen für das Veröffentlichen, Suchen und Abonnieren von Informationen zu Diensteanbietern („businesses“) und deren konkreten Diensten („services“). Dazu kommen Hilfsdienste etwa zur Authentifizierung der Teilnehmer oder zur Replikation von Daten zwischen UDDI-Instanzen.

UDDI erlaubt den Zugriff auf die gespeicherten Informationen u. a. über eine an SQL angelehnte Freitextsuche oder über bestimmte, mit den Einträgen verknüpfte Konzepthierarchien wie NAICS und UNSPSC.

UDDI hat sich in der Praxis bislang als universeller Registry-Standard nicht im ursprünglich erwarteten Maß durchsetzen können. Als ähnlich erfolgreich hat sich in zahlreichen Szenarien mittlerweile auch außerhalb der ebXML-Community ein anderer OASIS-Standard erwiesen, der ebXML Registry Standard, der ebenfalls in der Version 3.0 vorliegt und aus zwei Teilen besteht, dem Registry Registry Information Model (RIM) und dem eigentlichen ebXML Registry Services Standards (RS). Die grundsätzliche Funktionsweise ist sehr vergleichbar – auch ebXML RS ist im Kern eine Menge von SOAP-Schnittstellen mit zugehörigen Informationsmodellen –, aber einige Experten sehen die konkrete Umsetzung der Ideen sowie die Softwareunterstützung in der Gestalt des Open-Source-Servers Omar als besser an. Insbesondere werden das Modell vernetzter Registry-Instanzen („federated registries“), zu dem es in UDDI keine Entsprechung gibt, sowie die Metadatenunterstützung als sehr gelungen betrachtet.

Teils orthogonal, teils in direkter Konkurrenz zu UDDI und ebXML RS stehen Topic Maps. ISO/IEC 13250 „Topic Maps“ definiert die Datenstruktur zur Beschreibung von Wissensstrukturen / Ontologien sowie (als XTM) ein standardisiertes XML-Austauschformat dafür. Einerseits sind Topic Maps ein genereller Mechanismus zur Wissensmodellierung, mit dem sich bestimmte Konzepte mit flexibel definierbaren Assoziationen untereinander und mit konkreten Objekten / Ressourcen verknüpfen lassen. In diesem Sinne lassen sich Topic Maps auch etwa als Kategoriensystem in ebXML RS einsetzen. Andererseits bieten Topic-Map-Server wie etwa das Open-Source-Projekt tm4j oder kommerzielle Angebote wie die Ontopia Knowledge Suite von Ontopia oder TMCore von NetworkedPlanet vollausgeprägte semantische Suchmaschinen, die in einem Gridprojekt auch allein stehend die Funktion einer flexiblen Registry übernehmen können.

2.2 Web Services Addressing

Web-Services-Addressing spezifiziert, wie Instanzen eines Web-Service (oder Web-Service-Endpoints) transportprotokollunabhängig adressiert werden können. Dazu wird ein XML-Element namens EndpointReference definiert, welches die folgenden Unterelemente für die Adressierung einer Web-Service-Instanz besitzt:

- Address: die URI des Web-Service
- ReferenceParameters: optionale XML-Elemente zur Identifizierung der Instanz des Web-Service
- Metadata: optionale XML-Elemente, die das Verhalten und die Fähigkeiten der Web-Service-Instanz beschreiben

2.3Web Services Resource Framework (WSRF)

Die Spezifikation eines Web-Service macht keine Angaben über die Behandlung von Zuständen. Demnach kann ein Web-Service zwar einen (internen) Zustand haben (wie es für fast jede Applikation der Fall ist), die Schnittstelle zu dessen Abfrage und Manipulation ist jedoch nicht standardisiert. Dies erschwert eine Integration von Web-Services in eine Grid-Umgebung, in der z.B. der Zustand eines Web-Services in einem Informationsdienst hinterlegt werden soll; die Integration ist deswegen schwer, weil der Informationsdienst die Zustandsabfrage des Web-Service nicht nach einem Standard abhandeln kann, sondern auf proprietäre Schnittstellen angewiesen ist. Um dieses offensichtliche Problem der nicht standardisierten Behandlung der Zustände von Web-Services zu beheben, wurde der Standard "Web Services Resource Framework" (WSRF) eingeführt. WSRF ist eine Sammlung von mehreren Spezifikationen, deren Inhalt und mögliche Bedeutung für TextGrid in den folgenden Abschnitten beschrieben werden. Neben dem eigentlichen WSRF-Standard wird der Standard "Web Services Notification" (WSN) beschrieben, der die Kommunikation und den Nachrichtenaustausch von Web-Services untereinander standardisiert.

Bei WSRF handelt es sich noch um eine sehr neue Technologie, die noch nicht in einfach zu benutzenden und produktionsreifen Bibliotheken zur Verfügung steht. Deshalb muß sich TextGrid zunächst darum bemühen, WSRF-Funktionalität vor dem Anwendungsprogrammierer zu kapseln.

2.3.1 Web Services Resource (WS-Resource)

Um Zustände in Web-Services einzuführen, definiert der WS-Resource-Standard den Begriff Resource. Eine **Resource** ist demnach eine logische Entität, die eine Menge von in XML darstellbaren Eigenschaften (ResourceProperties) hat. Diese ResourceProperties werden im nächsten Abschnitt ausführlicher definiert. Kurz gesagt: Eine Resource kapselt den Zustand eines Web-Service. Ein Zustand (Resource) ohne eine Möglichkeit der Abfrage oder Manipulation dieses Zustandes macht keinen Sinn, deshalb definiert dieser Standard den Begriff WS-Resource. Demnach ist eine **WS-Resource** ein Paar bestehend aus einer Resource und einem Web-Service (WS), durch den die Resource angesprochen werden kann. Eine WS-Resource wird über eine End-Point-Reference (EPR) gemäß der WS-Addressing-Spezifikation (s.u.) adressiert. Kurz gesagt ist eine WS-Resource ein Web-Service mit Zustand. Der Zustand ist über die Web-Service-Schnittstelle ansprechbar.

Da moderne Grid-Implementierungen auf WSRF aufsetzen, werden auch innerhalb von TextGrid Grid-Dienste als WS-Resourcen angesprochen.

2.3.2 Web Services Resource Properties (WS-ResourceProperties)

Um den Zustand eines Web-Service allgemein ansprechen und manipulieren zu können, definiert der WS-ResourceProperties-Standard als Vorleistung, woraus der Zustand (die Resource) eines Web-Service überhaupt besteht, um dann anschließend zu beschreiben, wie dieser Zustand über eine Web-Service-Schnittstelle abgefragt und manipuliert werden kann.

Die Entitäten, aus welchen der Zustand eines Web-Service besteht, werden als **ResourceProperties** bezeichnet und sind als XML-Elemente beschreibbar. Der Zustand eines Web-Service ist eine Zusammenstellung solcher ResourceProperties, die in einem XML-Dokument namens **Resource-Properties-Document** hinterlegt sind. Dieses Dokument ist mit einem WSDL-PortType des Web-Service verbunden. Genau diese Verbindung macht einen Web-Service und eine Resource zu einer Web-Service-Resource. Somit bietet ein Resource-Properties-Document eine logische Sicht auf den Zustand einer WS-Resource. Die Web-Service-Schnittstelle, mit deren Hilfe der Zustand eines Web-Service abgefragt und manipuliert werden kann, stellt folgende Operationen zur Verfügung:

- GetResourcePropertyDocument: gibt das gesamte Resource-Properties-Document einer WS-Resource zurück
- GetResourceProperty: gibt die Werte eines einzelnen ResourceProperties des Resource-Properties-Documents zurück
- GetMultipleResourceProperties: gibt die Werte mehrerer ResourceProperties zurück
- QueryResourceProperties: Abfrage gegen das Resource-Properties-Document mit Hilfe eines XPath-Ausdrucks
- PutResourcePropertyDocument: ersetzt das Resource-Properties-Document durch ein neues
- SetResourceProperties: setzt die Werte von mehreren ResourceProperties
- InsertResourceProperties: fügt neue Werte eines ResourceProperties in das Resource-Properties-Document einer WS-Resource ein
- UpdateResourceProperties: ersetzt die bisherigen Werte eines ResourceProperties durch neue Werte.
- DeleteResourceProperties: entfernt alle Werte eines ResourceProperties einer WS-Resource

Sobald TextGrid-Werkzeuge als WS-Resourcen implementiert werden, könnte jedes Werkzeug einen Teil der oben angegebenen Operationen implementieren, um WSRF-konform zu sein und zum Beispiel für Informationsdienste einheitlich zugängliche Informationen zur Verfügung zu stellen.

2.3.3 Web Services Resource Lifetime (WS-ResourceLifetime)

In einer Grid-Umgebung sollen WS-Resourcen dynamisch erzeugt werden - belegen dann serverseitigen Speicherplatz, halten offene Verbindungen zu Datenbanken, etc. - und anschließend wieder gelöscht werden. Dies erfordert eine eigene Speicherverwaltung - im Gegensatz zu lokalen Anwendungen im von der Laufzeitumgebung verwalteten Speicher - über das gesamte Grid hinweg. Eine explizite Verwaltung der Lebenszeit von WS-Resourcen über new- und delete-ähnliche Konstrukte reicht in einer Grid-

Umgebung nicht aus, weil z.B. ein Client, der eine Instanz einer WS-Resource erzeugt, nicht mehr willens oder fähig sein könnte, diese zu löschen, z.B. wenn er selbst aufgrund eines Netzwerkfehlers oder internen Fehlers den Kontakt zur WS-Resource verliert; somit würde die WS-Resource "für immer" im Grid Speicherplatz beanspruchen. Um diese Speicherlecks zu vermeiden, definiert der WS-ResourceLifetime-Standard eine halbautomatische Speicherverwaltung, in der WS-Resourcen nach Ablauf einer konfigurierbaren Lebenszeit - das ist die Zeit, die von ihrer Instantiierung bis zu ihrer Zerstörung vergeht - gelöscht werden. WS-ResourceLifetime spezifiziert außerdem, wie die Lebenszeit einer WS-Resource manipuliert werden kann:

- Immediate Destruction (Destroy-Operation): die WS-Resource wird augenblicklich zerstört
- Scheduled Destruction (SetTerminationTime-Operation): die WS-Resource wird zu einer bestimmten Zeit zerstört

Im Rahmen einer objektorientierten TextGrid-API könnte jeder Klasse (z.B. der File-Klasse) ein Web-Service (z.B. File-WS) zugeordnet werden, der die öffentlichen Methoden der Klasse in seiner WSDL-Beschreibung anbietet. Die Instanzbildung einer Klasse kann dadurch implementiert werden, daß eine neue WS-Resource des entsprechenden Web-Service erzeugt wird. Die Lebenszeit dieser neuen WS-Resource könnte mit WS-ResourceLifetime verwaltet, und so eine halbautomatische Speicherverwaltung gewonnen werden.

