WissGrid

Deliverable 2.1.5

Arbeitspaket 2: Blaupausen und Beratung

Evaluation und Dokumentation existierender Architekturkonzepte[1]

Deliverable	2.1.5 Existierende Architekturkonzepte
Autoren	Arbeitspaket 2: Blaupausen und Beratung
Editoren	F. Schlünzen, I. Agapov
Datum	26-02-2010
Dokument Version	1.0.0

A: Status des Dokuments

Deliverable 2.1.5, Version 1.0.0, 2010-Iteration vom Projekt akzeptiert.

B: Bezug zum Projektplan

Die Dokumentation der Architekturkonzepte abstrahiert die Erfahrungen der CGs zur Weiterverwertung durch die Fachberater-Teams.

C: Abstract

Die Dokumentation und Evaluiering existierender Architektur-Konzepte im D-Grid dient als Grundlage für die Erstellung von Blaupausen architektonischer Konzepte für Grid-Infrastrukturen. Daraus ensteht eine Sammlung generischer Modelle, die neuen Communities durch das Expertenteam zur Verfügung gestellt werden können.

[1] This work is created by the WissGrid project. The project is funded by the German Federal Ministry of Education and Research (BMBF).

D: Änderungen

Version	Date	Name	Brief summary
0.1.0	10.11.2009	F.Schlünzen	Working Draft Creation
0.1.1	22.12.2009	I. Agapov	Some additions
0.1.2	07.01.2010	I. Agapov F.Schlünzen	1st full draft
0.1.3	16.01.2010	F.Schlünzen	Adding Input from MediGrid. Typos.
0.1.4	31.01.2010	I.Agapov, H.Enke, J.Falkner, C.Grimme, B.Löhnhardt, T. Rathmann, F.Schlünzen.	Several additions and corrections.
0.2.0	31.01.2010	I.Agapov, F.Schlünzen.	Several minor additions and corrections. Updated figures.
0.2.1	31.01.2010	H.Enke	Wording, grammar, added AstroGrid architectural schema.
0.2.2	25.02.2010	F.Schlünzen, H. Enke	Last corrections from B.Fritzsch and cosmetics, typos

E:

Inhaltsverzeichnis

1 Summary **5**

2 Einleitung **6**

3 Grid Architektur im Überblick **7**

3.1 Der Network-Layer 8

3.2 Der Resource-Layer 8

3.3 Der Middleware-Layer 10

3.3.1 Datenspeicherung, Transfer and Replikation 10

3.3.2 Information Services 11

3.3.3 Metadata 12

3.3.4 VO-Management 12

3.3.5 Security und User Management 12

3.4 Der Application-Layer 15

3.4.1 Workflow 15

3.4.2 Grid-Portale 16

4 Auswertung und Einordnung der Materialsammlung **18**

4.1 Architektur AstroGrid-D [4] 18

4.2 Architektur C3-Grid [5] 19

4.3 Architektur HEP Grid [6] 20

4.4 Architektur MediGrid [7] 21

4.5 Architektur TextGrid [9] 22

4.6 Zusammenfassung der Architekturkonzepte 23

5 Evaluierung der existierenden Architektur-Konzepte **24**

5.1 Middleware 24

5.2 Data-Management & Metadata 25

5.3 Authentifizierung / Authorisierung 26

5.4 Portals und Clients 26

5.5 Workflows 26
5.6 Prototypische Architekturen 27
5.6.1 AstroGrid-D Architektur 27
5.6.2 Photon Science Architekturen 28

A Detaillierte Beschreibung von Grid-Diensten 30
A.1 Grid Workflow Execution Service (GWES) 30
A.2 C3Grid Workflow Scheduling Service (WSS) 30

1 Summary

This document contains an overview of existing Grid Architectures, focusing on implementations of the mature D-Grid Community Grids. The document should serve as a basis for expert teams to support the creation of a new Community Grid utilizing the building blocks of the architecture concepts in a modular fashion.

It is meant as a collection of material for further study, providing easy access to otherwise mostly published, but very randomly distributed material. Since the working group is preparing in-depth studies of approaches to build and run a Community Grid (CG), this document is a starting point for a drill down. Then, it is a reference document for the WissGrid team of expert consultants to build on the experiences of the CGs of D-Grid. The document is not (yet) a complete compendium of CG experiences, solutions and services and will grow through added entries from the CGs.

How to read the document:

For introduction to grid architecture concepts:	read sections 2 and 3
For overview of solutions in Community Grids:	read section 4
For comparison of existing solutions :	read section 5
For an in-depth study:	Follow the links in the bibliography

2 Einleitung

Die etablierten D-Grid CGs verwenden eine Vielzahl unterschiedlicher Ansätze und Methoden um den spezifischen Anforderungen ihrer Community gerecht zu werden. Die Anforderungen orientieren sich auf einer übergeordneten Ebene nach dem primären Focus des CGs, also die grobe Klassifizierung als Computational Grid, Data Grid oder Service Grid. Verschiedene Faktoren wie zum Beispiel geographische Verteilung der Mitglieder, Sicherheitsbedürfnisse, Heterogenität der Applikations-Landschaft, Standardisierung von Datenformaten und Metadaten, erforderliche Services und Resourcen - um nur einige wenige zu nennen - bedingen letztlich die Diversität der Architekturkonzepte in der Grid-Landschaft.

Neue CGs stellen sich grundsätzlich dieselben Fragen und Probleme:

- welche Grid-Komponenten passen am besten zu den spezifischen Anforderungen,
- welche Komponenten lassen sich problemlos modular kombinieren,
- wo müssen Komponenten angepasst oder entwickelt werden,
- wo sind Informationen zu den Komponenten zu finden.

Aufgrund der Vielzahl von Lösungsansätzen ist es für eine Community, die erstmals mit dem Grid in Berührung kommt, praktisch unmöglich einen Überblick zu bekommen, die verschiedenen Ansätze im eigenen Kontext zu evaluieren oder Hintergrundinformationen zu den Ansätzen zu lokalisieren.

Die Materialiensammlung (Deliverable 2.1.1) dokumentiert zusammenfassend die Vorgehensmodelle, Techniken und Organisationsformen der etablierten CGs.

Das vorliegende Dokument entwickelt daher ausgehend von der Materialsammlung aus den unterschiedlichen Ansätzen und Implementierungen ein generisches Architekturkonzept, das als Basis für den konzeptionellen Aufbau neuer CGs dienen kann. Um die Konzepte in einen globalen Grid-Kontext zu stellen, wird zunächst die Grid-Architektur im Allgemeinen umrissen, wobei spezifische Punkte und Fragen, die für neue CGs von wesentlicher Bedeutung sein könnten, in einen Fragen- oder Anforderungskatalog einfließen.

Diesen Anforderungskatalog gilt es mit den vorhandenen Architekturkonzepten abzugleichen. Dafür wird eine ausführliche Auswertung der in der Materialiensammlung vorgestellten Architekturkonzepte vorgenommen und die wesentlichen Aspekte in komprimierter Form dargestellt. Da die etablierten CGs nicht das ganze konzeptionelle Spektrum abdecken können, wird es möglicherweise nötig sein, Konzepte aus anderen Communities mit einzubeziehen. Ein Beispiel wäre das International Lattice Data Grid (ILDG), das auf sehr spezielle Weise gLite, Data Management und HPC miteinander verbindet.

Um ein modulares Architekturkonzept zu gewinnen und letztlich mit einem CG spezifischen Anforderungskatalog zu verknüpfen, müssen anschließend die einzelnen Komponenten extrahiert und evaluiert werden.

3 Grid Architektur im Überblick

Grid-Architekturen werden üblicherweise mit einer Layer- oder Ebenen-Struktur beschrieben (Abb. 1). Jede dieser vier Ebenen, Application & Service, Middleware, Resources und Network, besteht wiederum aus einer Reihe von Ebenen und Bausteinen. Jedes CG verwendet eine ihr eigene Kombination von Ebenen und Grid-Bausteinen, um den Anforderungen ihrer Community zumindest weitgehend gerecht zu werden. Um anhand dieser divergierenden Implementationen ein generisches Architekturkonzept zu ermitteln, müssen zunächst die verwendeten Ebenen und Bausteine identifiziert werden, im Wesentlichen basierend auf der Materialiensammlung des Deliverable 2.1.1. Daraus sollte sich eine generische Architektur-Matrix konzipieren lassen, die schließlich mit der Anforderungs-Matrix der neuen CGs zu korrelieren sein wird.

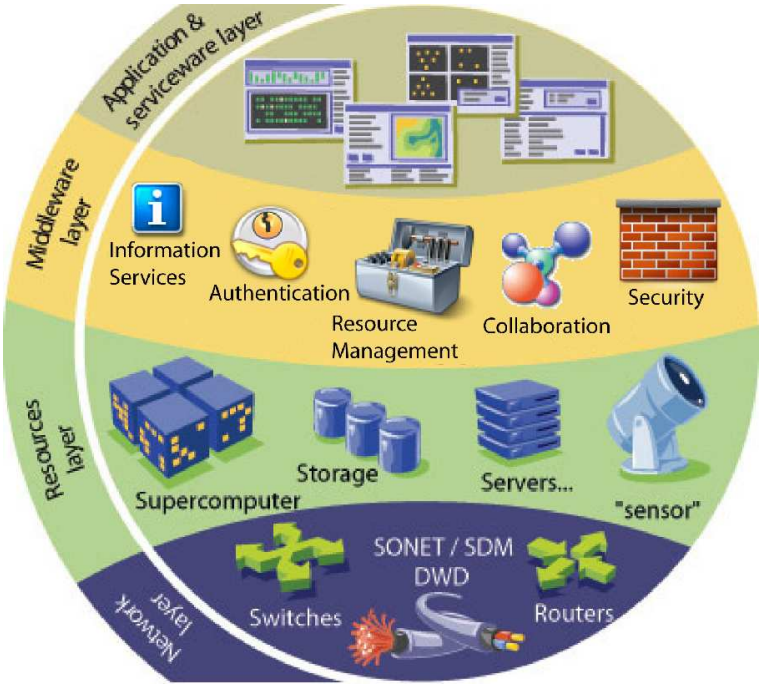

Abbildung 1: Layer Struktur des Grids. Adaptierte Abbildung aus [2]

Eine grundsätzliche Frage, die sich ein neues CG vorab stellen sollte, ist die Wahl der Grid-Topologie. Im Wesentlichen könnte man drei Topologien definieren:

1. IntraGrid:
   Das Grid wird intern innerhalb einer Organisation betrieben. Authentifizierung/Authorizierung basieren dann auf den Organisations-internen Trust-Mechanismen.
2. ExtraGrid:
   Das Grid wird von einem Konsortium von Organisationen innerhalb eines (virtuellen) privaten Netzwerk betrieben. Authentifizierung/Authorizierung basieren dann auf B2B-Mechanismen.
3. InterGrid:
   Das Grid basiert auf einem globalen, verteilten Zugriff der Resourcen über das Internet. Authentifizierung/Authorizierung basiert dann auf Zertifikaten (nicht notwendigerweise persönlichen Grid-Zertifikaten).