2.3.4 Web Services Service Group (WS-ServiceGroup)

Web-Services treten nicht nur einzeln, sondern auch in Gruppen auf. Beispielsweise kann ein Informationsdienst Informationen von mehreren gleichartigen Web-Services sammeln, die dadurch als Gruppe betrachtet werden können. Der WS-ServiceGroup-Standard beschreibt, wie Web-Services zu Gruppen zusammengefaßt und verwaltet werden können. Dazu definiert er eine **ServiceGroup** als eine heterogene Gruppierung von Web-Services. Die Mitgliedschaft eines Web-Services in einer solchen Gruppe kann dabei an bestimmte Bedingungen (z.B. die Implementierung bestimmter WSDL-Schnittstellen oder das Vorhandensein bestimmter XML-Elemente im Resource-Properties-Document des Web-Service) geknüpft sein. Mit Hilfe der Add-Operation kann ein neuer Eintrag in eine ServiceGroup eingefügt werden. Parameter sind unter anderem der dazuzufügende Web-Service (MemberEPR) und die mit ihm assoziierte Information (Content). Um einen Web-Service aus einer ServiceGroup entfernen zu können, ohne den Web-Service selbst zu löschen, bedient sich der WS-ServiceGroup-Standard eines Hilfskonstruktes, nämlich indem die Mitgliedschaft eines Web-Service in einer ServiceGroup mit Hilfe einer weiteren den Web-Service stellvertretenden WS-Resource verwaltet wird. Der Web-Service kann dann aus der ServiceGroup entfernt werden, indem die Lebenszeit seiner stellvertretenden WS-Resource terminiert wird (ohne die Lebenszeit des Web-Service selbst zu terminieren).

Mit diesen Eigenschaften bildet WS-ServiceGroup die Grundlage für MDS (Monitoring and Discovery System) in Globus Toolkit 4, insbesondere für den Informationsdienst Index-Service.

2.3.5 Web Services Base Faults (WS-BaseFaults)

Der WS-BaseFaults-Standard spezifiziert ein XML-Format für Fehlermeldungen, die von Web-Service-Operationen zurückgeliefert werden. Ein BaseFault-XML-Element beinhaltet dabei folgende Informationen:

- Timestamp: die Zeit, zu der der Fehler aufgetreten ist
- OriginatorReference: eine optionale EndpointReference des Web-Service, der den Fehler erzeugt hat
- Description: eine optionale von Menschen lesbare Beschreibung des Fehlers
- FaultCause: enthält ein weiteres optionales BaseFault-XML-Element, wodurch Verkettungen von BaseFaults ermöglicht werden

Web-Services dürfen BaseFaults via XML-Schema-Extension beliebig erweitern, um anwendungsspezifische Fehlertypen einzuführen.

Mit den soeben beschriebenen Eigenschaften soll der WS-BaseFaults-Standard die Verständlichkeit und sinnvolle Behandlung von Fehlern in Grid-Umgebungen erleichtern.

2.4Web Services Notification (WSN)

Web-Services können untereinander (asynchron) kommunizieren, indem sie Nachrichten austauschen. Nachrichten können in Grid-Umgebungen in vielerlei Situationen auftreten: beispielsweise kann ein Informationsdienst eine Benachrichtigung an Interessenten immer dann verschicken, wenn es einen neuen Eintrag gibt, der bestimmten Bedingungen genügt; oder ein Web-Service gibt seine baldige Zerstörung in Form einer Benachrichtigung im Voraus bekannt; oder ein Web-Service teilt anderen Web-Services mit, daß er seine Aufgabe schon zu 50% erledigt hat.

Der WSN-Standard beschreibt einen Rahmen, in welchem Web-Services Nachrichten austauschen können. Er besteht aus drei Unterspezifikationen, die in den folgenden Abschnitten zusammengefasst werden: Der WS-BaseNotification-Standard beschreibt die zum Nachrichtenaustausch notwendigen Web-Service-Schnittstellen, der WS-Topics-Standard beschreibt die Ereignisse, die einen Nachrichtenaustausch auslösen und der WS-BrokeredNotification-Standard beschreibt einen Nachrichtenaustausch über eine dritte Partei.

2.4.1 Web Services Base Notification (WS-BaseNotification)

Dieser Standard spezifiziert einen Web-Service-basierten Ansatz für Benachrichtigungen (notifications). Teilnehmer beim Nachrichtenaustausch sind ein Subscriber-Web-Service, ein NotificationProducer-Web-Service (Nachrichtenerzeuger) und ein NotificationConsumer-Web-Service (Nachrichtenempfänger). Der Subscriber meldet einen NotificationConsumer bei einem NotificationProducer an, Nachrichten zu einem bestimmten Ereignis (Topic) von ihm zu empfangen.

Bei Eintreten eines Ereignisses, für welches der NotificationConsumer durch einen Subscriber Interesse signalisiert hat, ruft der NotificationProducer die Notify-Operation des NotificationConsumers auf. Parameter sind das Ereignis (Topic), der NotificationProducer selbst und eine Nachricht (Message).

Ein NotificationProducer muß ein TopicExpression-Resource-Property zur Verfügung stellen, welches beschreibt, welche Ereignisse (Topics) von ihm unterstützt werden.

Ein NotificationProducer muß eine Subscribe-Operation zur Verfügung stellen. Ihre Parameter sind:

- ConsumerReference: ein Verweis auf einen NotificationConsumer, der Ereignisse vom NotificationProducer empfangen soll
- Filter: eine Beschreibung der Ereignisse, die dem NotificationConsumer mitgeteilt werden sollen.

Kombinationen aus folgenden Filtern sind zulässige Parameter:

- TopicExpression: eine Beschreibung eines Ereignisses (Topic).
- ProducerProperties: ein boolscher Ausdruck, der auf den ResourceProperties des NotificationProducer ausgewertet wird.
- MessageContent: ein boolscher Ausdruck, der auf der Nachricht (Message) des NotificationProducers ausgewertet wird.

Alle angegebenen Filter müssen wahr ergeben, damit eine Nachricht beim Eintreten eines Ereignisses gesendet wird.

Als Ergebnis der Subscribe-Operation entsteht eine Subscription-WS-Resource, die die Beziehung zwischen dem NotificationProducer und dem NotificationConsumer beschreibt.

Die Unsubscribe-Operation der SubscriptionManager-Schnittstelle beendet eine Subscription, wodurch der NotificationConsumer der Subscription entsprechende Nachrichten nicht mehr empfängt. Weitere Operationen der SubscriptionManager-Schnittstelle sind:

- Renew: modifiziert die Lebenszeit der Subscription-WS-Resource.
- PauseSubscription: die Erzeugung von Nachrichten wird für eine bestimmte Zeitdauer ausgesetzt.
- ResumeSubscription: hebt die Wirkung von PauseSubscription wieder auf.

2.4.2 Web Services Brokered Notification (WS-BrokeredNotification)

Ein NotificationBroker ist ein Web-Service, der Publisher und NotificationConsumer trennt. Ein NotificationBroker kann sich im Namen mehrerer NotificationConsumer für die Nachrichten eines NotificationProducers subskribieren und die so erhaltenen Nachrichten anstelle des NotificationProducers an die NotificationConsumer weiterleiten. Somit entlastet ein NotificationBroker einen NotificationProducer bei der Verwaltung und Benachrichtigung der NotificationConsumer.

WS-BrokeredNotification ist in Globus-Toolkit noch nicht implementiert.

2.4.3 Web Services Topics (WS-Topics)

WS-Topics definiert einen Mechanismus, Ereignisse als Topics zu beschreiben und zu organisieren. Ein Web-Service kann eine Menge von Topics veröffentlichen, für die sich ein Client subskribieren kann; bei einer Änderung eines Topics wird der Client benachrichtigt.

Jedes Topic ist einem XML-Namespace zugeordnet, um Namenskonflikte bei gleichnamigen Topics zu vermeiden. Die Menge der Topics, die einem XML-Namespace zugeordnet sind, wird als Topic

Namespace bezeichnet. Jedes Topic in einem Topic-Namespace kann keine oder mehrere Child Topics haben, und ein Child-Topic kann selbst weitere Child-Topics haben. Ein Topic ohne Vorgänger heißt Root Topic. Ein Root-Topic und seine Nachfahren bilden einen Topic Tree.

Ein Client kann sich für ein Topic subskribieren und automatisch Benachrichtigungen für alle seine Nachfahren im Topic-Tree erhalten, ohne sich für sie explizit subskribieren zu müssen.

2.5Grid Security Standards

Grundsätzlich läßt sich der Bereich Security im Grid in zwei Kernbereiche unterteilen: Authentifizierung, also der Vorgang des Beweises einer Identität eines Benutzers, und Autorisierung, nämlich der Vorgang einem authentifizierten Benutzer aufgrund von dessen Attribute (insbesondere Rollenzuweisungen) eine Entscheidung zu treffen, ob der Benutzer auf bestimmte Ressourcen zugreifen darf oder nicht.

Eine Authentitierungs- und Autorisierungsinfrastruktur (AAI) ist eine unabdingbare Voraussetzung für den Betrieb einer Grid-Infrastruktur um Benutzern sowohl Daten-Ressourcen, als auch Compute-Ressourcen kontrolliert zu gewähren

Standards zu den Bereichen Authentifizierung, Autorisierung, SingleSignOn und Zugriffskontrolle werden im Folgenden näher behandelt.

2.5.1 Authentifizierung

Authentifizierung setzt ein Benutzermanagement voraus, welches Daten enthält, mit denen der Identitätsbeweis überprüft werden kann. Im einfachsten Fall sind die Login-Name (userid) und Passwort. Verschiedene Rechnerarchitekturen verfügen hierbei über verschiedene Mechanismen und Protokolle solche einfachen Login-Vorgänge durchzuführen.

Hierbei können zentralisierte Dienste sowohl ein einheitliches Passwort für verschiedene Rechner und Anwendungen ("unified password") ermöglichen, als auch darüberhinaus ein, dass ein einmaligen Authentifizierungsprozess einen Benutzer über einen gewissen Zeitraum als Authentifiziert gelten läßt (SingleSignOn).

Wichtige Standards für solche Authentifizierung sind LDAP, SASL, und Kerberos. Identität läßt sich aber auch - sicherer als einfache Passwortabfrage - durch Identitätszertifikate beweisen. Hier ist der Standard X.509 auch im Gridbereich von herausragender Relevanz. Schließlich haben sich im Zuge der Web-Service-Standardisierung weitere auf die bisher genannten Standards aufbauende Standards entwickelt, insbesondere der allgemeine Standard WS-Security, sowie SAML.

Alle diese Standards werden im Folgenden kurz behandelt.

2.5.1.1 SASL

SASL (Simple Authentication and Security Layer) standardisiert den Security-Aspekt von verbindungsbezogenen Protokollen. Innerhalb dieses Layers werden verschiedene

Authentifizierungsmechanismen unterstützt, insbesondere ein direktes Passwort-Login, ein sogenannter Challenge-Response-Mechanismus (Digest-MD5) der verhindert, dass das Passwort über das Netz übertragen werden muss, sowie eine Mechanismus für GSSAPI, durch den weitere Verfahren zur Verfügung stehen.

2.5.1.2 GSSAPI

GSSAPI (Generic Security Service Application Program Interface) ist ein weiterer Standard, der verschiedene Authentifizierungsmethoden auf eine generische Anwendungs- und Protokollunabhängige Weise ermöglicht. Insbesondere wird der wichtige Standard Kerberos 5 unterstützt. Durch GSSAPI wird eine einfache Programmierung von Sicherheitsmechanismen von Netzanwendungen ermöglicht.

2.5.1.3 Kerberos

Kerberos ist ein Netzwerk-Authentifizierungsprotokoll für Client-Server-SCenarien, bei dem jeder Teilnehmer ein gemeinsames Geheimnis (Passwort) teilt. An einem zentralen Authentifizierungsdienst kann sich der Benutzer anmelden (ohne dass dabei das Passwort übertragen werden muss) und erhält sogenannte Tickets, die den Benutzer für eine bestimmte Zeit als authentifiziert ausweist. Hierdurch ermöglicht Kerberos SingleSignOn.

2.5.1.4 LDAP

LDAP (Lightweight Directory Access Protocol) hat sich aus dem Standard X.500 entwickelt. Es beschreibt Datenmodell und Zugriffsprotokoll für einen Verzeichnisdienst. Insbesondere im Bereich Benutzermanagement hat sich LDAP durchgesetzt, sodass zentrale Benutzerverwaltungen, auf die von verschiedenen Rechnern und Anwendungen zugegriffen werden können, es also nur eine Stelle gibt, an der Daten und Passwörter gepflegt werden müssen, sehr oft mittels LDAP-Servern implementiert werden. Das LDAP-Protokoll spezifiziert hierbei verschiedene Authentifizierungsmethoden, neben einfacher Passwortabfrage auch alle SASL-Mechanismen, bis hin zu Kerberos 5. LDAP wurde in früheren Grid-Implementierungen als Informationsspeicher verwendet. Verzeichniseinträge werden in einem hierarchischen Baum organisiert und sind über den sogenannten Distinguished Name (DN), welcher die Knotennamen bis hin zur Baumwurzel enthält, eindeutig referenzierbar.

2.5.1.5 X.509

Ursprünglich nur als starker Authentifizierungsmechanismus für X.500 spezifiziert hat sich X.509 zu einer der wichtigsten Netz-Sicherheitstechnologien entwickelt. Es basiert auf asymmetrischer Verschlüsselung mittels eines Schlüsselpaares bestehend aus einem öffentlichen und einem privaten Schlüssel. Mittels des privaten Schlüssels kann man sowohl digital signieren, als auch Information, die mittels des öffentlichen Schlüssels verschlüsselt wurde, wieder entschlüsseln. Um mit diese Technologie zum Identitätsbeweis (Authentifizierung) verwenden zu können, muss der öffentliche Schlüssel wiederum von einer Zertifizierungsstelle (CA) digital signiert sein, wobei diese bestätigt, dass die Identität des Schlüsselpaarbesitzers, z.B. durch Einsicht in einen Personalausweis überprüft worden ist. Ein so signierter öffentlicher Schlüssel wird Zertifikat genannt.

Wichtige Bestandteile eines X.509-Zertifikats sind der subjectDN und der issuerDN, welche eindeutig den Besitzer des Zertifikats und die Zertifizierungsstelle referenzieren.

Auf X.509 beruht nicht nur die im Web gängige Verbindungsverschlüsselung SSL (bzw. TLS), sondern auch die neueren unter dem Namen Web Service Security zusammengefassten Standards. Auch im Grid-Computing wird augenblicklich vorwiegend X.509 für Authentifizierungszwecke verwendet. Auf X.509 aufbauende Infrastrukturen (insbesondere Zertifizierungsstellen) werden PKI (Public Key Infrastructure) genannt.

2.5.1.6 Web Service Security

Web Service Security ist eine Erweiterung des Web Service Standards, durch die Authentifizierung, sowie Integrität und Verschlüsselung der mittels SOAP übertragenen Daten gewährleistet wird. Hierbei werden die XML-Standards XML-Signature und XML-Encryption verwendet, die wiederum auf X.509 beruhen. Diese werden im Header der SOAP-Daten integriert.

2.5.1.7 SAML

Die auf XML basierende Beschreibungssprache SAML (Security Assertion Markup Language) ermöglicht es, Ergebnisse von Authentifizierungsvorgängen (ähnlich wie bei Kerberos) mittels sogenannter authenticationAssertions zu übertragen und ermöglicht so SingleSignOn.

Auch im Bereich Authorisierung können SAML-Assertions übertragen werden, einerseits Attribute Assertions die bestimmte Attribute eines Benutzers übergeben und somit Autorisierungsentscheidungen vorbereiten, als auch Authorization Decision Assertions womit festgestellt wird ob und wie auf eine spezifische Ressource zugegriffen werden darf. Solche Assersions werden von sogenannten Authoritäten ausgegeben und mittels X.509-Technologie digital signiert. SAML entwickelt sich auch im Grid-Computing zunehmend als wichtigster Standard für AAIs.

2.5.1.8 GSI

Unter Grid Security Infrastructure versteht man eine ursprünglich für Globus entwickelte Sicherheitsschicht, die sich aber zunehmend auch bei anderen Grid-Implementierungen durchsetzt, und im Rahmen der GGF/OGF standardisiert wurde und wird.

Hauptbestandteile von GSI sind einerseits die bereits besprochene PKI und SSL/TLS Verbindungsverschlüsselung, andererseits das auf PKI aufbauende Konzept eines Proxy-Zertifikats. Mittels eines normalen X.509-Zertifikats können kurzfristig gültige aber sonst einem X.509-Zertifikat entsprechende sogenannte Proxy-Zertifikate ausgestellt werden, mit denen sich ein vom Inhaber des Benutzerzertifikats gestarteter Prozess im Grid authentifizieren kann. Innerhalb des Grids wird der subjectDN des Zertifikats einem lokalen User auf dem jeweiligen Rechner zugeordnet, dessen Zugriffsrechte durchgesetzt werden. So authentifizierte Prozesse können selber wiederum weitere Prozesse mit vom Proxy-Zertifikat abgeleiteten weiteren Proxy-Zertifikaten auf weiteren Ressourcen starten.

GSI verwendet die bereits erwähnte GSSAPI als API um die X.509-Funktionen Auentifizierung, Daten-Verschlüsselung und -Integrität zur Verfügung zu stellen. Für das Grid-Computing wurden hierbei einige GSSAPI-Erweiterungen standardisiert, insbesondere um die Delegation mittels Proxy-Zertifikaten zu ermöglichen.

2.5.2 Autorisierung

Auch im Bereich Autorisierung gibt es wichtige Standards, die im Grid-Computing relevant wurden. Neben dem bereits behandelten SAML sind hier insbesondere X.509-Attributzertifikate, RBAC, XACML zu nennen. Alle diese Standards sind im Bereich Grid-Computing relevant und werden im Folgenden kurz besprochen. Die GGF/OGF-Arbeitsgruppe OGSA-AuthZ spezifiziert entsprechende Profile. Es ist jedoch zu beobachten, dass sich SAML zunehmend als Transport für Attributinformationen durchsetzt.

In diesem Zusammenhang ist das Konzept der Virtuellen Organisation (VO) besonders wichtig. Eine VO ist der Zusammenschluss von Benutzern und Ressourcen aus verschiedenen realen Organisationen (der so genannten Heimatorganisation), zum Zwecke der Zusammenarbeit in einem gemeinsamen Projekt. Zugriffsrechte der einzelnen Benutzer werden von der VO über Attribute gesteuert. Eine wichtige Frage ist hierbei, inwieweit auch Attribute, die von der Heimatorganisation vergeben wurden, im Rahmen einer VO für Autorisierungszwecke verwendet werden können.

2.5.2.1 X.509 Attributzertifikate

Der X.509 Standard sieht neben einer PKI auch - darauf aufbauend - eine PMI (Priviledge Management Infrastructure) vor. Diese basiert auf Attributzertifikaten, die sich auf PKI-Benutzerzertifikate beziehen, aber von einer anderen Infrastruktur erstellt werden. In Attributzertifikaten werden einer Person Eigenschaften in Form von Attributen von sogenannten Attributautoritäten (AA) zugesprochen. Grundsätzlich lassen sich auch in PKI-Zertifikate Attribute integrieren, Vorteil des getrennten Attributzertifikat ist, dass eine andere Autorität diese Ausstellen kann. Würde der Staat ein PKI-Zertifikat als Identitätsausweis ausstellen, würde z.B. eine Rechtsanwaltkammer in einem Attributzertifikat das Attribut "diese Person ist Rechtsanwalt" ausstellen.

2.5.2.2 RBAC

RBAC (Role Based Access Control) ist ein standard, der rollenbasierte Zugriffskontrolle spezifiziert. Hierbei werden nicht nur verschiedene Rollenmodelle spezifiziert (einfach und hierarchisch) sondern auch Sessions (Netzverbindungen) berücksichtigt (ein Benutzer kann in verschiedenen Sessions verschiedene Rollen haben), sowie sogenannte Constraints, also Einschränkungen von Zugriffsrechte, z.B. auf bestimmte Tageszeiten, etc.

Des Weiteren definiert RBAC konkrete Funktionen z.B. zum Erstellen und Löschen von Rollenzuweisungen, oder zur Abfrage von Zugriffsrechten.

2.5.2.3 XACML

XACML (eXtended Access Control Markup Language) ist eine XML-Beschreibungssprache, mittels derer Zugriffsrechte in Form von Regelwerken (Policies) beschrieben werden können. XACML ist sehr ausdrucksfähig sodass komplexeste Regeln erstellt werden können. Unter Anderem kann der RBAC-Stamdard vollständig mit XACML abgebildet werden.

2.6Workflow-Beschreibungssprachen

Ein Workflow (oder Arbeitsablauf) ist eine Anordnung von einzelnen Aufgaben bzw. Prozessen, bei dem Eingabedaten über etwaige Zwischenstufen zu Ausgabedaten transformiert werden. Eine Workflow-Beschreibungssprache legt fest, wie ein Workflow-Management-System die rechnerische Abarbeitung eines bestimmten Workflows zu befolgen hat. Es existieren momentan einige Standards für Workflow-Beschreibungssprachen, sowie Formate für Grid-spezifische Workflows, wobei diese Mengen eher disjunkt zu sein scheinen, d.h. es gibt im Grid-Bereich noch keine Standardisierungs(bestrebungen) für Workflow-beschreibungen.

Bei den meisten Systemen ist eine zweigeteilte Architektur üblich:

- Im Workflow-*Design* (oder Editor) wird der Workflow erstellt, bearbeitet oder modifiziert und wird anschließend an
- die Workflow-Engine (oder Enactor) übergeben, die einen konkreten Workflow ausführt. Diese kann oftmals im Hintergrund, z.B. als Web Service, auf einem dedizierten Rechner laufen.

Die Workflow-Beschreibungssprache ist das Bindeglied zwischen diesen beiden Komponenten, d.h. Dokumente in dieser Sprache werden vom Editor erstellt und vom Enactor verarbeitet.