Naturgemäß ist nur das InterGrid ein echtes CG und das in diesem Kontext bevorzugte Modell. In Einzelfällen mag aber ein IntraGrid durchaus eine erwägenswerte Alternative sein. Das EMBL zum Beispiel verfolgt primär das Ziel ihren User Communities Services, Resourcen und Instrumente anzubieten. Die Einbettung in eine globale Infrastruktur mit anderen vergleichbaren Einrichtungen ist zwar von großem Vorteil, aber nicht unabdingbar.

3.1 Der Network-Layer

Der Network-Layer definiert die zugrunde liegende Netzwerk-Infrastruktur und Topologie. Letztlich wird die Infrastruktur eine untergeordnete Rolle in den Architektur-Konzepten spielen, da die vorhandenen Strukturen, seien es die Strukturen des DFN oder überregional des GEANT, hinreichend sein sollten. Die Topologie, im Wesentlichen wohl die Entscheidung zwischen Peer to Peer oder Client-Server Topologie, mag hingegen durchaus eine gewichtige Rolle spielen.

Die Minimierung von Latenzen, die z.B. aus der Kommunikation mit den Information Services entstehen, wird für manche Operationen, insbesondere die Anbindung von Real-Time Instrumenten, erforderlich sein.

Für extrem datenintensive Operation wie zum Beispiel im Umfeld der Photonenpyhsik wird es unabdingbar sein, Transfer und Replikation von Daten auf den vorhandenen Strukturen zu optimieren.

3.2 Der Resource-Layer

Der Resource-Layer vereint die verschiedenen Ressourcen, wie Compute-Elemente, Storage und Archive, oder physikalische Instrumente. In Bezug auf die Ressourcen gibt es durchaus einige Community-spezifische Fragen:

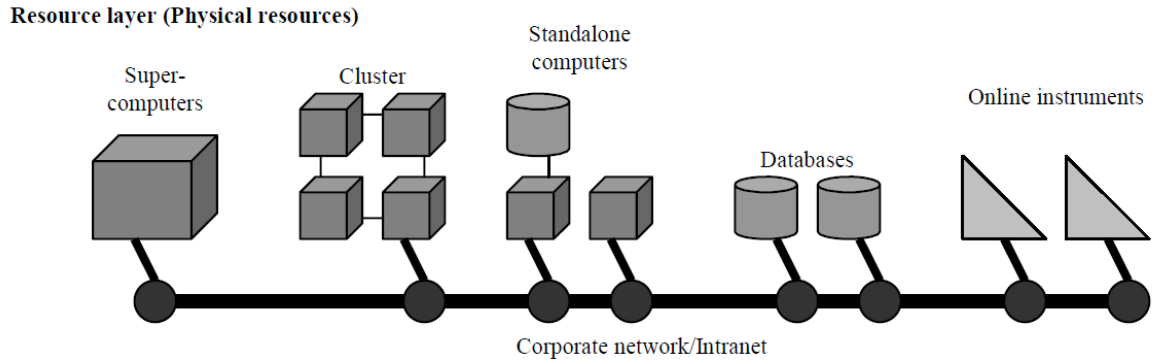

Abbildung 2: Resource Layer - aus [1]

- Instrumente:
  - Ist die Einbindung physikalischer Instrumente notwendig?
  - Sind diese Instrumente strikt lokal, national oder weltweit verteilt?
  - Welche Latenzen sind bei dem Zugriff auf die Instrumente tolerabel?
  - Sind existierende Ansätze wie DORII brauchbar?

- Computing:
  - Welche Betriebssysteme muüssen unterstützt werden?
  - Sind lokale Resourcen (Batch-Farmen, Cluster) mit einzubinden?
  - Ist HPC-Computing (MPI) erforderlich?
  - Sind die Anwendungen zeitkritisch (0-Latenz)?
- Daten:
  - Wie groß sind Datenvolumen?
  - Wie groß sind Datenraten im Mittel?
  - Wie groß sind Datenraten in der Spitze (bursts)?
  - Wie groß ist eine typische Datei?
  - Lassen sich einzelne Dateien logisch in größeren Datensätzen organisieren?
  - Existieren Standards für Daten und Metadaten?

3.3 Der Middleware-Layer

Der Middleware-Layer, wie er klassischerweise in Abbildung 1 definiert wird, umfasst eine Vielzahl von Komponenten, Services und Zwischenebenen. In dieser Auslegung ist der Middleware-Layer ultimativ entscheidend für die Akzeptanz des CGs innerhalb der User Community. Als Middleware-Systeme, die die wesentlichen Komponenten und Services integrieren, kommen im Wesentlichen drei verschiedene Ansätze in Frage: Globus [61], gLite [60] und Unicore [62]. Diese drei Middleware-Systeme werden von DGI unterstützt und auch OMNII-Europe fokussiert primär auf diese drei Platformen.

Globus, gLite und Unicore haben unterschiedliche Charakteristika (Abb. 3), die sich in vielen Aspekten wie Authentifizierung, Job-Submission und Monitoring, Betriebssystem-Unterstützung niederschlagen. Unicore z.B. fokussiert primär auf den Remote-Zugang zu Supercomputer-Zentren, während gLite sich stark an den Anforderungen des LHC orientiert. Das Globus Toolkit ist außerhalb der HEP-Welt die populärste Middleware. Die meisten der D-Grid VOs basieren auf dem Globus Toolkit und gLite selbst inkorporiert einige der Globus Module. Gravierenden Einfluss auf die Entscheidung für eine spezifische Middleware können insbesondere die unterstützten Authentifizierungs-Konzepte und Job-Submission-Module haben. Die Integration von Shibboleth z.B. scheint in Unicore am weitesten fortgeschritten. Im Compute-Grid könnten die Möglichkeiten für Bulk-Job-Submission und parametrische Jobs in gLite den Ausschlag geben.

Neben diesen drei Plattformen sind in Europa zumindest noch NorduGrid's ARC, OMNII-UK und XtreemOS im Einsatz, werden in den Architektur-Konzepten aber keine Rolle spielen. Schon aus Effizienz-Gründen sollten die Architekturkonzepte auf einer möglichst kleinen Zahl von Middleware-Systemen basieren.

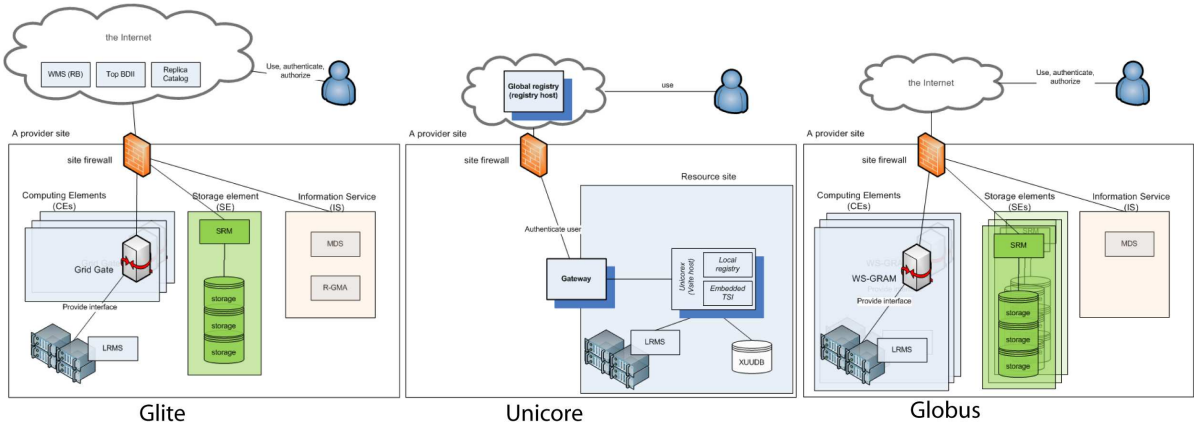

Abbildung 3: Middleware Layout - entnommen des wiki der DGI Referenz-Installation [3]

3.3.1 Datenspeicherung, Transfer and Replikation

Im Grid wird die Datenspeicherung durch Storage Elements (SEs) realisiert. Die SEs kontrollieren den physikalischen Speicher unabhängig von den physikalischen Medien (Festplatten, Bänder, etc). Der Zugriff auf die Daten erfolgt zum Beispiel mit GSIFTP (FTP mit GSI als Sicherheitsmechanimus) für den Transfer ganzer Dateien. Lokaler oder Remote-Zugriff auf Datei-Segmente kann - abhängig von dem Storage-Backend - mittels RFIO oder gsidcap erfolgen. Die unterstützten Pro-

tokolle können für eine neue Grid-Community durchaus von Bedeutung sein. Für GridFTP z.B. existieren HDF5-Plugins, einem der Standard-Formate der Photon Science Communities. Für den dCache Client hingegen ist dies bislang keine Option, direkte Implementierung des dCap-Protokolls in CIF und HDF5 ist aber in Planung.

In der gLite Middleware wird u.a. der Zugriff und die Reservierung von Speicherplatz durch Storage Resource Manager (SRMs) kontrolliert [13], unabhängig von dem eigentlichen Storage Backend wie DPM oder dCache. Beispiele für SRMs sind Castor-SRM, dCache-SRM, Berkeley Storage, STORM, SRB oder iRods. Alle SRMs sollen der SRM-Spezifikation folgen und interoperabel sein, so dass die Wahl eines spezifischen SRMs weitgehend unabhängig von der Middleware getroffen werden kann.

SRMs implementieren u.a. die folgenden Konzepte:

- Reservierung von Speicherplatz
- Dynamisches Management des Speicherplatzes
- Lokalisierung der Dateien auf den physikalischen Medien
- Unterstützung abstrakter Konzepte für Dateinamen wie Site-URLs
- Temporäre Zuordnung von Dateinamen für den Transfer wie Transfer-URLs
- Directory Management und Authorisierung
- Transfer Protocol Negotiation
- Support für Peer to Peer (P2P) Anfragen
- Support für asynchrone Multi-File Anfragen
- Support für Abort, Suspend, und Resume
- Respektierung lokaler Policies

Vergleichbare Middleware Services in Globus sind Replica Location Service (RLS) und Reliable File Transfer (RFT).

3.3.2 Information Services

Information Services stellen die essentiellen Informationen zu Grid-Resourcen und deren Status zur Verfügung, was grundlegend für den Betrieb des Grids an sich ist. Die veröffentlichten Informationen beinhalten ebenfalls Daten für die Überwachung und das Accounting im Grid. gLite stellt diese Informationen konform mit dem GLUE Schema bereit.

Globus IS verwenden Monitoring and Discovery (web) Services (MDS) in Verbindung mit XPath oder XQuery.
gLite IS basieren auf dem Berkeley Database Information Index (BDII) und OpenLDAP.
In MediGRID ist eine Service-Registry implementiert, in welcher Dienste zentral bekannt gemacht werden können. Umgesetzt ist diese Service-Registry durch eine eXist-XML-Datenbank mit Ressourcen und Service-Einträgen im D-GRDL Format. Insbesondere der Grid Workflow Execution Service nutzt diese Service-Registry zur Auswahl der Ressourcen bei der Ausführung von Jobs. Der Resource Matcher Daemon, sofern er auf den Ressourcen installiert ist, versorgt diese Datenbank automatisch mit aktuellen Informationen zu den Ressourcen.

3.3.3 Metadata

Metadaten sind essentiell für die Organisation der Daten, den kontextbasierten Zugriff oder Data Mining. Metadaten Management kann basierend auf RDBMS wie MySQL oder Directory Services wie OpenLDAP implementiert werden. Um Grid PKI Authentifizierung und Authorisierung zu unterstützen sollten Metadata Services mit der entsprechenden Middleware zusammenarbeiten. AMGA ist eine Metadaten Engine die von gLite unterstützt wird, ACLs implementiert und die gängigsten Datenbank-Systeme wie MySQL, PostgreSQL oder Oracle unterstützt. Verschiedene APIs erlauben komplexe Datenbank-Suchen direkt aus den Anwendungen heraus. Ein Beispiel für eine AMGA-Implementierung findet sich in den *Grid-based digital libraries* (https://glibrary.ct.infn.it/).

Das Globus Toolkit unterstützt ein breiteres Spektrum an Metadata Engines oder Informationskataloge in weiterem Sinne. Einige der Metadata-Services basieren auf Globus Metacomputing Directory Service (MDS) wie zum Beispiel RIB-Globus (Repository in a Box) oder NorduGrids Information-System. Metadata-Management-Systeme wie AMGA, Stellaris, MCAT (der Metadaten-Katalog des DICE Storage Resource Brokers SRB), UNIDART (Uniform Data Request Interface), CDMS (Climate Data Management System) oder iRODS (um nur einige zu nennen) lassen sich an die Globus-Middleware anbinden.

3.3.4 VO-Management

Da VOMS/VOMRS Services weitgehend Middleware-übergreifend einsetzbar sind, ist die Abhängigkeit zu spezifischen Middleware Systemen eher gering. Es stellt sich für neue CGs mehr die Frage, ob eine eigene VO-Infrastruktur erforderlich ist, eine generische VO (z.B. xray-vo für Photon Sciences) sinnvoll ist und in welches Framework (national/international) die VO einzubetten wäre. Gravierende Unterschiede bestehen allerdings - abhängig von der verwendeten Middleware - im VO-Management und damit verbundenen AAA-Aspekten. Das D-Grid Projekt *Interoperability and Integration of VO management Technologies in D-Grid* (IVOM, [15]) zielte darauf ab eine Middleware-übergreifende VO-Management Struktur zu implementieren. Das AeroGrid-Projekt verwendet laut eigenen Angaben die IVOM-Entwicklungen, um die VO-Sicherheit in der UNICORE/OGSA-DAI basierten Grid-Infrastruktur zu gewährleisten.

3.3.5 Security und User Management

Das User Management ist eines der zentralen Elemente, insbesondere die AAA-Mechanismen. Die meisten Authentifizierungs- und Authorisierungs-Infrastrukturen (AAI) im Grid verwenden einfache Authorisierungs-Mechanismen basierend auf dem *Distinguished Name* (DN) eines X.509 User-Zertifikats. Da die Grid Security Infrastructure (GSI) auf X.509 Proxy-Zertifikaten basiert, die sich aus X.509 User-Zertifikaten ableiten, sind die Informationen über den DN für die Grid-Resourcen, auf denen ein User oder User-authorisierte Grid Services agieren, immer verfügbar. Daher war es eine naheliegende Wahl, nicht nur die Authentifizierung, sondern ebenfalls die Authorisierung an den DN zu binden.

Dieses Authorisierungs-Schema erfordert allerdings, dass die DNs aller User, die Zugriff auf eine Grid-Resource haben sollen, in einem Grid-Mapfile gepflegt werden müssen, das eine eindeutige Zuordnung zwischen DN und lokalem Account ermöglicht.

Dies ist allerdings eine Lösung, die sehr schlecht skaliert. In der Photonenphysik ist das Problem besonders ausgeprägt, da ACLs und GridMaps für jährlich rund 3000 neue Anwender zu pflegen wären, die die Infrastruktur nur für einen sehr begrenzten Zeitraum nutzen. Viele Anwender haben keinen komfortablen Zugang zu RAs. Die Hürden für die Bildung einer DFN-RA sind zwar nicht besonders hoch, der Aufwand, die 3000 Anwender lokal zu registrieren, macht das Vorhaben aber wenig attraktiv für die Service-Anbieter. Zur Zeit werden die Probleme durch die Minimierung der Anzahl der User Accounts (ein Account pro Experiment oder Instrument) umgangen. Dies ist aber kaum mit den Policies der personalisierten Grid-Zertifikaten in Einklang zu bringen.

Zudem gibt es das Konzept der Attribut-basierten Authorisierung. Mit dem Grid-Zertifikat werden dabei Attribute wie VO-Mitgliedschaft, Gruppen und Rollen für den DN übermittelt. Dieser Mechanismus findet allerdings bislang nur in gLite-basierten Grids Verwendung und ist auf wenige Attribute (*fully qualified attribute names*, FQAN)) beschränkt, Nationalität oder Institut-Zugehörigkeit gehören zum Beispiel nicht dazu.

Eine große Hürde bei der Nutzung der PKI-basierten Authentifzierung ist der Upload von Proxies und das Beziehen von Credentials durch Endnutzer. Hierfür gibt es inzwischen eine Lösung aus dem Gap-SLC Projekt. Die Lösung heißt gPUT (Grid Proxy Upload Tool) und kann via Fraunhofer IAO bezogen werden.

Ein anderer Weg, die Attribut-basierte Authorisierung zu realisieren, beruht auf Federated IDs. Die Photon Science Communities haben z.B. einige der gängigen Identity-Provider (IdP) und Mechanismen evaluiert. Als beste Kandidaten wurden dabei OpenID und Shibboleth basierte Authorisierung identifiziert. Eine OpenID-Authorisierungs-Infrastruktur und Trust-Mechanismen zwischen den IdP's sind vergleichsweise einfach zu realisieren. Allerdings bietet OpenID bestenfalls rudimentäre Revocation-Mechanismen, wodurch das Management kompromittierter Id's erheblich erschwert wird. Im Rahmen des EuroFEL-Projektes haben daher einige der beteiligten Institute begonnen, einen Shibboleth-Prototyp aufzubauen. Das besondere Problem liegt hier in der Internationalität der Community, die nicht nur Trust-Mechanismen zwischen dem Identity-Provider und dem Service-Provider erfordert, sondern darüber hinaus Trust-Mechanismen zwischen den IdP's selbst. Im Grid könnte man den IdP-Trust prinzipiell dem VO-Management-Systemen überlassen, aber insbesondere die Authorisierung auf lokalen Resourcen, die im Photon Science Umfeld unbedingt gegeben sein muss, lässt sich damit nicht abbilden.

Die Abbildung 4 skizziert den Ablauf einer GridShib-Transaktion (übernommen aus [21]), bei der der Grid Service Provider (Grid SP) die Attribute vom Identity Provider (IdP) anfordert (Pull Modus):

- Ein Nutzer authentifiziert sich mit seinem X.509 Zertifikat bei einem Grid Service Provider (Grid SP), um einen Grid-Dienst zu nutzen. Der Grid SP entnimmt dem Zertifikat den Distinguished Name (DN).
- Der Grid SP authentifiziert sich am IdP des Nutzers und sendet eine SAML-Attributabfrage. Als Schlüssel dient der DN.
- Die Attribute Authority (AA) des IdP authentifiziert die Attributabfrage, bildet die DN auf einen lokalen Namen ab und sendet unter Vorbehalt der Attribute Release Policy (ARP) eine Zusicherung (Assertion) mit den gewünschten Attributen an den Grid SP zurück.
- Der Grid SP trifft anhand der Attribute eine Entscheidung über die Gewährung und die Art des Zugangs, bearbeitet im positiven Fall die Anfrage und liefert das Ergebnis an den Nutzer zurück. Als Einschränkung wird in GridShib derzeit die mangelnde Skalierbarkeit des dateibasierten Name-Mapping angesehen. Die Datei soll zukünftig durch eine Datenbank

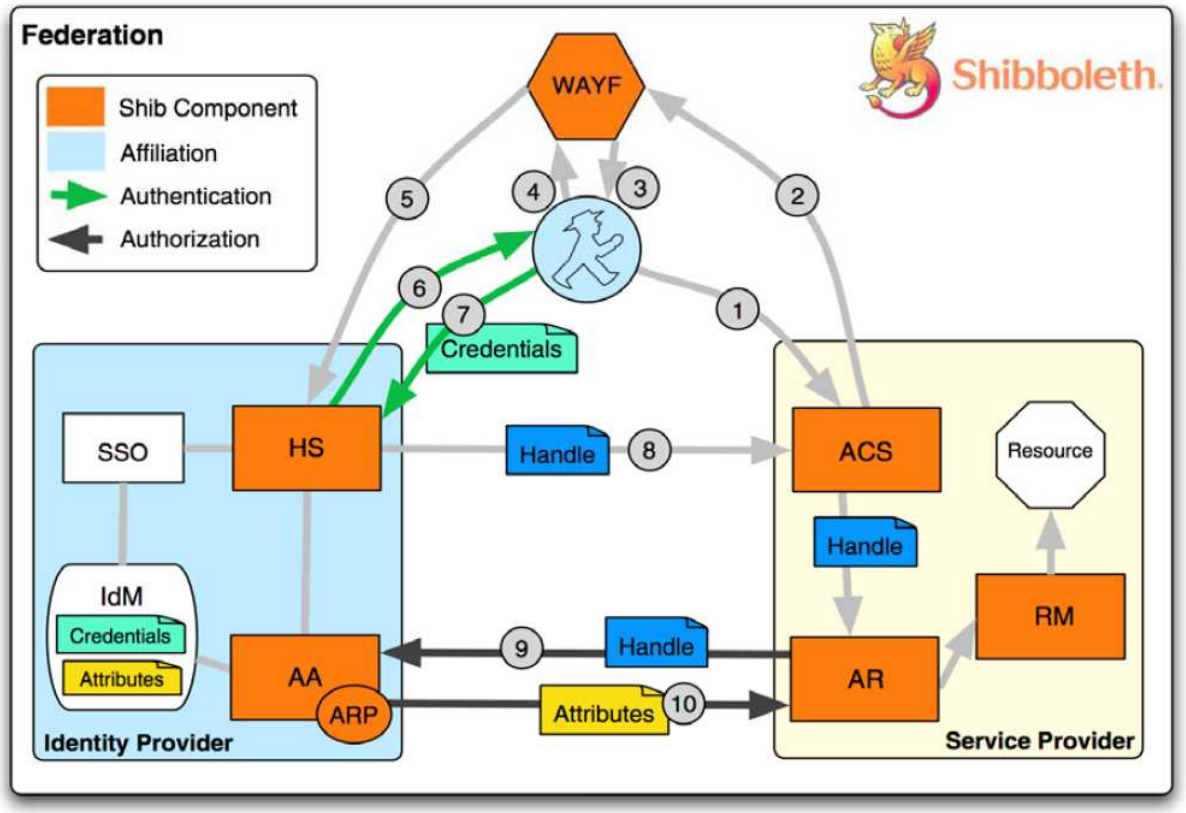

Abbildung 4: GridShib [17], [21]

ersetzt werden. Es fehlt zudem ein Mechanismus, um den Identity Provider (IdP) einer Person automatisch zu ermitteln. Alternativ zum Pull Modus wird ein Push Modus diskutiert: der IdP übergibt dem Nutzer (Client) die Attribute Assertion, die der Nutzer direkt bei der Kommunikation mit einem IdP verwenden kann.