2.6.1 Workflow-Standards (nicht Grid-spezifisch)

Als wichtigste nicht Grid-spezifische Workflowbeschreibungssprachen sind vor allem drei Standards zu nennen:

- Die XML Process Definition Language (XPDL) ist der älteste Standard, herausgegeben von der Workflow Mangagement Coalition (WfMC).
- Die Business Process Execution Language (BPEL4WS bzw. WS-BPEL) ist ein OASIS-Standard und
- die Web Service Choreography Description Language (WS-CDL) ist eine W3C Recommendation (Public Working Draft)

Gemeinsam ist allen Standards, dass XML als Dateiformat verwendet wird. XPDL geht dabei nicht speziell von Web Services als Prozessen aus. Es dient u.A. als Dateiformat für BPMN-Graphen (Business Process Modeling Notation) und ist eine Graph-strukturierte Beschreibungssprache ohne eingebettete Prozesse.

BPEL geht von Web Services als Prozessen aus. BPEL baut auf dem Dienstmodell von WSDL auf. Weiterhin werden die Standards XMLSchema, XPath und WS-Addressing benutzt. Die Sprache ist Block-strukturiert, d.h. lokale Variablen könnnen angelegt und Prozesse geschachtelt werden. Ziel ist die `Progammierung im Großen', d.h. es wird die Schnittstelle eines Webservice, der die an einem Prozess beteiligten Web Services steuert, beschrieben (Orchestrierung).

Im Gegensatz zur Orchestrierung beschreibt die Choreographie (WS-CDL) die Interaktion von Prozessen verschiedener Parteien objektiv, also von einem unabhängigen Standpunkt. WS-CDL kann also dazu verwendet werden, ein Protokoll über die auszutauschenden Daten zwischen Organisationen zu spezifizieren, und diese setzen diese Schnittstelle dann beispielsweise jeweils mit WS-BPEL um.

2.6.2 Grid-spezifische Formate

Es ist eine Vielzahl von proprietären Lösungen zur Beschreibung von Workflows im Umlauf; hier wird nur auf diejenigen eingegangen, die im Grid-Kontext verwendet werden. Zu den generellen Anforderungen an Workflows kommt für Grid-spezifische Workflows noch hinzu, dass diese

- zustandsbehaftete Dienste verwalten können müssen,
- Mechanismen zur Fehlertoleranz vorhanden sein müssen,
- heterogene Rechen- und Speicherressourcen einbezogen werden müssen und
- intelligente Verfahren zur dynamischen Zuweisung und Verteilung von Ressourcen benötigt werden.

Dies spiegelt sich jedoch eher weniger in den verwendeten Workflow-Beschreibungsformaten wieder und ist dagegen meist Aufgabe der jeweiligen Workflow-Engines. Allerdings gibt es für einzelne dieser Aspekte in bestimmten Sprachen auch speziell dafür vorgesehene Konstrukte, z.B. für die Fehlerbehandlung in Condor DAGMan.

Die wichtigsten im Grid-Kontext verwendeten Formalismen sind:

- Gerichtete Graphen (z.B. Triana TaskGraph, Java CoGkit Karajan Format): Hier wird durch die Menge der Kanten eine Relation auf der Menge der Knoten definiert. Der Graph kann zusammenhängend oder nicht zusammenhängend sein, kann Schleifen (Loops) enthalten und es kann mehrere von Knoten A zu Knoten B führende Kanten geben (Mehrfachkanten).
- Gerichtete azyklische Graphen (DAG, directed acyclic graph) (z.B. Condor DAGMan, XSCUFL): Spezialfall von gerichteten Graphen, die keine Schleifen enthalten
- Petri-Netze (z.B. Fraunhofer Resource Grid GWorkflowDL): Hier werden Aktivitäten als Transitionen und Daten als Token in Stellen modelliert. Jede Transition hat eine Menge von Eingabe- und Ausgabe-Stellen. Eine Transition 'feuert', wenn sich in allen Eingabestellen Token befinden; anschließend werden alle Ausgabestellen mit Token belegt. Petri-Netze in ihrer ursprünglichen Form sind gleich mächtig wie endliche Automaten

Kurz zu nennen sind noch:

- UML (Unified Modeling Language), hauptsächlich der Sprachbestandteil der Aktivitätsdiagramme
- BPMN (Business Process Modeling Notation) ist vor allem interessant wegen der Möglichkeit, die Graphen in ein BPEL-Format abzubilden
- Endliche Automaten, ein mathematisches Modell

Anzufügen ist noch, dass bislang von zwei Fällen bekannt ist, dass einer der obigen Standards (BPEL) für Workflow-Beschreibung produktiv in einem Grid-Umfeld verwendet wurde: die britische Open Middleware Infrastructure Initiative (OMII) und das US-amerikanische Projekt caGrid. Bei beiden wird die ActiveBPEL Engine als Workflow-Enactor verwendet.

2.7Datengrid

Zur Erstellung eines Datengrid werden unterschiedlichste Aufgaben und Funktionalitäten gebraucht, vom Datentransfer, der schon relativ tief in den Systemfunktionalitäten angesiedelt ist, bis hinauf zur Zuweisung von Speicherplatz auf einer eher strategischen Ebene. Diese unterschiedlichen Aufgaben und Funktionalitäten in einem Datengrid werden im Sinne einer modularen Architektur zumeist in einzelnen Komponenten gekapselt, die untereinander verknüpft sind. Man kann hierbei unterscheiden zwischen den Bereichen: Datentransfer, Datenmanagement, und Information Services. Diese drei Bereiche beinhalten wiederum einzelne Aufgaben und Funktionalitäten.

Der *Datentransfer* muss robust, möglichst effizient und sicher sein. Verschiedene Grid Middleware Systeme haben unterschiedliche Umsetzungen für den Datentransfer: die entsprechende Komponente im Globus Toolkit heisst Reliable File Transfer (RFT), die von gLite File Transfer Service (FTS), und OGSA-DAI kann mit der Data Transport Komponente direkt auf Datenbanken zugreifen (siehe Kapitel Grid Software Pakete in den jeweiligen Kapiteln).

Der Bereich *Datenmanagement* hat ein breiteres Aufgabenspektrum. Zu den Aufgaben zählen die grundlegende Registrierung der vorhandenen Daten im Grid und die Virtualisierung der verteilten Daten, sowie Replica Management.

Information Services setzen auf diesen beiden Bereichen auf und sind auf einer vergleichsweise hohen Abstraktionsebene angesiedelt. Eine Information Service Komponente könnte die Zuteilung von Speicherplatz - welche Daten werden wo gespeichert und, wenn sie repliziert werden, wo repliziert? Je nach Systemarchitektur könnten diese Aufgaben direkt in der Applikation abgewickelt werden, mit dem Datenmanagement verschwimmen, oder in einer separaten Komponente verwaltet werden. Ein Information Service, das zumeist in einer separaten Komponente gekapselt ist, ist das Metadatenmanagement. Metadatenmanagement ist bereits relativ Anwendungs- bzw Community-spezifisch, und es gibt die unterschiedlichsten Ansätze und Softwarepakete dazu. Neben den Metadatenmanagement Komponenten von speziellen Grid Middleware Systemen, wie zB das Metadata Catalog Service (MCS) in Globus, kann gerade auch das Metadatenmanagement unabhängig von einer Grid Middleware gehandhabt werden aber eng damit verknüpft sein.

TextGrid's Datengrid-Strategie wird auf 3 Kernfaktoren basieren:

- eine Schnittstelle zu Archiven. Archive mit relevanten textwissenschaftlichen Dokumenten sind unabhängige Organisationen. Eingriffe in deren Systeme sind nur sehr begrenzt denkbar. Eine Schnittstelle muss also minimal invasiv sein, aber doch weit genug gehen, sodass das verteilte TextGrid Datengrid darauf aufgebaut werden kann.
- TextGrid Daten (meist dynamische Daten) werden möglichst direkt mit den Tools der Middleware verwaltet.
- ein übergreifendes Metadatenmanagement, das möglichst effizienten Zugang zu den heterogenen Archiven und den TextGrid Daten ermöglicht.

Die Schnittstelle zu Archiven wird in einer spezialisierten Arbeitsgruppe in Zusammenarbeit mit kooperierenden Archiven spezifiziert. Datengrid Komponenten, separat oder als Teil von Grid Middleware Systemen, werden im folgenden Kapitel besprochen. Metadatenmanagement wird im

nächsten Kapitel ebenfalls bei den jeweiligen Grid Middleware Systemen und noch einmal separat besprochenen.

3.Grid Software Pakete

Es gibt eine Vielzahl von Grid Software Paketen, die bestrebt sind, die in den vorherigen Abschnitten beschriebenen Standards zu implementieren, um beim Aufbau von Grids behilflich zu sein. Grundsätzlich können zwei Arten von Softwarepaketen unterschieden werden: zum einen gibt es Low-Level-Software-Pakete, die lediglich Grid-Standards wie WSRF implementieren, und zum anderen gibt es High-Level-Software-Pakete, die auf den ersteren aufbauen und dem Applikationsentwickler eine vereinfachte Schnittstelle zu den Low-Level-Software-Paketen anbieten. Dementsprechend werden in den folgenden Abschnitten zuerst Low-Level-Software-Pakete wie Globus-Toolkit-4 und gLite beschrieben und darauf aufbauend High-Level-Software-Pakete wie "Java CoG Kit", GAT und SAGA.

3.1Globus Toolkit 4 (GT4)

Globus Toolkit 4 ist ein Grid-Software-Paket, welches von seinen Unterstützern gerne als "great Grid enabler" bezeichnet wird. Es beinhaltet eine Implementierung von WSRF/WSN und bietet darauf aufbauend Low-Level-Software-Komponenten - unter anderem in Form von Web-Services - an, die verschiedene Aspekte von Grids umzusetzen helfen. Diese Komponenten lassen sich in die folgenden Bereiche unterteilen: Sicherheit, Datenmanagement (behandelt in Grids vorkommende Daten), Executionmanagement (behandelt in Grids vorkommende Programme) und Informationsdienste (behandelt in Grids vorkommende Informationen über Daten/Programme). Eine weitere Komponente ist die "Common Runtime", die sprachgebundene Low-Level-APIs zu diesen Komponenten anbietet. Im Folgenden wird GT4 anhand dieser Komponenten beschrieben.

3.1.1 Security

Die Sicherheitsinfrastruktur von GT beruht auf GSI (2.5.1.8), welches im Rahmen von Globus konzipiert und implementiert wurde. Authentifizierung geschieht über X.509-Zertifikate, bzw. davon abgeleitete Proxy-Zertifikate. Diese Authentifizierung wird auch zur Autorisierung verwendet. Hierfür wird auf den lokalen Ressourcen eine sog. gridmap-Dateigepflegt, in der vom SubjectDN des Zertifikats, welcher den Benutzer eindeutig identifiziert, auf einen lokalen User-Accounts gemapt wird, und die Zugriffsregeln auf diese lokalen Accounts wirken. Dieses aus der frühen Zeit des Grid-Computing stammende Konzept kann aber nicht skalieren, weshalb neue Wege bezüglich Authentifizierung und Autorisierung begangen werden mussten.

Ein Ergebnis der neuen Bemühungen um Grid-Security ist das Pojekt GridShib, welches GT4 mit Shibboleth-Mechanismen anreichert, um den durch Shibboleth gegebenen föderierten Ansatz zum Identitätsmanagement im Grid implementieren zu können. Hiermit muss nicht mehr auf jeder Ressource ein Benutzermapping durchgeführt werden, sondern die Benutzer können

sich lokal in ihrer Heimatorganisation authentifizieren und diese Authentifizierungsinformation, aber auch Autorisierungsattribute über SAML-Assertions werden dann vom Identity Provider (IdP) der Heimatorganisation an die Ressource übermittelt. Der hierzu notwendige Vertrauensrahmen wird in der Föderation über Verträge hergestellt. GridShib integriert diese Vorteile unter weitestgehender Beibehaltung der X.509-Proxy-Zertifikatsbasierten GSI, die Infrastruktur selbst muss also nicht geändert werden.

Hierzu werden kurzlebige X.509 Zertifikate (SLC: Short Lived Certificate) durch eine shibbolethisierte Online-CA ausgestellt. Der Ressourcenprovider (Service Provider) erhält Autorisierungsattribute vom IdP über Pull- oder Push-Mechanismen.

3.1.2 Data Management

Die Datenmanagement-Komponente von GT4 besteht aus Diensten, mit denen verteilte Daten lokalisiert, übertragen und verwaltet werden können.

Der GridFTP-Dienst erweitert das bekannte FTP-Protokoll und dient dem schnellen Übertragen großer Datenmengen. GridFTP

- verwendet die Grid Security Infrastructure sowohl in Kontroll- und Daten-Kanälen für Authentifizierung und Autorisierung,
- erlaubt Dateiübertragungen zwischen zwei Parteien, die von einer dritten Partei initiiert und überwacht werden (third-party-transfer),
- erlaubt parallele Datenübertragungen, indem mehrere TCP Streams verwendet werden,
- erlaubt die Übertragung von Teilbereichen einer Datei und
- erlaubt automatisches Ermitteln der optimalen Größe für TCP-Puffer/Fenster und Unterstützung für verläßliche Dateiübertragungen.

Der Reliable-File-Transfer-Dienst RFT hat dieselbe Grundfunktionalität wie GridFTP, bietet aber darüberhinaus job-scheduling-ähnliche Funktionalitäten für die Überwachung der Übertragung von Dateien und Verzeichnissen und erlaubt, diese Überwachung WSRF-konform durchzuführen. Für die Übertragung großer Datenmengen verwendet RFT intern eine Datenbank, die Informationen über die aktuelle Übertragung in regelmäßigen Abständen zwischenspeichert, so daß bei Fehlübertragungen nur fehlerhaft übertragene Teile neu übertragen werden müssen im Gegensatz zu einer vollständigen Neuübertragung.

Die Dienste Replica-Location-Service (RLS) und Data-Replication-Service (DRS) ermöglichen das Verwalten von replizierten Dateien eines Grids.

Der Data-Access-and-Integration-Dienst OGSA-DAI von GT4 ist ein Java-Framework für Zugriffe und Integration von Datenquellen (z.B. Dateien, relationale und XML-Datenbanken) in Gridumgebungen. Datenzugriffe (z.B. SQL-Abfragen für relationale Ressourcen, XPath-Anweisungen für XML-Daten) sind dabei über Web-Service-Schnittstellen möglich.

In TextGrid können die Dienste RLS und DRS dazu verwendet werden, Replikas von Wörterbüchern oder Teile von Archiven zu verwalten; mit RFT können Dateien aus Performancegründen in die Nähe

von den sie verarbeitenden Werkzeugen gebracht werden, und mit OGSA-DAI kann eine Anbindung an die Archive von TextGrid realisiert werden.

3.1.3 Execution Management

Für die Verwaltung von Programmen (oder Jobs) - im Gegensatz zu Web-Services - eines Grids implementiert GT4 eine Schnittstelle namens GRAM (Grid Resource Allocation and Management). GRAM stellt Dienste zum Veröffentlichen, Ausführen, Überwachen und Abbrechen von Grid-Anwendungen (jobs) auf verteilten Grid-Ressourcen zur Verfügung. GRAM ist selbst kein Job-Scheduler, stellt aber Protokolle für die Kommunikation mit lokalen Schedulern wie PBS, Condor, LSF, SGE zur Verfügung.

Da TextGrid primär ein Data-Grid ist, welches Web-Services im Gegensatz zu Jobs verwendet, wird das für ein Computing-Grid nützliche Execution-Management von GT4 zunächst nicht benötigt.

3.1.4 Information Services

Die Informationsdienste (alternativ auch bezeichnet als MDS = Monitoring and Discovery System) von GT4 sind eine Sammlung von WSRF-Web-Services, mit denen Ressourcen eines Grids überwacht und entdeckt werden können. Der Index-Service ist ein solcher Informationsdienst, der Informationen von Grid-Ressourcen sammelt und an einer zentralen Stelle veröffentlicht. Index-Services können Informationen von anderen Index-Services sammeln, sodaß hierarchisch angeordnete Informationsdienste aufgebaut werden können. Der Trigger-Service ist eine Erweiterung des Index-Service insofern, als er das Ausführen von Aktionen erlaubt, wenn konfigurierbare Bedingungen bzgl. der gesammelten Informationen eintreten (z.B. if "harddisc is full" then "send email to admin"). Der WebMDS-Dienst ist eine Web-Browser-basierte grafische Benutzerschnittstelle zum Index-Service oder Trigger-Service. All diese Dienste sind auf einer gemeinsamen Basis implementiert, dem Aggregator Framework von GT4. Dabei handelt es sich um ein Framework, welches Schnittstellen für Datenquellen (sources) und für Datensenken (sinks) spezifiziert und einen Mechanismus bereitstellt, mit dem die Datensenken Informationen aus den Datenquellen extrahieren können. Dieser Mechanismus schließt ResourceProperty-Abfragen - wie sie in WSRF spezifiziert sind - der Datensenken bzgl. den Datenquellen und Subscription/Notification-Anfragen - wie sie in WSN spezifiziert sind - mit ein. Datenquellen und Datensenken können dabei WSRF-Web-Services oder beliebige Programme sein. Beispielsweise sind der Index-Service und der Trigger-Service von GT4 als Datensenken implementiert. Mit diesem Framework können proprietäre Datenquellen und Datensenken in ein Grid integriert werden.

In TextGrid kann der MDS-Index-Service anstelle von UDDI verwendet werden, um die Instanzen der community-spezifischen Werkzeuge an einer zentralen Stelle zu verwalten und beispielsweise dem Workflow-Editor zur Verfügung zu stellen.

3.1.5 Common Runtime Components

GT4 stellt dem Applikationsentwickler Bibliotheken, Werkzeuge und APIs zur Verfügung, um die in den vorherigen Abschnitten beschriebenen Dienste zu verwenden. Momentan werden die Sprachen Python, C und Java unterstützt.

3.2gLite

gLite ist ein Grid Middleware Paket, das - ähnlich wie Globus - alle Aspekte des Grid Computing adressiert. Es wird im Rahmen des Europäischen Projektes EGEE (Enabling Grids for E-sciencE, http://public.eu-egee.org/intro/) entwickelt, das ursprünglich aus der Hochenergiephysik stammt und starke Bande mit dem Teilchenbeschleuniger CERN (www.cern.ch) hat.

Die gLite Software hat sich aus dem LCG (Large Hydron Collider Computing Grid, http://lcg.web.cern.ch/LCG/) entwickelt, welche wiederum auf Globus 2.4 basiert. Mit der Ergänzung durch zusätzliche Komponenten und der Migration zu WSRF soll eine moderne, "vollständige" Umgebung für Grid Computing geschaffen werden.

Innerhalb von D-Grid setzt nur HEPCG ausschliesslich gLite ein. Alle anderen verwenden ausschliesslich Globus, oder betrachten Globus als ihre wichtigste Middleware Infrastruktur. Auch ausserhalb von D-Grid spannt Globus eine wesentlich grössere Zielgruppe und Entwickler Community auf als gLite, trotz der grossen Unterstützung durch Europäische Förderungen.

Vor allem die "saubere", standard-orientierte Architektur von gLite ist spannend. Durch die starke Ausrichtung auf Hochenergie-Physik und die "harten" Wissenschaften, bedient gLite allerdings meist andere Anforderungen als die von TextGrid. So wird die Verteilung von umfangreichen Compute Jobs unterstützt, und die hoch-effiziente Speicherung und Verteilung von unveränderlichen Daten - was beides nicht unbedingt Kernbereiche von TextGrid, resp an den Anforderung zur Haltung von dynamischen, eher kleinen Datenmengen vorbei geht. Dies heisst nicht, dass TextGrid in gLite *nicht* umgesetzt werden könnte, und ist alleine noch nicht Grund genug zur Verwendung von Globus. Allerdings gibt die grössere Community von Globus den Ausschlag. Spezifische Komponenten wie zB GridShib (http://gridshib.globus.org/), die Umsetzung der AAI Shibboleth für das Grid, werden vornehmlich für Globus entwickelt. Obwohl eine spätere Verfügbarkeit in Globus wie auch gLite denkbar ist, ist für TextGrid die unmittelbare Verfügbarkeit, sowie die in der Community verfügbare Erfahrung und möglicher Support Ausschlag gebend. Insofern ist aus TextGrid Sicht Globus anfänglich gLite vorzuziehen. Da aber beide, Globus und gLite, durch WSRF und spezifische Standards konvergieren (siehe hierzu auch die Grid Interoperability Now, GIN, Inititative der GGF) könnte eine Migration der TextGrid Service Layer von Globus auf gLite, bzw eine parallele und transparente Nutzung beider Middleware Systeme zu einem späteren Zeitpunkt möglich sein, sollte sich das als sinnvoll herausstellen.

3.3Java CoG Kit

Das Java Commodity Grid Kit (Java CoG Kit) ist eine Softwarekomponente, die es Gridnutzern, Gridapplikationsentwicklern und Gridadministratoren erlaubt, Grids von einem höheren Standpunkt aus zu benutzen, zu programmieren und zu administrieren. Das Java CoG Kit beinhaltet unter anderem folgende Komponenten:

- das Karajan Workflow Framework, welches eine Workflowsprache und eine Workflow-Engine beinhaltet
- Unterstützung für Grid Portale
- ein Grid-Desktop, welches den nativen Desktop des Betriebssystems um Gridkonzepte erweitert

Neben diesen benutzernahen Komponenten beinhaltet das Java CoG Kit die Java CoG Kit jGlobus Bibliothek, welche ein integraler Bestandteil der Middleware Globus Toolkit ist. Insbesondere sind die javabasierten APIs der Globus-Toolkit-Komponenten Grid-Security-Infrastructure (GSI), gridFTP, myProxy und GRAM-Clients mit Hilfe der jGlobus-Bibliothek implementiert.