Die Zuordnung der Authorisierung zu dem DN beruht in GridShib wie auch anderen AA-Schemata auf Mapping-Dateien, was für große Anwenderzahlen nicht gut skaliert.

Für viele Communities, insbesondere die Photon Science Communities, wird eine nahtlose Integration von Federated IDs in die Grid-Umgebung den Einstieg zumindest erheblich erleichtern. Das GAP-SLC Projekt bietet solche Lösungen an. Anwender, die einen Short-Lived Credential Service (SLCS) nutzen, können die DFN-AAI Föderation zur Authentifizierung via Shibboleth nutzen. Um die Autorisierung bei den Ressourcenprovidern feingranular zu gestalten, werden SAML (Secure Assertion Markup Language) Assertions verwendet, die auch Informationen aus dem VO-Management enthalten sollen [22].

3.4 Der Application-Layer

3.4.1 Workflow

Wissenschaftliche Workflows stellen flexible Werkzeuge bereit für den Zugriff und die (komplexe) Analyse wissenschaftlicher Daten, wie medizinische oder Satelliten-Bilder, Simulationsergebnisse oder bio-physikalische Daten.

Einen Überblick über Grid-fähige Workflows findet man unter http://www.gridworkflow.org/. Zur Zeit verfügbare Grid-Workflowsysteme sind unter anderem Taverna [52], GWES [53] , Kepler [55], Triana [56], oder P-Grade [48]. In einigen Fällen wie z.B. GWES oder P-GRADE citePgrade-Portal ist das Worklow Management in das entsprechende Grid-Portal integriert. Auch in MediGRID gibt sehr komfortable GWES Portlets, die zudem Application Monitoring in Echtzeit ermöglichen. Diese sind für Gridsphere umgesetzt, eine Portierung zu LifeRay ist derzeit in Planung. Weiterhin gibt es Beispielportlets für den Aufruf des GWES-Service aus portalbasierten Grid-Service-Clients.

ORNL [47] zum Beispiel verwendet einen Workflow (basierend auf dem Kepler Workflow Management) das eine Simulation auf einem Remoterechner überwacht, die Ergebnisse der Simulation instantan zu einer anderen Resource transferiert, wo die Daten analysiert, visualisiert und gespeichert werden. Eine andere Anwendung dient der Modellierung komplexer Prozesse des Plasmas in Tokamak-Reaktoren. Dabei laufen zahlreiche verschiedene Applikationen auf unterschiedlichen Resourcen. Der Workflow steuert und synchronisiert die Ereignisse, sichert den Datenaustausch zwischen den Applikationen und die Provenienz während der Simulation.

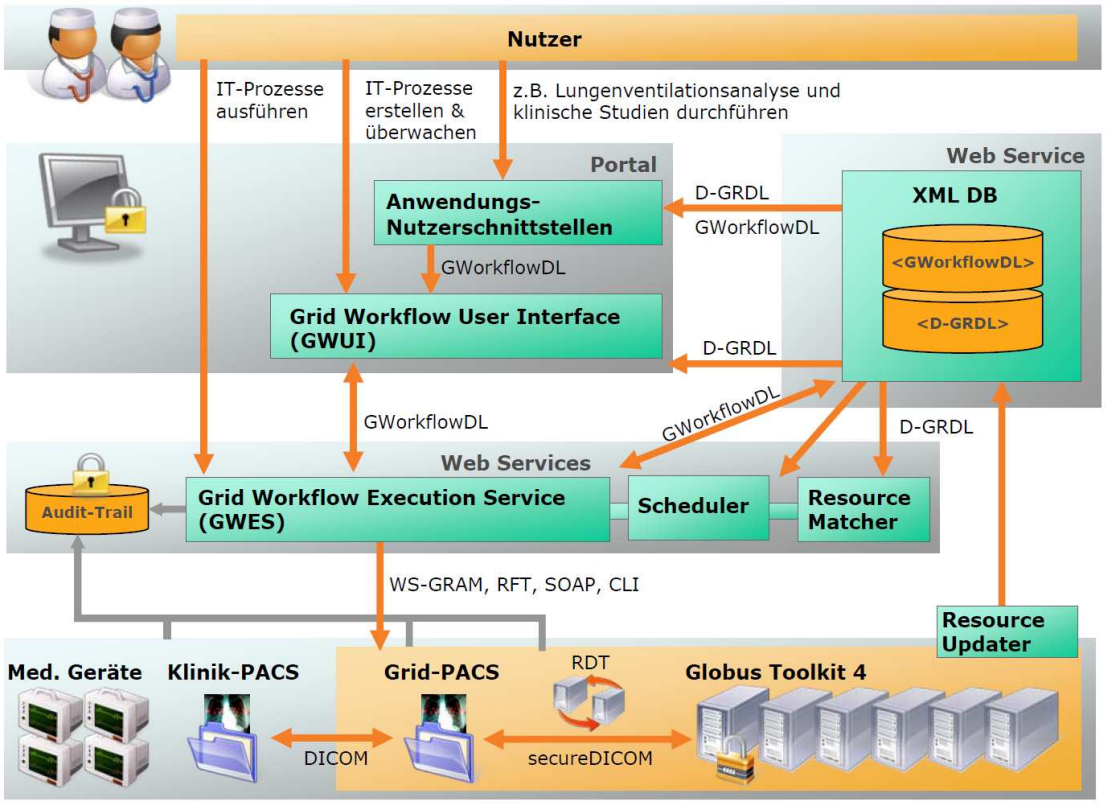

Abbildung 5: Ein Beispiel (GWES) für die Verwendung von Workflow-Systemen im Grid [54]

3.4.2 Grid-Portale

Grid Portale stellen einen integrierten Zugang zu den heterogenen Backend-Resourcen und Services bereit. In den meisten Fällen sind die Grid-Portale Web-basiert (und unterliegen damit den Limitierungen des http-Protokolls).

Grid-Portale existieren mittlerweile für einen weiten Bereich wissenschaftlicher Kollaboration und Communities, wie zum Beispiel Chemie (GridChem), Astronomie (NVO), Physik (FusionGrid, PPDG, Cactus, LatticeQCD), Medizin/Biologie (BIRN), Nanotechnologie (nanoHUB), Geophysik (GEON, NASA QuakeSim), Klima und Wetter (Earth Sciences Grid/ESG, LEAD) und viele mehr.

Ursprünglich waren Grid-Portale eng mit der Middleware wie Globus verknüpft und bedienen sich einer recht starren Implementation:

- HTTPS, HTML, JSP, JavaScript für die Web-Dienste
- MyProxy and Globus GSI für die Authentifizierung
- GRAM für Job Submission
- MDS für Information Services.
- GSIFTP und GridFTP für Datentransfer

Im wesentlichen bedienen die Services eine 3-tiered Architektur: Das Web Portal öffnet eine sichere Verbindung zu dem Web-Server; der Server erhält ein Zertifikat von einem Proxy-Server; der Web-Server startet einen Application-Manager und delegiert die User-Proxy-Zertifikate zu dem Application-Manager.

Die zweite Generation der Grid-Portale bietet, basierend auf Portlets, eine deutlich flexiblere Lösung. Die Portlets können dynamisch addiert, entfernt oder modifiziert werden. Ein Vielzahl von Toolkits implementieren Portlets im weiteren Sinne, wie z.B. Liferay[44], Microsoft SharePoint Server [33], Plumtree Corporate Portal [38], Viador E-Portal [41], IBM WebSphere Portal Server [43], Sun ONE Portal Server [40], Apache Jetspeed [30], GridSphere [29], and Oracle Portal Development Kit [35].

GridSphere zählt zu den Thin Clients, die ausschließlich einen Webbrowser und keine weiteren Applikationen auf dem Client benötigen und sich ausschließlich des HTTP zur Kommunikation bedienen. GridSphere wird in D-Grid CGs vielfach eingesetzt. Im Gegensatz dazu bieten Fat Clients wie Migrating Desktop (http://desktop.psnc.pl/), Unicore Rich Client[45] oder Remote Component Environment[46] ein integriertes GUI, dass speziell auf Anwender-Bedürfnisse zugeschnitten ist und die Einbindung von Applikationen sowie die Visualisierung im Grid wesentlich vereinfachen soll, unabhängig von der verwendeten Middleware. Migrating Desktop bietet unter anderem:

- Einfache Integration mit Grid-Applikationen
- Einfache Job-Definition, Submission, Monitoring und Visualisierung der Ergebnisse
- Support für Batch- und interaktive Jobs
- Support für sequentielle wie parallelisierte Anwendungen
- Intuitives Data-Grid-Management
- Leicht erweiterbares Framework

Jetspeed verfolgt ein interessantes Konzept, das exemplarisch im Alliance Portal Project [26] Verwendung findet. Jetspeed erlaubt die Integration dynamischer Informationen, Personen und Prozesse

über die Grenzen von Organisationen hinweg. Inhalte werden durch verschiedene Portlets aggregiert. Jetspeed offeriert Features wie (http://portals.apache.org/jetspeed-2/):

- Sicherer Zugang - Sicherheit basiert auf Standards, ACLs
- Single Point of Entry (SSO, Federated)
- Enterprise Integration - (EAI, integration points)
- Personalisierung
- Dynamische Web-Komponenten - (standardisierte Portlets)
- Skalierbare, modulare Architektur mit Multi-Threading-Support.

gEclipse ist in gewissen Sinne ebenfalls ein Fat Client, geht aber weit über den Ansatz von Migrating Desktop hinaus, da es eine Middleware- und Platform-unabhängige, integrierte Grid-Umgebung für Grid-Anwender wie Operateure und Entwickler anbietet. gEclipse ist letztlich ein Plugin, das auf auf dem OpenSource Produkt Eclipse aufsetzt, und daher leicht durch weitere Plugins an die spezifischen Anforderungen angepasst werden kann. gEclipse findet sowohl in TextGrid als auch im HEPCG Verwendung.

Weitergehende Entwicklungen (*third generation portals*) zielen darauf hin, das Grid als abstrakte Problem-Lösungs-Umgebung darzustellen. PortalLab oder e-lico integrieren Portlets und Semantics um Applikationen und Data Mining kontext-sensitiv ins Grid zu portieren.