3.4Grid Application Toolkit (GAT)

Das Grid Application Toolkit ist eine objektorientierte API, die bestrebt ist, Grid-Applikationsentwicklern einen einfachen (und folglich in einer gewissen Weise eingeschränkten) Zugang zu Grid-Funktionalitäten anzubieten. Dementsprechend bedient GAT nicht jeden möglichen Anwendungsfall, sondern nur allgemeinere und häufiger auftretende Anwendungsfälle, die allerdings die augenblicklichen TextGrid-Anforderungen erfüllen.

3.4.1 Komponenten

Im Folgenden werden die Hauptkomponenten von GAT beschrieben.

3.4.1.1 File Management

Die Dateiverwaltung von GAT erlaubt Applikationsentwicklern, lokale und entfernte Dateien ortsunabhängig und protokollunabhängig zu verwalten. Dateien können erzeugt, gelöscht, kopiert, bewegt und untersucht werden.

3.4.1.2 FileStream Management

Die Dateiverwaltung von GAT behandelt Dateien als Ganzes, ohne auf ihren Inhalt einzugehen. Um den Inhalt einer Datei zu lesen und zu schreiben bietet GAT das Konzept eines FileStreams an. FileStreams können erzeugt und gelöscht werden, aus ihnen können Bytes gelesen werden, in sie können Bytes geschrieben werden, ihr interner Dateizeiger kann beliebig gesetzt werden.

3.4.1.3 LogicalFile Management

Eine logische Datei (LogicalFile) repräsentiert eine Menge von Dateien, die Byte-für-Byte identisch sind und geographisch verteilt sein können. Dies ermöglicht die Verwaltung von Replikas und ein effizientes Zur-Verfügung-Stellen von Dateien. Eine logische Datei kann erzeugt und gelöscht werden; identische physische Dateien können zu der Menge der Dateien, die die logische Datei repräsentiert, hinzugefügt oder entfernt werden; eine logische Datei kann repliziert werden, d.h. es wird eine bzgl. der Entfernung im Netzwerk geeignete Datei aus der Menge ihrer physischen Dateien ausgewählt, und an eine neue Stelle kopiert; eine logische Datei kann untersucht werden bzgl. ihrer Menge von physischen Dateien.

3.4.1.4 Advert Management

GAT stellt einen Informationsdienst namens AdvertService bereit, um Gridobjekte (Advertisables genannt) zu speichern und zu veröffentlichen. Advertisables können Strings, Referenzen auf Objekte, Jobs, (logische) Dateien und Ressourcen sein. Eine Referenz auf eine bestehende Instanz eines AdvertService kann angefordert und wieder fallengelassen werden. Ein Advertisable kann im AdvertService veröffentlicht werden, indem das Advertisable beschreibende Metadaten mitgegeben werden. Im AdvertService veröffentlichte Advertisables können über die sie beschreibenden Metadaten wiedergefunden werden.

3.4.1.5 Resource Management

GAT kennt Software-Ressourcen (z.B. Programme oder Betriebssysteme) und Hardware-Ressourcen (z.B. PalmPilots). Diese Ressourcen werden von einem ResourceBroker insofern verwaltet, als er sie finden und für einen Clienten reservieren kann. Eine Referenz auf einen ResourceBroker kann angefordert und wieder fallengelassen werden. Ressourcen werden über sogenannte ResourceDescriptions beschrieben, speziell über SoftwareResourceDescriptions und HardwareResourceDescriptions. Mit diesen ResourceDescriptions kann der ResourceBroker Anfragen beantworten und auf diesem Wege gefundene Ressourcen zurückgeben oder reservieren.

3.4.1.6 Interprocess Communication

Mit GAT können Prozesse eines Grid miteinander kommunizieren. Dazu veröffentlicht ein Prozess A eine Referenz (einen Endpoint) auf sich selbst in einem AdvertService und hört auf eingehende Verbindungen mittels einer Listen-Operation. Der veröffentlichte Endpoint kann von einem Prozess B über den AdvertService gefunden werden und zum Aufbau einer Verbindung mit dem Prozess A mittels einer Connect-Operation verwendet werden. Nach dem Zustandekommen der Verbindung können die beiden Prozesse mittels Read- und Write-Operationen Informationen untereinander austauschen.

3.4.1.7 Job Management

GAT unterstützt die Verwaltung von Jobs. Jobs sind Prozesse, die auf Gridressourcen ausgeführt werden. Dementsprechend wird ein Job durch zwei Komponenten beschrieben, zum einen durch eine SoftwareDescription, welche die Software beschreibt, die den Job ausführt, und zum anderen durch eine ResourceDescription, welche die Hardware beschreibt, auf welcher der Job ausgeführt werden soll. Eine SoftwareDescription und eine ResourceDescription ergeben zusammen eine JobDescription. Um einen Job zu erzeugen, wird die ihn beschreibende JobDescription einem ResourceBroker übergeben und mit der SubmitJob-Operation gestartet. Die Destroy-Operation beendet den Job. Jobs können untersucht werden: die GetJobDescription-Operation gibt die JobDescription des Jobs zurück und die GetState-Operation liefert seinen aktuellen Zustand (initial, scheduled, running, ...). Ein Job kann mit der Stop-Operation angehalten werden. Mit der Checkpoint-Operation kann der aktuelle Zustand eines Jobs zwischengespeichert werden. Ein Job kann mit der CloneJob-Operation vervielfältigt werden. Mit der Migrate-Operation kann ein Job auf eine andere Ressource "umziehen".

3.4.1.8 Monitoring

GAT erlaubt, Gridobjekte wie (logische) Dateien, FileStreams, Ressourcen, Jobs, etc. zu überwachen. Ein solches Monitorable genanntes Gridobjekt veröffentlicht eine Menge von Metrics genannten meßbaren Größen. Beim Eintreten bestimmter Bedingungen sendet ein Monitorable ein MetricEvent genanntes Ereignis, welches einen konkreten Wert für eine meßbare Größe beinhaltet. Dieses Ereignis kann dann von zuvor angemeldeten Interessenten empfangen und weiterverarbeitet werden.

3.4.2 Implementierungen

In den folgenden Abschnitten wird beschrieben, welcher Mechanismus einer jeden Implementierung der GAT-API zugrunde liegt und in welchen Programmiersprachen GAT eingesetzt werden kann.

3.4.2.1 Adaptoren

GAT ist, wie in den vorherigen Abschnitten beschrieben, lediglich eine abstrakte Spezifikation einer API. Um diese API als Applikationsentwickler einsetzen zu können, bedarf es einer konkreten Realisierung dieser API, d.h. es müssen Vorkehrungen getroffen werden, die gewährleisten, daß beispielsweise die Operationen der Dateiverwaltung von GAT effektiv durchgeführt werden und ihre Wirkung entfalten. Kandidaten für eine konkrete Realisierung solcher Operationen sind etablierte Grid-Middleware-Software-Pakete wie Globus Toolkit 4 oder gLite. Da die Programmierschnittstelle solcher Software-Pakete mit derjenigen von GAT nicht übereinstimmt, muß eine Anpassung dieser verschiedenen Schnittstellen vorgenommen werden. GAT bedient sich des bekannten Adaptor-Entwurfsmusters, um eine solche Anpassung durchzuführen. So gibt es Adaptoren für viele Komponenten von Globus Toolkit 4 und (geplante) Adaptoren für gLite. Kurz gesagt brechen diese Adaptoren die High-Level-Operationen von GAT auf die Low-Level-Operationen einer bestimmten Grid-Middleware herunter. Eine Folge des Einsatzes von Adaptoren ist, daß der Applikationsentwickler, der GAT einsetzt, unabhängig von den konkreten Details der Schnittstellen von Grid-Middleware-Software-Paketen ist, von Änderungen in diesen Softwarepaketen abgeschirmt wird und durch den Austausch von Adaptoren verschieden leistungsfähige (und zukünftige) Grid-Middleware ansprechen kann.

GAT könnte im Rahmen von TextGrid wie folgt eingesetzt werden: Die GAT-Komponenten File-Management, FileStream-Management und LogicalFile-Management könnten helfen, den DataGrid-Anteil von TextGrid umzusetzen; das Advert-Management von GAT könnte als Informationsdienst dazu verwendet werden, TextGrid-spezifische Informationen - wie z.B. Informationen über Archive - an einer zentralen Stelle zu veröffentlichen und den TextGrid-Werkzeugen zur Verfügung zu stellen.

3.4.2.2 sprachgebundene Realisierungen der GAT-API

Ein Applikationsentwickler kann GAT in den Sprachen C, C++ und Java einsetzen. Die Referenzimplementierung von GAT ist in C geschrieben. Es existiert ein C++-Wrapper um die C-Implementierung herum, mit dem die objektorientierte GAT-API mit Hilfe der objektorientierten Sprachkonstrukte von C++ direkt angesprochen werden kann. Eine von der C/C++-Implementierung

abgesonderte Implementierung der GAT-API ist JavaGAT (in Java). Eine Folge dieser verschiedenen Sprachbindungen ist leider, daß Adaptoren für jede Sprache separat geschrieben werden müssen.

3.5Simple API For Grid Applications (SAGA)

SAGA ist ebenso wie GAT eine einfache API für Grid-Anwendungen, die sich gerade noch in der Entstehungsphase befindet. Für den Entwurf von SAGA sind zahlreichere Anwendungsfälle als für GAT herangezogen worden, wodurch eine breitere Schicht von Grid-Anwendungen bedient werden soll. Trotzdem sind SAGA und GAT sehr ähnlich, was auch die Bezeichnung GAT2 für SAGA rechtfertigt. Um eine kurze Beschreibung der gerade im Entstehen und Fluss begriffenen SAGA-API zu geben, konzentrieren wir uns im Folgenden auf die Unterschiede zu GAT.

SAGA und GAT bieten beide eine Unterstützung, ähnliche Konzepte und eine ähnliche API für die Bereiche Dateien, Streams, logische Dateien, Jobs und Monitoring (bzgl. Monitoring bietet SAGA allerdings auch schreibbare Monitorables an im Gegensatz zu den nur lesbaren Monitorables von GAT). Die Bereiche Session und Task sind dagegen gegenüber GAT neu hinzugekommen. Session ist ein Mechanismus, mit dem SAGA-Objekte in unabhängigen Mengen zusammengefaßt werden können, und so bzgl. Sicherheit oder Lebenszeit unterschiedlich behandelt werden können. Tasks erlauben SAGA-Operationen, asynchron ausgeführt zu werden. Dabei repräsentiert jeder Task eine asynchrone Version einer SAGA-API-Methode.

Für die Verwendung von SAGA in TextGrid ist zu sagen, dass momentan erste Implementierungen von SAGA in den Sprachen C++ und Python entstehen, die ihre Stabilität erst noch gewinnen und beweisen müssen. Da die SAGA-API selbst noch im Entstehen und Änderungen unterworfen ist, kann sich TextGrid leider noch nicht auf eine in Stein gemeißelte ausgereifte API verlassen. Deshalb wird die Strategie verfolgt, vorläufig eine eigene an SAGA angelehnte TextGrid-API zu entwerfen, die in Zukunft gegen SAGA ausgetauscht werden könnte.