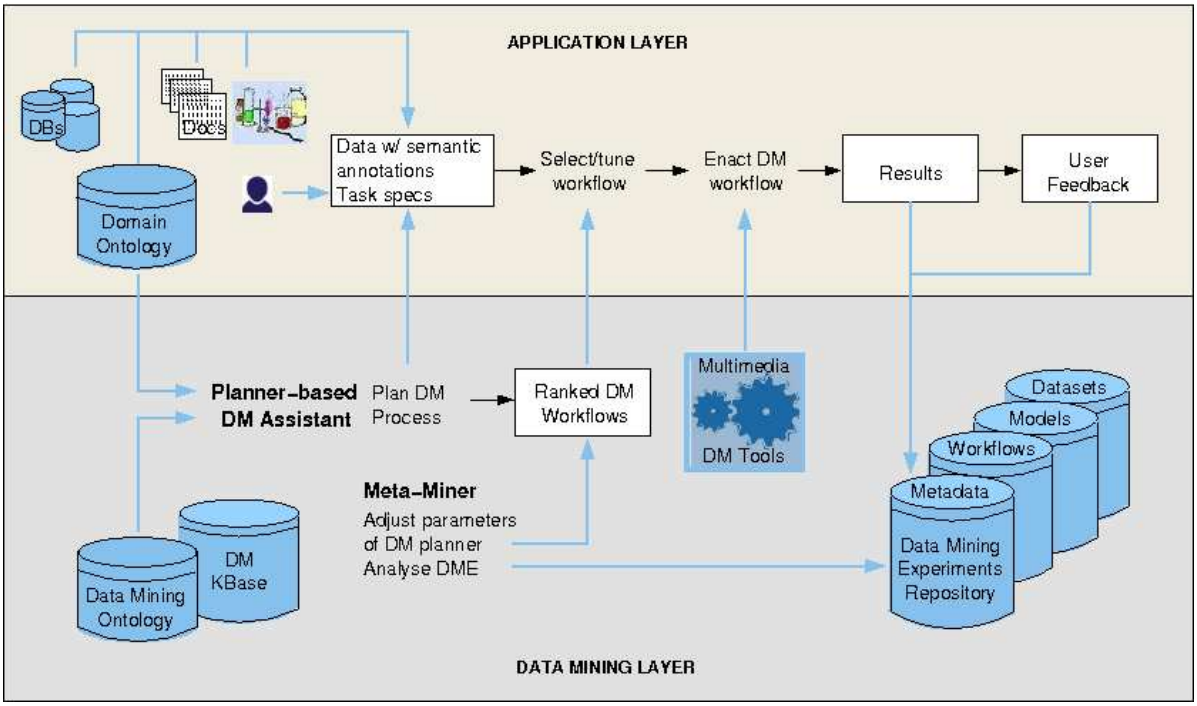

Abbildung 6: e-lico Architektur des Application und Data-Mining Layers

4 Auswertung und Einordnung der Materialsammlung

4.1 Architektur AstroGrid-D [4]

- Network
  - Topologie: Peer to Peer
- Resources
  - Data-Management: AstroGrid-D DataManagement (ADM), Stream basiertes Datem
  - Data-Replication: Globus Replica Location Service/ADM
  - Data-Transfer: u.a. GridFTP+HDF5plugin, gsiscp
  - Data-Ownership: Creator
  - Data-Access: ACLs
  - Information Service: Stellaris
  - Metadata: Globus-XML, RDF/w3c (Resource Description Framework),RTML
  - Metadata-Query: SPARQL/w3c (RDF Query Language)
  - MPI Support: PBS, LSF, Condor, SGE, LoadLeveler
- Middleware
  - Platform: Globus
  - Job-Submission: Globus, WS-Gram (Web Services Grid Resource Allocation and Management)
  - Job-Monitoring: Globus + Anwendungsspezifische WebTools (z.B. Cactus), TimeLine
  - Job-Language: Job Description Document (JSDL)
  - API: Globus, Grid Application Toolkit (GAT)
- User-Management
  - VO-Management: OSG VOMRS server + UserManagement auf lokalen Resourcen
  - Authentication: Grid certificate
  - Registration: GridKA, DFN-RA Netzwerk, Nationale CAs und lokale RAs
  - Zugriff auf lokale Resourcen: GridMap VO-Member-ID auf lokale UID
  - Support: Mailing-Listen, JIRA
- Security
  - X509 based, GSI/Globus Middleware, MyProxy
  - Revocation EugridPMA conform
  - Globus Port Monitoring
- Application
  - Portal: GridSphere based
  - Lizenzmanagement: keines.

4.2 Architektur C3-Grid [5]

- Network
  - Topologie: Client-Server
- Resources
  - Data-Management: WSRF Data-Management-Service
  - Data-Replication: nutzerinitiierte Replikate (mit beschränkter Lebensdauer) werden vom DMS verwaltet; bisher jedoch keine automatische Erstellung von Replikaten
  - Data-Transfer: intern GridFTP, vom User-Client aus HTTP
  - Data-Ownership: Datenzentrum
  - Data-Access: Wrapper + Legacy Systems, WSRF-Service
  - Information Service: Dateninformationsservice (DIS)
  - Metadata: OAI, automatisch generiert. ISO Standard.
  - Metadata-Query: OAI
  - MPI Support: keiner
- Middleware
  - Platform: Globus
  - Job-Submission: Globus, WS-Gram
  - Job-Monitoring: Globus
  - API: Globus
  - Job-Description: JSDL
  - Workflow Management: C3Grid Workflow Scheduling Service (WSS)
  - Workflow Specification Language (WSL)
  - Ressource Information Service RIS based on Globus MDS4
- User-Management
  - VO-Management: -
  - Authentication: UID, Registration, MyProxy
  - Registration: Instituts-eigene RAs
  - Authentication(i.A.): Shibboleth & SAML, DFN/SLCS, DFN-MyProxy
  - Registration(i.A.): DFN-AAI
  - Zugriff auf lokale Resourcen: rollenbasiert
  - Support: durch Portalverwalter für Nutzer; Mailinglisten für Admins und Entwickler
- Security
  - X509 based, GSI/Globus Middleware
  - Proxy delegation
  - GSI & GridShib

- Application
  - Portal: auf Basis von GridSphere
  - Lizenzmanagement: keines.

4.3 Architektur HEP Grid [6]

- Network
  - Topologie: Client-Server
- Resources
  - Data-Management: LFC, dCache, CASTOR
  - Data-Replication: lcg, oracle-streams
  - Data-Transfer: lcg-cp, gridFTP, gsidcap, NFS4
  - Data-Ownership: per Experiment
  - Data-Access: ACLs (oder World-Readable)
  - Information Service: gLite-BDII
  - Metadata: Amga (Backends Oracle, mySQL, postgreSQL)
  - Metadata-Query: Amga-Query-Language, SQL, WS-DAI Interface zu WSDAIR (relational DB) und WS-DAIX (XML DB)
  - MPI Support: SGE (NAF)
- Middleware
  - Platform: gLite
  - Job-Submission: gLite-BDII
  - Job-Monitoring: lcg-job-monitor
  - Job-Language: JDL
  - Middleware API: gLite
  - Workflow Management: gLite WMS
  - Bulksubmission, parametric Jobs supported
- User-Management
  - VO-Management: VOMRS/VOMS Betrieb
  - Authentication: Certificate
  - Registration: eigene RA für GridKA CA
  - Zugriff auf lokale Resourcen: Gridmapfile und proxy-forwarding, myProxy
  - Support: GGUS
- Security
  - X509, myProxy
- Application
  - Portal: Eclipse, p-Grade, Ganga, dq2
  - Lizenzmanagement: keines.

4.4 Architektur MediGrid [7]

- Network
  - Topologie: Client-Server
- Resources
  - Data-Management: SRB / iRods
  - Data-Replication: SRB / iRods
  - Data-Transfer: GridFTP, SRB-gridFTP-Connector
  - Data-Ownership: private
  - Data-Access: private
  - Information Service: eXist XML DB mit Informationen über verfügbare Dienste
  - Metadata: D-GRDL, OGSA-DAI
  - Metadata-Query: XML basiert
  - MPI Support: -
- Middleware
  - Platform: Globus
  - Job-Submission: WS-GRAM
  - Job-Monitoring: Globus, GWES
  - Middleware API: Grid Application Toolkit (GAT)
  - Grid Workflow Execution Service (GWES)
  - Resource Broker (GWES)
  - Meta-Scheduler (GWES)
  - Grid-Workflow-Description-Language (GWorkflowDL)
  - D-GRDL/eXist-basierte Service und Resource Registry
- User-Management
  - VO-Management: D-Grid VOMRS, geplant: VOMS-SAML-Service
  - Authentication: Certificate
  - Registration: DFN-RA Netzwerk
  - Zugriff auf lokale Resourcen:
  - Support: DGUS, Mailinglisten, Schulungen (Entwickler-Workshops)
- Security
  - X509, myProxy
  - Grid Proxy Upload Tool (gPUT)
  - Zertifikatsbasierter Login am Portal
  - Organisatorische Sicherheitsmaßnahmen (Policies, MediGRID Security Board)
- Application
  - Portal: GridSphere (Migration nach LifeRay)
  - Lizenzmanagement: keines.

4.5 Architektur TextGrid [9]

- Network
  - Topologie:
- Resources
  - Data-Management: Globus
  - Data-Replication: Globus
  - Data-Transfer: GridFTP
  - Data-Ownership: Institut
  - Data-Access: Shibboleth/RBAC
  - Information Service:
  - Metadata: XML, RDF/w3c, METS
  - Metadata-Query: XML basiert
  - MPI Support: -
- Middleware
  - Platform: Globus
  - Job-Submission: GLobus
  - Job-Monitoring: Globus
  - Middleware API: SAGA, GAT
  - Grid Workflow Execution Service (GWES)
  - Resource Broker (GWES)
  - Meta-Scheduler (GWES)
- User-Management
  - VO-Management: D-Grid VOMRS
  - Authentication: Certificate, Shibboleth erforderlich
  - Registration: DFN-CA Netzwerk
  - Zugriff auf lokale Resourcen:
- Security
  - X.509 basiert
- Application
  - Portal: Eclipse basiert, GridSphere
  - Lizenzmanagement: -

4.6 Zusammenfassung der Architekturkonzepte

	AstroGrid	C3-Grid	HEPCG	MediGrid	TextGrid
Class	Comp./Data	Data	Comp./Data	Compute	Data
Network	P2P	Client-Server	Client-Server	Client-Server	Client-Server
Resources::Data					
Management	Stream	WSRF	LFC, dCache	SRB/iRods	Globus
Replication	-	manuell DMS	lcg, oracle	-	-
Transfer	GridFTP GSIscp	GridFTP HTTP	GridFTP lcg-cp gsidcap, NFS4	DICOM gsiFTP	GridFTP
Ownership	Creator	RZ	Experiment	private	Institut
Access	ACLs	Wrapper Legacy System WSRF Service	ACLs	ACLs	RBAC Shibboleth
Resource::Information					
IS	Stellaris	DIS	gLite-BDII	iRods	
Metadata	XML, RDF	OAI	AMGA	DAI, GRDL	XML, RDF
Metadata Query	SPARQL	OAI	SQL, Oracle WS-DAI(X)	XML	XML
MPI	PBS, LSF Condor, SGE	-	SGE (NAF)	-	-
Middleware					
Toolkit	Globus	Globus	gLite	Globus	Globus
Job Submission	WS-Gram	WS-Gram	gLite	WS-Gram	Globus
Job Monitoring	Globus Cactus Timeline	Globus	lcg	Globus	Globus
Job Language	JDD/JSDL	JSDL	JDL	JSDL	JSDL
API	GAT	Globus	gLite WMS	GAT	GAT/SAGA
User Management					
VO Management	OSG VOMRS	-	VOMRS	VOMRS	VOMRS
Authentication	Certificate	UID, MyProxy	Certificate	Certificate	Certificate
Registration	GridKa DFN-RA Nationale CAs Lokale RAs	eigene RA	GridKa eigene RA	DFN-RA	DFN-RA
lokaler Access	GridMap	rollenbasiert	GridMap	-	-
Support	email JIRA	email PortalAdmin	email GGUS	email DGUS	email DGUS
Security					
Mechanismen	X.509, GSI	X.509, GSI Proxy GSI & GridShib	X.509, GSI Proxy	X.509, GSI Proxy	X.509, GSI
Revocation	EuGrid/PMA		EuGrid/PMA		
Monitoring	Globus	-			
Application					
Portal	GridSphere	GridSphere	ganga, dq2 gEclipse	GridSphere LifeRay	GridSphere gEclipse
Workflow		C3WSS	-	GWES	
WF Language		WSL	-	GWDL	

Die verwendeten Architekturen. Geplante Veränderungen sind farbig markiert.

5 Evaluierung der existierenden Architektur-Konzepte

Anhand der Auswertung der existierenden Architekturen ergibt sich ein recht einheitliches Bild der eingesetzten Komponenten und Services. Für eine grafische Zusammenfassung der Architekturen siehe Abbildung 7.

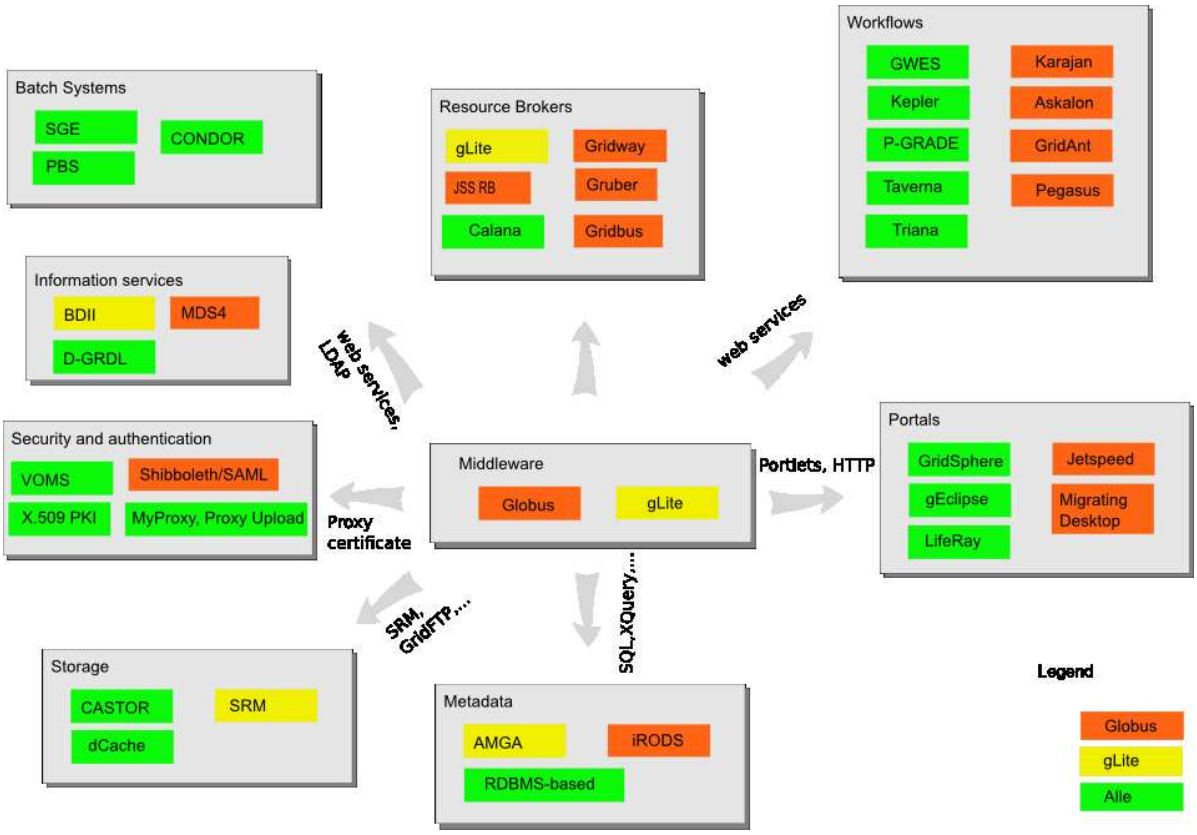

Abbildung 7: Grid Architektur Komponenten

5.1 Middleware

Globus ist mittlerweile der *de facto* Standard der Middleware Toolkits und vier der fünf Community Grids in WissGrid verwenden Globus. Einzige Ausnahme ist das HEPCG, das aufgrund der Einbettung in WLCG gLite und die damit verbundenen Services einsetzt. Unicore wird zwar im Rahmen der DGI-Referenzinstallation unterstützt, ist in WissGrid aber nicht vertreten und wird daher in den Konzepten nur eine untergeordnete Rolle spielen.

In Rahmen des AustrianGrids wurden gLite und Globus für eine medizinische Grid-Anwendung detailliert verglichen [63]. Dabei ergaben sich die folgenden Ergebnisse, die aber nicht repräsentativ sein müssen:

- gLite behauptet 'Interactive jobs' zu unterstützen, das funktioniert in Wirklichkeit aber nicht
- gLite hat einen besseren Resource Discovery Mechanismus als Globus (basiert auf JDL job description)

- Job-Submission-Overhead in gLite ist größer.
- Globus arbeitet effizienter, was aber partiell an den Unterschieden im Resource-Discovery liegt.
- *Our comparisons show that while the Globus Toolkit 4 is faster and more efficient, gLite is much more sophisticated and developer friendly.*

Die Auswertung zeigt darüber hinaus:

- gLite erlaubt Bulk-Job-Submission und parametrische Jobs, was in Globus nicht implementiert scheint.
- Integration von einigen Portals/Portlets und Workflows ist nur in Globus möglich.
- Unterstützung für Shibboleth ist besser in Globus.
- VO Management spielt kaum eine Rolle bei der Wahl der Middleware.
- Ein Vorteil von Globus ist, dass sehr viel kompatible Zusatz-Middleware für Globus verfügbar ist.

5.2 Data-Management & Metadata

Die WissGrid CGs setzen unterschiedlich Metadata-Standards und Protokolle ein, wie

- Open Archive Initiave (OAI) Standards (OAI ORE, PMH)
- Open Grid Services Architecture Data-Access and Integration (OGSA-DAI)
- W3 Resource Description Framework (RDF)
- AMGA
- iRods

Alle Systeme unterstützen (zumindest über Erweiterungen wie iCat oder WS-DAI) relationale wie XML Datenbanken. Während AMGA eine reines Metadaten-Backend ist, bieten DAI und iRods eine (vollständige) Datenmanagement-Umgebung. Mit Globus als Middleware bieten sich sicher iRods und DAI als Datenmanagement-System an, OGSA-DAI insbesondere, da es auch in der DGI Referenz-Installation [16] eingesetzt wird. AMGA verfügt über eine Vielzahl von APIs, die einfach in Experiment-nahe Applikationen eingebettet werden können. Für neue Grid Communities hängt die Wahl eines Metadaten- oder DM-Systems stark von den bereits vorhandenen Strukturen ab. Manche Photon Science Facilities verwenden zum Beispiel iSpyB, um Metadaten zu sammeln, zu verwalten und als Web-Service verfügbar zu machen. In diesem Falle könnte man leicht eine Schnittstelle zu AMGA generieren oder die Daten via Portlets im Grid verfügbar machen. Andere Photon Science Facilities verwenden iCat, so dass eine Integration mit iRods realisierbar wäre.

Eine wesentliche Rolle spielen die vorhandenen SRMs. dCache z.B. wird vielfach im HEP-Umfeld, aber auch im C3-Grid eingesetzt. Alle DM-Systeme erlauben den Zugriff auf die Daten via GridFTP/gsiFTP. Für sehr rechenintensive Datenanalysen, die sowohl im Grid als auch auf lokalen Resourcen ablaufen sollen, bieten Protokolle wie gsidcap oder NFS4 einen direkteren Zugriff auf die Daten. Die Existenz nativer Clients für alle gängigen Betriebssysteme (Linux, Windows, OSX) ist für einige Communities ein wichtiger Gesichtspunkt. iRods/iCat und Globus basierte Lösungen bieten dies an, dCache, Castor oder StoRM bislang nur mit Einschränkungen.

5.3 Authentifizierung / Authorisierung

Die meisten WissGrid-CGs verwenden Grid-Zertifikate zur Authentifizierung und Proxy-Delegation Mechanismen. Grid-Zertifikate erfreuen sich allerdings in manchen Communities nur begrenzter Beliebtheit. Integration von Shibboleth-basierter Authentifizierung und Authorisierung ist daher dringend gewünscht. GridShib [18] erlaubt den Austausch zwischen SAML/Shibboleth und X.509 PKI, allerdings nur im Globus Umfeld.

Switch bietet eine Integration von Shibboleth (oder Federated IdPs i.a.) und gLite an. Unterstützt werden sowohl Short Lived Credential Services (SLCS) als auch *VOMS Attributes from Shibboleth* (VASH) (siehe z.B. [19]). SLCS können ebenfalls an eine Organisation (via Kerberos-enabled CAs) gebunden werden, beide Mechanismen scheinen aber kaum Verwendung in gLite zu finden. Das DFN stellt in Form von DFN-AAI mit Shibboleth WAYF-Server ebenfalls eine Implementierung zur Verfügung. Darüber hinaus gibt es auch einen DFN-SLC-Service. Der EGEE-III Bericht zur gLite Security Architecture [20] geht davon aus, dass für die kommenden Jahre X.509 PKI die Basis der gLite Sicherheits-Architektur bleiben wird.

Zugang zu lokalen bzw. Instituts eigenen Resourcen wird in den meisten Fällen via GridMaps gewährt. Diese Lösung ist passabel, skaliert aber schlecht, was für grösse Communities wie die Photon Sciences ein gravierendes Problem darstellt.

5.4 Portals und Clients

Die meisten WissGrid CGs setzen GridSphere basierte Portals bzw. Clients ein. Eclipse basierte Clients (gEclipse) werden in HEPCG und TextGrid verwendet. gEclipse kann unabhängig von Middleware, Platform und Information Services eingesetzt werden.

Außerhalb der WissGrid Communites finden eine Vielzahl anderer Portals und Clients Verwendung. Jetspeed scheint unter diesen ein vielversprechender Kandidat, da Jetspeed den Einsatz generischer (semantischer) Portlets erlaubt und vergleichsweise einfach in Betrieb zu nehmen ist.

Sowohl Jetspeed als auch GridSphere basierte Portals scheinen bislang nur in Globus integrierbar.

5.5 Workflows

Workflows und Workflow Execution Services spielen nicht in allen Community-Grids die gleiche Rolle und werden daher nur in einigen der WissGrid CGs eingesetzt. Neben dem Einsatz von GWES (Grid Workflow Execution Service) im MediGRID und TextGrid setzt das C3Grid ein proprietäres Workflowmanagementsystem (C3Grid WSS) ein, um Workflows bestehend aus Datenmanagement-Operationen (Staging, Import, Export, Transfer) und Rechenjobs effizient zu verwalten.