3.6Portale

Ein Portal ist eine Alternative zur Verwendung von Grid-Diensten über die Kommandozeile oder als API. Der Benutzer braucht dazu lediglich einen Web-Browser, einen Account im Portal sowie die für das jeweilige Grid benötigten PKI-Zertifikate. Die momentan am häufigsten in den verschiedenen Grids eingesetzte Portalsoftware ist GridSphere, gegebenenfalls mit diversen Erweiterungen.

GridSphere, wie GAT aus dem EU-Projekt GridLab hervorgegangen, ist zunächst ein nicht speziell auf Grid-Belange zugeschnittenes, universell einsetzbares Portal. Einzelne Komponenten des Portals werden als *Portlets* -- stadardisiert als JSR 168 -- dem Benutzer als kleinere Unterfenster sichtbar. Portlets sind im Portal innerhalb von Portlet-Gruppen bzw. den aus Formularen bekannten Karteireitern organisiert. Ein Portlet implementiert jeweils eine bestimmte Funktionalität, z.B. gibt es E-Mail-Portlets, RSS-Portlets oder Portlets, die wiederum eine Webseite darstellen können. Nachdem sich der Benutzer mit seinen Accountdaten im Portal angemeldet hat, stehen seine Authentifizierungsdaten allen Portlets zur Verfügung (*Single-Sign-On*).

Die Grid-Funktionalitäten in GridSphere sind über entsprechende Portlets implementiert, die separat installiert werden können. Im Katalog der vom GridSphere-Team angebotenen GridPortlets sind enthalten:

- ein Credential Management zum Verwalten der Benutzerzertifikate
- ein Resource Browser für allgemeine Ressourcen, Services und Jobs
- ein File Browser zur Verwaltung von (entfernten) Dateisystemen per GridFTP
- ein Job Portlet zur Anzeige und zum Abschicken von Jobs per GRAM
- eine Benutzerschnittstelle zu WebMDS (Paket gt4portlets)

Beim GridPort-Projekt können zusätzliche Portlets für GridSphere bezogen werden:

- Condor Job Submission Portlet
- SDSC Storage Resource Broker (SRB) Portlet
- weitere Portlets mit ähnlicher Funktionalität wie die bereits erwähnten GridPortlets

In verschiedenen Grid-Installationen wurden diese Basis-Dienste, die von GridSphere direkt angeboten werden, um weitere Funktionalitäten erweitert. Zu nennen ist hier beispielsweise das P-Grade-Portal, das auf GridSphere basiert. Hier wurden zusätzlich Portlets zur Visualisierung der Auslastung von Ressourcen und zur Verwaltung von Workflows entwickelt. Über letztere kann u.A. ein interaktiver Workflow-Editor als Java-Applet mittels Java Web Start geöffnet werden. Die bearbeiteten Workflows können dann zum Portal hochgeladen, dort validiert und ausgeführt werden.

3.7Workflow-Komponenten

3.7.1 Kriterien

Hier sollen die wichtigsten Kriterien aufgeführt werden, nach denen Workflow-Systeme für TextGrid nachfolgend zu bewerten sind. Im Einzelnen:

- Software muss Open-Source sein, damit für TextGrid Modifikationen gemacht werden können
- Trennung zwischen "Editor" (dem Workflow-Kompositionstool/GUI) und "Enactor" (dem Workflow-Manager) ermöglicht durch
  - geeignete Workflow-Beschreibungssprachen (möglichst Standards wie WS-BPEL, zumindest XML)
  - speziellen "Deploy"-Mechanismus für auszuführende Workflows
  - separates (Web-)Interface zum Enactor, mit dem die Ausführung von laufenden Workflows überwacht werden kann
- Information Retrieval: der Editor sollte mit Metadaten umgehen können, d.h. vor allem mit Metadaten über Dateien im Grid, also z.B. Informationen vom Globus Toolkit MDS4 Index Service oder UDDI.
- Einbindung: die Software muss als Eclipse-Plugin lauffähig sein, sowohl für den Editor, als auch für das Administrationsinterface des Enactors. Idealerweise wäre auch ein Web-Interface (über ein Grid-Portal o.Ä.) wünschenswert; d.h. der Editor wäre etwa auch als Java-Applet verfügbar.
- Der Enactor muss unbedingt Web Services starten und überwachen können, für TextGrid Architektur V.2 wären zusätzlich WSRF Services wünschenswert.
- Der Editor muss intuitiv bedienbar sein.

Weitere Kriterien sind in [Yu/Buyya2005] genannt und können für eine feinere Klassifikation eingesetzt werden (vgl. dort).

3.7.2 Analyse von Workflow-Systemen

- **Intalio|BPMS** (http://bpms.intalio.com): Die Software ist leider nicht quelloffen, sonst allerdings vorbildlich, was die Architektur angeht: eine strikte Trennung zwischen Editor ("Designer") und Enactor ("PXE 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 PXE-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, Implementierbarkeit als Applet fraglich
  - Web Services: können gestartet werden da WS-BPEL
- **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. Momentan (Version 2.1.0, Nov. 2006) wird WS-BPEL 2.0 nur in Teilen unterstützt, sonst BPEL4WS 1.1. 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, für den von Eclipse aus ein externer Internet-Browser gestartet wird
  - Einbindung: der Editor ist Eclipse-Plugin, Implementierung als Applet nicht geplant; zur Administration der Engine muss auf ein Web-Interface zurückgegriffen werden.
  - Web Services: können gestartet werden, da BPEL
- **P-Grade Portal** (http://www.lpds.sztaki.hu/pgportal/v23) P-Grade ist eine graphische Umgebung zum Erstellen und zur Ausführung und Überwachung von Workflows in verschiedenen Grids. Das P-Grade *Portal* ist eine auf GridSphere basierende Implementierung mit der gleichen Funktionalität, aber mittels Portlets bzw. Applets. Der Workflow-Editor kann als lokales Applet über Java WebStart einfach und komfortabel heruntergeladen werden, die Bedienung ist intuitiv. Nach Erstellung kann der Workflow dem Portal Server übergeben werden. Alle Workflows liegen auf dem Portal Server und können von dort aus gestartet, überwacht und auch zur weiteren Modifikation im Applet heruntergeladen werden. Weitere Merkmale:
  - Open Source: Das gesamte Portal ist weder als Open noch als Closed Source verfügbar - die Entwickler fordern einen entsprechend konfigurierten Rechner, auf dem sie dann selbst das Portal installieren

    - Web Services: Jobs müssen als Executable vorliegen und werden zum Portal und von dort aus zum gewünschten Grid-Rechner hochgeladen. Momentan gibt es (noch, Juli 2006) keine Möglichkeit, Web Services auszuführen und zu überwachen
    - Einbindung: als Portlet bzw. Applet realisiert, Eclipse-Plugin unbekannt, da Code nicht Open Source
    - Information Retrieval: UDDI nicht unterstützt, MDS zur Registrierung und Überwachung von Ressourcen
    - Format der Workflow-Spezifikation: textuell, nicht-XML
- **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, für Applet untauglich da zu groß und zu viele Abhängigkeiten von Bibliotheken (lt. Triana-Entwicklern)
    - 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 wird in keiner Version unterstützt
    - Open Source: ja
    - Einbindung: Eclipse-Plugin fraglich, Applet fraglich (zu groß und zu viele Abhängikeiten von diversen Bibliotheken)
    - Format der Workflow-Spezifikation: proprietäres 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 und Applet wahrscheinlich realisierbar, das Tool ist relativ klein und kompakt. Integration in GridSphere theoretisch machbar, da die GridSphere "GridPortlets" auch das Java CoG Kit verwenden. (Bei der OGCE war offensichtlich auch ein Karajan-Portlet für GridSphere geplant, allerdings ist der Status seit über einem Jahr noch auf "coming soon".)
    - 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 (die neueste Dokumentation im Juli 2006 ist vom Januar 2006) 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).
- **GWES** (http://www.gridworkflow.org/kwfgrid/gwes/docs/): GWES ist ein Projekt des Fraunhofer FIRST. Das System wird im K-WF-Grid, InstantGrid und MediGrid eingesetzt, vormals im Fraunhofer Resource Grid (FhRG). Es basiert auf Petri-Netzen, bei denen Aktivitäten mit Transitionen und Daten als Token in Stellen modelliert werden. Es gibt eine klare Trennung zwischen der Enactor-Engine (GWES, Grid Workflow Execution Service), und graphischem Client (GWUI, Grid Workflow User Interface). Die Engine läuft als Web Service, der Workflows in der Open-Source-XML-Datenbank eXist speichert. Das Client-Applet bietet momentan (Sep'06) nur eingeschränkte Editiermöglichkeiten, soll aber demnächst zu einem vollen Workflow-Editor ausgebaut werden. Es gibt neben dem GWUI auch einen Kommandozeilen-Client. Ein AJAX-Client ist ebenfalls angedacht; diese neue Technologie könnte eine einfachere Integration des Clients im Browser bedeuten.
    - Open Source: ja, aber freie Lizenz nur für nicht-kommerziellen Einsatz

  - Einbindung: GWUI ist Applet, das im Browser (direkt oder als GridSphere-Portlet) läuft; Integration in Eclipse "prinzipiell möglich" (laut Entwickler)
  - Format der Workflow-Spezifikation: proprietär, aber offen und gut dokumentiert (GWorkflowDL, Grid Workflow Description Language), eine XML-Repräsentation für Petri-Netze.
  - Web Services: werden unterstützt, sowie RFT und WS-GRAM Jobs
  - Information Retrieval: Zitat (E-Mail des Chef-Entwicklers Andreas Hoheisel vom 29.8.2006): *"Das Konzept sieht vor, dass beliebige "WorkflowHandler" implementiert werden können, die unterschiedliche Information-Retrieval-Methoden unterstützen können. Bisher implementiert (für FhRG, Instant-Grid, Medi-Grid und K-Wf Grid) ist der "KWfGridWorkflowHandler" welcher entweder die Informationen über K-Wf Grid-spezifische Dienste holt oder aber aus einer XML-Datenbank. Dort werden Resourcenbeschreibungen als D-GRDL-Dokumente (entwickelt im Rahmen von DGI) abgelegt. Es gibt einen "ResourceMatcher" der abstrakte Ressourcen auf konkrete abbildet. UDDI verwenden wir derzeit nicht (sollte aber einfach zu implementieren sein, ist aber nicht mehr state-of-the-art), MDS-4 wird im Rahmen von Instant-Grid auf D-GRDL abgebildet. OGSA-DAI könnte man als externen Dienst ansprechen (z.B. in GRIA realisiert). RFT (ehemals GridFTP) wird unterstützt."*
- **Fazit der Evaluation**: Von den Merkmalen her, die wir in TextGrid benötigen, sind OMII-BPEL und GWES am ehesten geeignet. Hauptvorteil bei OMII-BPEL ist, dass die Integration in Eclipse nativ schon vorhanden ist, während man für die Eclipse-Integration des GWES noch einen weiteren Client implementieren müsste. Vorteile des GWES sind die Integration in das Gridsphere-Portal samt graphischen Client, die schon weit vorangekommen ist, und die konzeptionelle Einfachkeit des Formalismus (Petri-Netze) sowie die daraus resultierende intuitive Bedienbarkeit des Clients. Für beide Systeme gilt, dass sie Open Source sind, es gibt eine Trennung zwischen Enactor und Editor und Web Services werden unterstützt. Wegen der zwingenden Installation von OMII Client und Server bei OMII-BPEL könnte dieses System sich noch als nicht geeignet herausstellen, hierüber wird momentan (Nov2006) noch verhandelt. Der GWES setzt dagegen nur eine Instanz eines Servlet-Containers, z.B. Apache Tomcat, voraus.