Workflows wären für die prototypischen Prozesse in den Photon Sciences (Small Angle X-ray Scattering (SAXS) oder Macromolecular X-ray Crystallography (MX)) sehr nutzvoll, da die Analysen in wohldefinierte Frameworks eingebettet werden können.

Neben GWES hinterlässt das Kepler Workflow-System einen guten Eindruck, da es einfach zu handhaben und hinreichend flexibel einsetzbar scheint.

5.6 Prototypische Architekturen

In Hinblick auf konkrete Anforderungen, insbesondere aus den Photon Science Communities, lassen sich aus den generischen Architekturen in Abbildung 7 eine kleine Zahl protoypischer Architekturen extrahieren, die den Anforderungen genügen sollten. Die Anforderungen sind bislang nur grob skizziert und werden detaillierter an anderer Stelle dargestellt. Insbesondere beruhen die Anforderungen auf sehr vorläufigen Modellen zur Daten- und Compute-Infrastruktur oder Topologie und werden sicher im Laufe der Zeit Anpassungen erfahren. Die prototypischen Implementierungen dienen daher auch als Modell um gewisse Charakteristika zu evaluieren. Insbesondere die Photon Science Communities generieren Datenmengen, die vergleichbar mit denen des LHC im Vollbetrieb sein werden, wobei die Datenraten sehr viel variabler und zeitweise sehr viel grösser sein werden. Ob und welche Metadaten-Engines performant genug sind, die Anforderungen zu erfüllen, ist zum Beispiel ein wesentlicher Aspekt.

5.6.1 AstroGrid-D Architektur

Für die Architektur des AstroGrid wurden die Globus-Middleware Komponenten ergänzt bzw. erweitert, wo die nativen Globus-Komponenten nicht ausreichende Funktionalität geboten haben.

Hierbei sind folgende Ebenen zu unterscheiden (s. Abb. 8):

- Middleware-unabängig wurde der Informationsdienst Stellaris auf Basis von RDF/SPARQL implementiert. Damit ergab sich der Vorteil, daß der Information Server verschieden hohe Security-Anforderungen realisieren konnte, und mit einer Schnittstelle zu GSI ausgerüstet war. Client-Applikationen wurden realisiert, die mit verschiedenen Protokollen (HTTP/S, GSI) die gespeicherten Daten abrufen und weiterverarbeiten konnten (Timeline, Commandline Scripts, GridRessourceMap). Auch fuer die Einbindung der Robotischen Teleskope war dies eine wesentliche Erleichterung. Auch für das VO-Management sind middleware-unabhängige Dienste (VOMRS) und Java-Http Applets kombiniert worde, um das derzeitige D-Grid System hierfür zu realisieren.

- Eine Middleware-Erweiterung war, neben dem Paket für Robotische Teleskope, das AstroGrid-Datamanagement (ADM), welches auf dem Globus Replica Service aufbaute, und ein Peer-2-Peer Data-Stream Management.

- Eine Nutzung von Glosbus-Zusatzkomponenten wie dem Globus-Audit wurde zum Job-Monitoring ausgebaut, unter Einbeziehung von Stellaris. Hier ist jedoch insbesondere auch die Nutzung von GridWay als Grid-Metascheduler zu nennen.

- Einbindung vorhandener Komponenten: hier sind z.B. GAT und GridSphere und der Planck-Process-Coordinator (ProC), zu nennen, die jeweils fuer spezielle Applikationen (Cactus, ProC) Anwendung fanden.

Die allergrößte Zahl der UseCases in der Astronomie nutzt die Commandline, die Nutzung von standardiserten Web/Portal-Interfaces ist nur in Fällen von Interesse, in denen Daten aus astronomischen Archiven weiterverarbeitet werden.

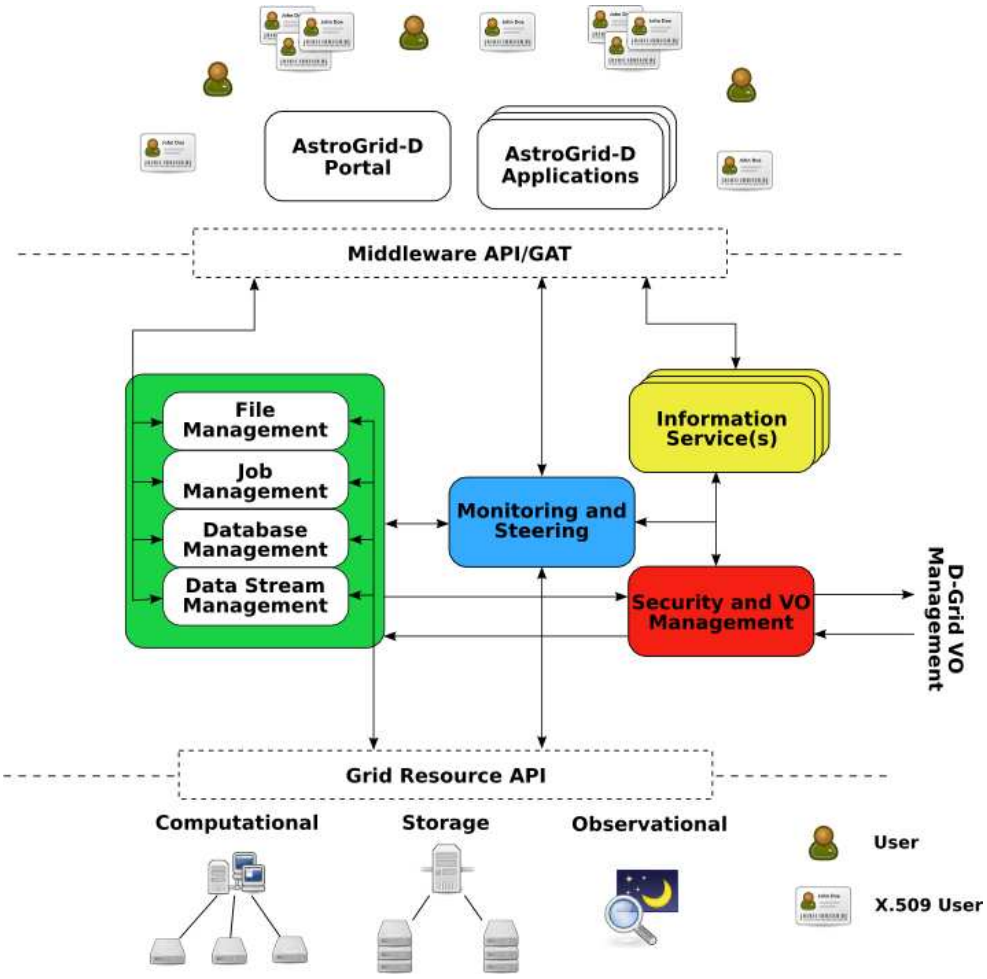

Abbildung 8: Architekturschema von AstroGrid-D für Grid-Komponenten.

Die Architektur des AstroGrid ist in einer Revisionsphase, wo der Fokus sich von Compute-zentrierten Grid-Job zu Datenmanagment und Data-zentriertem Grid-Job verlagert. Hier fehlen in der Architektur z.B. Elemente wie automatisierte Extraktion von Metadaten, eine Erweiterung und Verfeinerung des Metadatenschema für Grid-Dienste, ein verteiltes Data-Management sowie die Einbettung von Grid-Komponenten in Virtuelle Forschungsumgebungen.

5.6.2 Photon Science Architekturen

Anhand der Auswertung werden wir uns auf einige ausgewählte Komponenten beschränken. Einzig dCache als Storage System kann aufgrund der lokalen Umgebung zunächst als gegeben betrachtet werden (Abb. 9):

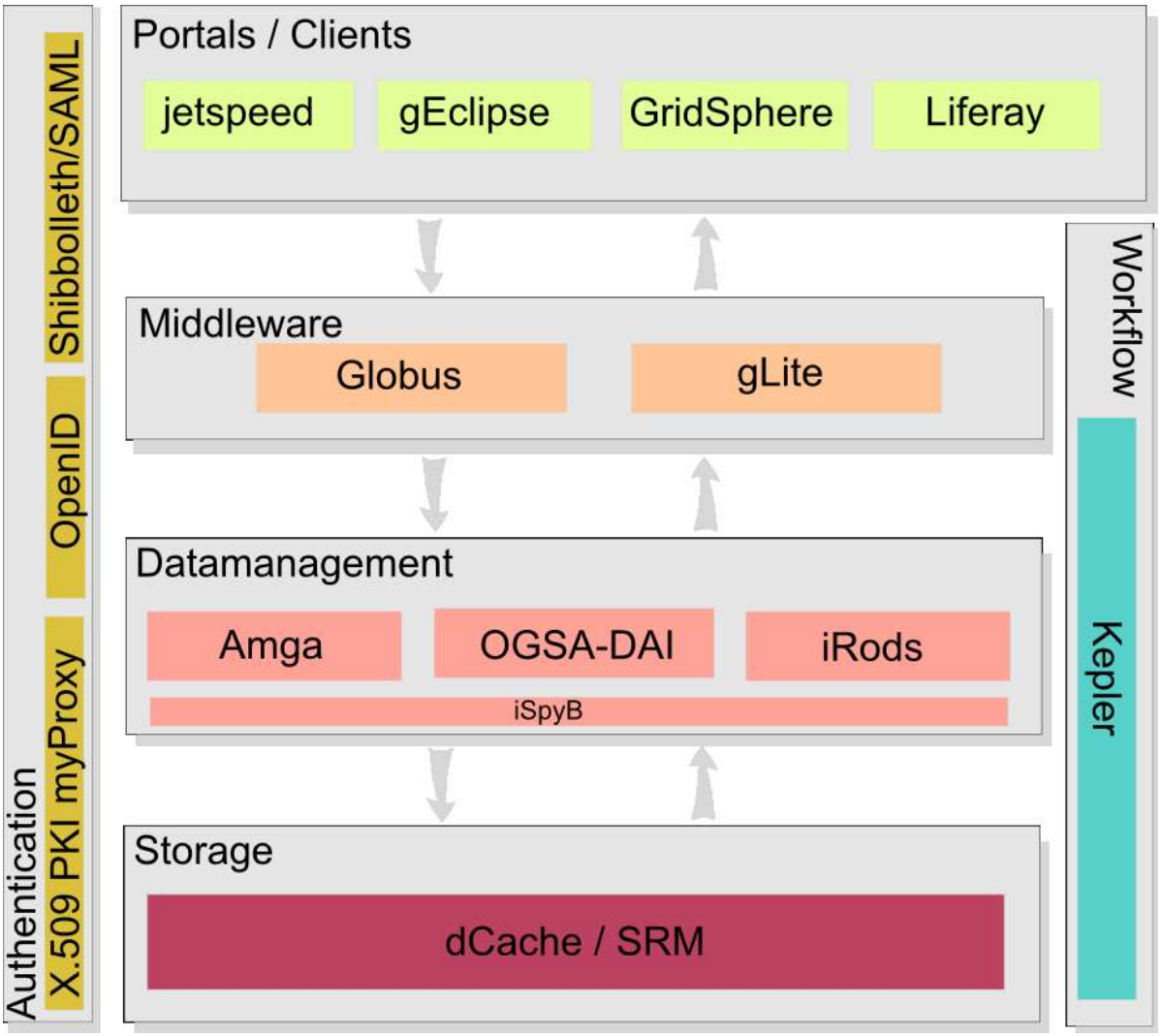

Abbildung 9: Architektur-Cluster für ein Photon Science CG

A Detaillierte Beschreibung von Grid-Diensten

A.1 Grid Workflow Execution Service (GWES)

Der Grid Workflow Execution Service interpretiert Workflows und bildet sie automatisch auf verschiedenartige Aktivitäten ab. Unter diese Aktivitäten fallen Globus Jobs, RFT Dateitransfers, SOAP Web Service-Aufrufe und Kommandozeilenprogramme. Die Anbindung weiterer Taskarten ist in Planung.

Ein Workflow kann unabhängig von der unterliegenden Infrastruktur in Form eines Petri-Netzes beschrieben werden. Die konstruierten Abhängigkeiten werden mithilfe der eigenen Workflowbeschreibungssprache GWorkflowDL XML-basiert abgelegt. Mit dieser Darstellungsform ist es sogar möglich, ganze Algorithmen als Workflows anzugeben und bearbeiten zu lassen. Typische Workflows modellieren Parameterstudien, sequentielle Taskfolgen oder komplexe Taskabfolgen inklusive Bedingungen und Schleifen.

Der GWES findet inzwischen in einigen Community-Grids des D-Gridverbundes Einsatz. Im Rahmen der WissGrid-Communities wurde die Software von MediGRID eingesetzt.

A.2 C3Grid Workflow Scheduling Service (WSS)

Der C3Grid Workflow Scheduling Service stellt einerseits Workflowmanagementfunktionalität zur Verfügung, und bietet gleichzeitig Schedulingfunktionalität für die effiziente Nutzung von Ressourcen sowie die Zusammenführung von Daten- und Rechen-bezogenen Jobs (Co-Scheduling). Im C3Grid-Kontext umfasst dies das Management komplexer wissenschaftlicher Workflows, welche aus Datenbeschaffung, -vorbereitung und -transfer sowie aus Jobausführung bestehen.

Abhängigkeiten zwischen atomaren Tasks eines Workflows werden über eine XML-basierte Beschreibungssprache (WSL - Workflow Specification Language) angegeben, die einzelne Jobbeschreibungen im JSDL-Format enthält. Daten-bezogene Tasks können über eine C3Grid-spezifische Taskbeschreibungssprache definiert und ebenfalls in die WSL eingebunden werden.

Die Architektur und Implementierung des Dienstes richtet sich nach der in C3Grid verwendeten Middleware Globus Toolkit 4.0.x. Der Scheduling- und Workflowmanagementdienst ist daher im Globus WSRFramework implementiert und läuft im Globus-Container. Damit passt er sich problemlos in die Globus-Infrastruktur ein. Wie in Abbildung 10 dargestellt, besteht der C3Grid WSS aus zwei Diensten, die in ihrer Zusammenwirkung den Workflow widerspiegeln und abarbeiten. Für jeden Workflow wird eine Instanz des JobManagementService erstellt, die die Workflowstruktur speichert sowie seine Ausführung überwacht und steuert, abhängig von den angegebenen Abhängigkeiten. Der TaskExecutionService stellt verschiedene Handler-Implementierungen zur Verfügung, die für die spezifische Ausführung von atomaren Tasks zuständig sind (z.B. Datenextraktion, Datentransfer, Jobausführung über WS-Gram usw.). Das Management und zugleich das Scheduling der Taskausführung wird über eine austauschbare (pluggable) Strategie gesteuert. Da alle Dienste entsprechend dem SOA-Prinzip lose gekoppelt sind, wird die Statusüberwachung nicht über Polling, sondern über den standardisierten WS Notification (WSN) Mechanismus durchgeführt.

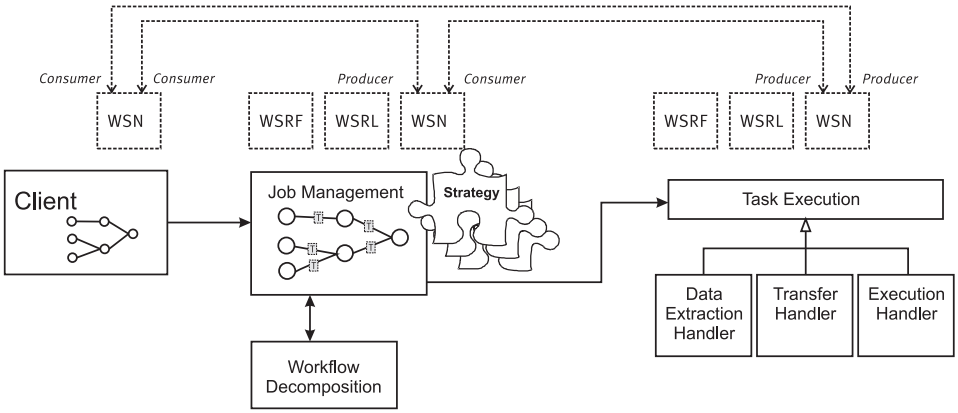

Abbildung 10: Schematische Darstellung der C3Grid WSS-Architektur.

Literatur

[1] C. Plevier: Grid Computing Lecture
http://www.astro.lu.se/ELSA/pages/PublicDocuments/Plevier.pdf

[2] GridCafe
http://www.gridcafe.org/

[3] DGI Referenz Installation
http://dgiref.d-grid.de

[4] Astrogrid Architecture Documentation
http://www.gac-grid.de/project-documents/architecture.html

[5] C3Grid Architecture Documentation
http://www.c3grid.de/index.php?id=54

[6] WLCG
http://lcg.web.cern.ch

[7] MediGrid
http://www.medigrid.de

[8] TextGrid Architektur
http://www.textgrid.de/fileadmin/TextGrid/reports/TextGrid_Report_3_2.pdf

[9] TextGrid
http://www.textgrid.de/

[10] The Anatomy of the Grid. (introduces VO's)
www.globus.org/alliance/publications/papers/anatomy.pdf

[11] The Physiology of the Grid.
www.globus.org/alliance/publications/papers/ogsa.pdf

[12] I. Foster and C. Kesselman.
The Grid: Blueprint for a new computing infrastructure,

[13] A presentation on Grid storage
https://twiki.grid.iu.edu/pub/Education/Syllabus/SRM.pdf

[14] RFC 3820
http://www.ietf.org/rfc/rfc3820.txt

[15] D-Grid. Interoperabilität und Integration der VOManagement Technologien im D-Grid.
http://dgi.d-grid.de/index.php?id=314

[16] D-Grid reference installation.
http://dgiref.d-grid.de/

[17] Siegfried Makedanz, Hans Pfeiffenberger: Shibboleth- Infrastruktur für das Grid. D-Grid Security Workshop 2006.
http://epic.awi.de/Publications/Mak2006b.pdf

[18] GridShib: Bridging SAML/Shibboleth and X.509 PKI for campus and grid interoperability.
http://gridshib.globus.org/

[19] Christoph Witzig, Interoperability Shibboleth - gLite.
Witzig_080219_ivom.pdf

[20] EGEE-III, G LITE SECURITY ARCHITECTURE
EGEE-III-MJRA1_4.pdf

[21] C.Grimm, M.Pattloch (Koordination). DGI Fachgebiet 3-4 - Aufbau einer AA-Infrastruktur für das D-Grid, Analyse von AA-Infrastrukturen in Grid-Middleware.

[22] Gap-SLC.
http://www.d-grid-ggmbh.de/index.php?id=93

[23] AMGA
amga.web.cern.ch/amga/

[24] Grid-Based Metadata Services, Ewa Deelman et al.
www.globus.org/alliance/publications/papers/deelman3.pdf

[25] iRods
https://www.irods.org/

[26] Alliance Portal Project http://www.extreme.indiana.edu/alliance/

[27] EarthSystemGrid
https://www.earthsystemgrid.org/

[28] Geodise
http://www.geodise.org

[29] GridSphere
http://www.gridsphere.org

[30] Jetspeed
http://portals.apache.org/jetspeed-2/

[31] Maozhen Li and Mark Baker, 'The Grid. Core technologies', Wiley 2005 (section Grid Portals)

[32] Maozhen Li and Mark Baker, A Review of Grid Portal Technology, in 'Grid Computing, software Environments and Tools', eds. J. Cunha, O. Rana, Springer 2006

[33] Microsoft Sharepoint Portal Server
http://sharepoint.microsoft.com/Pages/Default.aspx

[34] Mygrid
http://www.mygrid.info

[35] Oracle Portal Development Kit
http://www.oracle.com/technology/products/ias/portal/pdk.html

[36] Web ontology
http://en.wikipedia.org/wiki/Web_Ontology_Language

[37] P-GRADE portal
http://portal.p-grade.hu/

[38] Plumtree Corporate portal
http://www.plumtree.com/

[39] Vladimir Silva, 'Grid Computing for developers', CHARLES RIVER MEDIA 2006 (Section Grid portals)

[40] Sun One Portal Server
http://developers.sun.com/portalserver/reference/techart/ps_overview.html

[41] Viador E-Portal
http://www.viador.com/

[42] WS-BPEL
http://en.wikipedia.org/wiki/Business_Process_Execution_Language

[43] Websphere
http://www.ibm.com/websphere

[44] Liferay
http://www.liferay.com/

[45] UNICORE Rich Client
http://www.unicore.eu/unicore/architecture/client-layer.php

[46] Remote Component Environment
rce.html

[47] Norbert Podhorszki and Scott Klasky, 'Workflows in a secure environment', in Distributed and Parallel Systems. In Focus: Desktop Grid Computing, Springer 2008

[48] P-GRADE workflow
http://www.p-grade.hu/

[49] Several code parallelization with P-GRADE examples in Distributed and Parallel Systems: Cluster and Grid Computing, Springer 2005

[50] Workflows
http://www.gridworkflow.org/

[51] A taxonomy of grid workflows
http://www.buyya.com/papers/WorkflowTaxonomy-JoG.pdf

[52] Taverna workflow
http://www.taverna.org.uk/

[53] http://www.gridworkflow.org/snips/gridworkflow/space/GWES

[54] GWES 2.0 tutorial
Hoheisel_2009_GWES-Tutorial_de.pdf

[55] Kepler workflow
https://kepler-project.org

[56] Triana workflow
http://www.trianacode.org/

[57] Clouds Lab, Grid for business technologies
http://www.gridbus.org/

[58] Jun Qin et al., UML BASED GRID WORKFLOW MODELING UNDER ASKALON, in Distributed and Parallel Systems, Springer 2007

[59] A. Kertesz, P. Kacsuc, A taxonomy of Grid resource brokers, in Distributed and Parallel Systems, Springer 2007

[60] gLite
http://glite.web.cern.ch/glite/, http://en.wikipedia.org/wiki/GLite

[61] Globus
http://www.globus.org/toolkit/

[62] Unicore
http://www.unicore.eu/

[63] Globus vs. gLite for a medical grid use-case
AG-D4-1-2007_2.pdf

[64] ATLAS Computing TDR
http://cdsweb.cern.ch/record/837738/files/lhcc-2005-022.pdf

[65] CMS Computing TDR
http://cdsweb.cern.ch/record/838359/files/lhcc-2005-023.pdf