3.8Datengrid

Im vorigen Kapitel wurden 3 verschiedene Bereiche eines Datengrid - Datentransfer, Datenmanagement, und Information Services - unterschieden. Diesbezügliche Systeme können eine Komplettserie aus einzelnen Komponenten in umfangreichen Grid Middleware Systemen sein, bereichsübergreifende separate Lösungen, oder spezifische Lösungen für einzelne Bereiche oder Funktionen. Komponenten der Grid Middleware Globus wurde bereits oben besprochen. Diese Section listet zuerst eine Reihe von bereichsübergreifenden separaten Lösungen, die für TextGrid interessant sein könnten, und die auch vom DGI unterstützt werden (siehe dgi.d-grid.de), und geht dann auf spezialisierte Systeme zum Metadatenmanagement ein. Die hierunter genannten Systeme beziehen sich hauptsächlich auf Storage-Verwaltung und Dateimanagement. Ein Mechanismus zur Integration von verteilten Datenbanken, OGSA-DAI, wird im diesbezüglichen Abschnitt besprochen.

Die folgenden 3 kurz beschriebenen Systeme werden - neben den in Globus, gLite und Unicorn integrierten Funktionen - vom DGI unterstützt.

- **DCache** - Disk Cache Mass Storage System, http://www.dcache.org/

    - DCache virtualisiert und verwaltet heterogene Speicherressourcen. Dazu gehört Replicamangement, Load Balancing, etc. DCache wurde von HEP (High Energy Physics) für HEP geschrieben, und hat dementsprechende Eigenschaften:
        - Daten sind unveränderbar;
        - optimiert auf einzelne grosse Files; das Öffnen von Files ist teuer
        - hat eine gLite Tradition
        - gute Anbindung von diversen Storage Systemen
    - *D-Grid Partner*: DCache wird bei HEP angewendet.
- **DataFinder** - http://www.dlr.de/datafinder
    - Mit dem DataFinder können verschiedene File Strukturen auf einander abgebildet werden. Das System baut auf WebDAV, kann aber auch einfach als Katalog verwendet werden (siehe folgendes Kapitel). Der DataFinder wird vom DLR geschrieben und vertrieben, und ist nicht Open Source.
    - *D-Grid Partner*: Der DataFinder wird nach letztem Stand in D-Grid bisher nicht verwendet.
- **SRB** - San Diego Supercomputing Center (SDSC) Storage Resource Broker, http://www.sdsc.edu/srb/
    - Der Storage Resource Broker (SRB) bietet ein einheitliches Interface zu verteilten und heterogenen Storage Servern, und implementiert einen Metadatenkatalog für ein Datei- bzw Objektorientiertes Umfeld.
    - Auch der SRB garantiert keine automatische Konsistenzsicherung wenn Daten modifiziert werden (es wird einfach die Datei mit dem jüngstem Datum als die letzte Version herangezogen; Überprüfung nach manuellem Trigger).
    - Der SRB ist bzgl. der Serverlizenzen problematisch. Kommerzielle Projekte müssen nicht nur Lizenzgebühren bezahlen, sondern auch eine spezielle Distribution verwenden, die zur freien Variante nicht kompatibel ist.
    - *D-Grid Partner*: Der SRB wird bei MediGrid eingesetzt.

Alle drei kurz vorgestellten Pakete - DCache, DataFinder und SRB - haben ihre klaren Stärken in bestimmten Anwendungsgebieten, und haben dort mitunter auch eine breite Community. Keines dieser Pakete kann allerdings die Anforderungen von TextGrid komplett erfüllen: veränderbare Daten bei Datenreplikation, ein weitreichendes Metadatenmanagement, möglichst modular für Minimal-Installationen auf heterogenen Systemen, und eine Open Source Lizenz. Für diese Anforderungen ist eine Kombination aus den Datengrid Modulen in Globus (siehe im Kapitel dort) mit einem spezialisierten Metadaten Management, wie im folgenden Unterkapitel beschrieben, angebracht.

1.8.1 Metadaten Management

Eine für TextGrid zentrale Aufgabe ist das Metadaten Management. Hierbei sollen für jedes in TextGrid gespeicherte Objekt umfangreiche Metadaten gespeichert werden: bibliographische, administrative, sowie Sammlungsmetadaten.

Metadaten Management ist in der Grid Community - wie das Speichermanagement auch - herkunftsbedingt oft an den Anforderungen zur Speicherung von grossen Mengen wissenschaftlicher Daten orientiert. Neben in Grid Middleware Paketen integrierte Systemen, ist zB Tapestry/Chimera

(http://p2p.cs.ucsb.edu/chimera/) ein Datei- bzw Objektorientierte Metadatensystem. Tapestry/Chimera sind Peer-to-Peer overlay Netzwerk Systeme, die über jegliche Infrastruktur gelegt werden können.

Die Abbildung von Metadaten in der geisteswissenschaftlichen Community lehnt sich oft an Modellierungsmethodiken, wie sie von der Dublin Core Metadaten Initiative (speziell im "DCMI Abstract Model") und dem W3C unterstützt werden. Hierbei werden vor allem die Relationen zwischen verschiedenen Einheiten hervorgehoben und in RDF Triples gespeichert. Neben der damit gewonnen Flexibilität ist vor allem auch die Verwendbarkeit der Semantic Web Standards und Tools ein Vorteil für so eine Lösung.

Die Integration von RDF Repositories in eine Grid Umgebung ist ein relativ junges Gebiet mit - nichts desto trotz - einigen aktiven Initiativen. Im direkten Umfeld von TextGrid haben AstroGrid die Vorteile erkannt und bauen ihr Metadatenmanagement auf einem RDF Triple Store (ursprünglich Sesame, http://www.openrdf.org/, [sesame02]) mit der Abfragesprache SPARQL [sparql]. Die Integration von verteilten, heterogenen Triple Stores in einer übergreifenden Schicht ähnlich OGSA-DAI für konventionelle Datenbanken sind auch bereits angelaufen, aber - zum Zeitpunkt des Verfassens dieses Berichts - kaum wenige Wochen alt [OGSA-DAI-RDF]. Noch spannender aus der Motivation einen übergreifenden und performanten Metadatenkatalog zu erstellen sind Ansätze einen RDF Triple Store zu verteilen. Der RDF Triple Store Atlas [atlas] aus dem Projekt OntoGrid (www.ontogrid.net) erreicht dies mit Hilfe eines Distributed Hashtable Algroithmus. Atlas wird primär entwickelt um OntoGrid-spezifische, semantische Beschreibungen von Grid Services zu verwalten, dürfte aber auch für das Metadatenmanagement von Daten angewendet werden können. Entsprechende auf einem peer-to-peer Ansatz basierende Systeme sind in [ding] gegenübergestellt.

Für TextGrid ist derzeit noch nicht klar, ob die Anforderungen an das Metadatenmanagement (vor allem in Hinsicht auf die Schnittstelle zu den Archiven) und ein in Grenzen haltbarer Aufwand die Verwendung von RDF zwingend macht, oder ob vorhandene Grid Systeme wie die oben genannten oder der in Globus integrierte Metadata Catalog Service (MCS) verwendet werden.

Anhang A: Bibliographie

[asg] Adaptive Services Grid. EC Integrated Project 2004-2006, IST FP6 004617. www.asg-platform.org

[atlas] Zoi Kaoudi, Iris Miliaraki, Matoula Magiridou, Antonios PapadakisPesaresi and Manolis Koubarakis: Storing and Querying RDF Data in Atlas. Demos and Posters of the 3rd European Semantic Web Conference (ESWC 2006), Budva, Montenegro, June 2006. http://www.eswc2006.org/demo-papers/FD24-Kaoudi.pdf

[ding] Hao Ding and Ingeborg Sølvberg: Choosing Appropriate Peer-to-Peer Infrastructure for Your Digital Libraries. http://www.idi.ntnu.no/grupper/if/publikasjoner/conf.papers/HaoICADL05.short.pdf

[DINI-reinefeld] Alexander Reinefeld: Grid-Computing: Techniken, Standards, Perspektiven. Präsentation auf der 5. DINI-Jahrestagung, 29./30. September 2004.

[erl04] Thomas Erl: Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services. Prentice Hall, 2004. ISBN 0131428985.

[foster_phys] Ian Foster, Carl Kesselman, Jeffrey M. Nick, Steven Tuecke: The Physiology of the Grid. Globus Project, 2002.

[gsmo] Grid Service Modelling Ontology. www.gsmo.org

[oai] Open Archives Initiative, Protocol for Metadata Harvesting (OAI-PMH). http://www.openarchives.org/

[OGSA-DAI-RDF] Isao Kojima: Design and Implementation of OGSA-DAI-RDF. http://www.semanticgrid.org/GGF/ggf16/papers/kojima34.pdf

[ogsa_v1] The Open Grid Services Architecture, Version 1.0. http://www.ggf.org/documents/GFD.30.pdf

[reinefeld-schintke] Alexander Reinefeld und Florian Schintke: Dienste und Standards für das Grid Computing. In: J. von Knop, W. Haferkamp (Hrsg.), 18. DFN Arbeitstagung über Kommunikationsnetze, Düsseldorf, Lecture Notes in Informatics, Series of the German Informatics Society (GI), 2004, vol. P-55, pp. 293 - 304.

[sesame02] Jeen Broekstra, Arjohn Kampman, Frank van Harmelen: Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. In: Proceedings of the First Internation Semantic Web Conference, July 2002. http://www.cs.vu.nl/~frankh/abstracts/ISWC02.html

[sparql] SPARQL Query Language for RDF. W3C Candidate Recommendation 6 April 2006. http://www.w3.org/TR/rdf-sparql-query/

[Yu/Buyya2005] Jia Yu and Rajkumar Buyya: A Taxonomy of Workflow Management Systems for Grid Computing. GRIDS Lab, University of Melbourne Technical Report, GRIDS-TR-2005-1, Grid Computing and Distributed Systems Laboratory, University of Melbourne, Australia, March 10, 2005. http://www.gridbus.org/reports/GridWorkflowTaxonomy.pdf

[zing-srw] ZING (Z39.50 International Next Generation) - SRW: Search/Retrieve Web Service. Standard Maintenance Agency: Library of Congress. http://www.loc.gov/standards/sru/srw/