

INSTITUT FÜR INFORMATIK
DER LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN



Diplomarbeit

Entwicklung eines Ansatzes zur feingranularen Rechte-Delegation in Globus 5-basierten Grids

Christian Schulz

INSTITUT FÜR INFORMATIK
DER LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN



Diplomarbeit

Entwicklung eines Ansatzes zur
feingranularen Rechte-Delegation
in Globus 5-basierten Grids

Christian Schulz

Aufgabensteller: Prof. Dr. Kranzlmüller
Betreuer: Dr. Schiffers
Abgabetermin: 27. Dezember 2011

Hiermit versichere ich, dass ich die vorliegende Diplomarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 25. Dezember 2011

.....
(Unterschrift des Kandidaten)

Abstract

Der Bereich Grid Computing gewinnt im Feld der Informatik immer mehr an Bedeutung. Sogenannte *Grand Challenges* aus den Bereichen Wissenschaft und Ingenieurwesen sind nur mit Hilfe vieler, gekoppelter High Performance Computing Ressourcen zu lösen. Grid Middlewares, wie das Globus Toolkit, ermöglichen das Arbeiten in solchen Umgebungen. Durch das steigende Interesse an Grids, beispielsweise auch im industriellen Bereich, gewinnt dabei auch das Thema Sicherheit mehr und mehr an Bedeutung.

Die hier vorgelegte Arbeit basiert auf den Erkenntnissen eines Forschungspraktikums zum Thema *Sicherheit von Proxy Zertifikaten bei der Verwendung im Globus Toolkit* und greift einen der darin beschriebenen Sicherheitsmängel auf, um eine Lösung zu finden.

Im Globus Toolkit, wie auch in vielen anderen Middlewares, fehlen Mechanismen zur feingranularen Begrenzung delegierter Rechte. Die Beschränkung der weitergegebenen Rechte auf bestimmte Aufgaben oder Jobs ist nicht vorgesehen. In dieser Arbeit werden die daraus resultierenden Anforderungen genau analysiert und darauf aufbauend ein Ansatz entwickelt, der die gewünschten Mechanismen in Globus Toolkit 5-basierten Grids realisiert und dort eine regelbasierte Begrenzung delegierter Rechte ermöglicht.

Inhaltsverzeichnis

1	Einleitung	1
1.1	Motivation	1
1.2	Problem	2
1.3	Lösungsansatz	4
1.4	Vorgehensweise	4
2	Grundlagen	7
2.1	Globus Toolkit	7
2.2	GSI	8
2.2.1	PKI	8
2.2.2	SSL/TLS	9
2.2.3	Proxy Credentials	10
2.3	XACML	13
2.3.1	Framework	13
2.3.2	Datenstrukturen	14
2.4	Resource Specification Language	18
3	Analyse	21
3.1	Authentifizierung und Autorisierung - Ist-Zustand	21
3.1.1	Erkenntnisse aus dem Fortgeschrittenenpraktikum	21
3.1.2	Standardszenario	22
3.2	Anforderungen	23
3.2.1	Ziele und Systemgrenzen	23
3.2.2	Informelle Anforderungsbeschreibungen	23
3.2.3	funktionale Anforderungen und Use Cases	29
3.2.4	Nicht-funktionale Anforderungen	33
4	Verwandte Arbeiten	35
4.1	Delegation mit Proxies in anderen wichtigen Middlewares	35
4.1.1	Globus Toolkit 4/5	35
4.1.2	gLite	36
4.1.3	ARC Middleware	37
4.2	Unicore und feingranulare Delegation	40
4.3	Evaluation	42
4.3.1	Globus Toolkit 4/5	42
4.3.2	gLite	43
4.3.3	ARC	43
4.3.4	Unicore	44
5	Entwurf	45

5.1	Sicherheitskonzept - nötige Änderungen (sprachunabhängig)	45
5.1.1	Client - Erstellung und Einbringung von Policies	45
5.1.2	Authentifizierung und Autorsierung	47
5.1.3	Zusammenfassung	51
5.2	Policy-Sprache	52
5.2.1	Grundlagen	52
5.2.2	Kategorien von Beschränkungen	54
5.2.3	Eigene Policy-Sprache	55
5.3	PDP - Entscheidungsalgorithmus	59
5.4	Programmablaufpläne	60
5.4.1	Methodik	61
5.4.2	Anwendung	61
6	Implementierung	67
6.1	Umgebung und verwendete Technologien	67
6.1.1	Globus Toolkit 5	67
6.1.2	Betriebssystem	67
6.1.3	Programmiersprache	68
6.2	Implementierte Komponenten	69
6.2.1	Tool zur Policy-Definition	69
6.2.2	Sammeln und Zusammenfassen von Policies	71
6.3	Anmerkungen zu weiteren Komponenten	77
6.3.1	GRAM-Client und Kommunikation zum Gatekeeper	77
6.3.2	Gatekeeper - Authentifizierung und Autorisierung	78
7	Evaluation	79
7.1	Ansatz	79
7.2	Implementierung	81
7.3	Fazit der Evaluation	82
8	Zusammenfassung und Ausblick	83
8.1	Zusammenfassung	83
8.2	Ausblick	84
8.2.1	Vollständige Implementierung des Ansatzes	84
8.2.2	Mögliche Erweiterungen	85
	Abbildungsverzeichnis	87
	Literaturverzeichnis	89

1 Einleitung

Der Bedarf an Rechenzeit auf High Performance Computing Ressourcen ist in den letzten Jahren dramatisch angestiegen. Dies ist exemplarisch an den Anfragen bei der DEISA Extreme Computing Initiative (DECI)¹ zu erkennen. Waren es im Jahr 2005 noch Anfragen für 30 Millionen CPU-Stunden, so stieg die Zahl im Jahr 2008 bereits auf 134 Millionen und 2010 auf 570 Millionen CPU-Stunden [Led10, Folie 11f].

Sowohl Forschung, als auch Industrie, haben Aufgaben zu bewältigen, die der Verwendung von Hochleistungsrechnern bedürfen. Diese fundamentalen Aufgaben, genannt 'Grand Challenges', sind in den verschiedensten Bereichen von Wissenschaft und Ingenieurwesen angesiedelt. Es handelt sich hierbei beispielsweise um Themen der Bereiche Klimawandel und Bevölkerungsentwicklung, die mit Hilfe vieler, gekoppelter Rechenressourcen in Echtzeit verstanden werden sollen. Aber auch in Bereichen wie Biologie und Gesundheitswesen gibt es jede Menge derartiger Probleme.

Eine neue Form des wissenschaftlichen Arbeitens spielt hier eine entscheidende Rolle. Hinter dem Begriff e-Science verbirgt sich eine Kombination aus Theorie, Experimenten und den dazugehörigen Simulationen. Für die dabei notwendige Datenanalyse sind einzelne Rechen-systeme häufig nicht ausreichend.

1.1 Motivation

Durch Grid Umgebungen wird die Möglichkeit geschaffen, Ressourcen dynamisch, in virtuellen, multi-institutionellen Organisationen zu teilen. Ein Arbeiten über Organisationsgrenzen hinweg, ohne zentrale Kontrolle, ermöglicht es hierbei einer größeren Zahl von Nutzern, High Performance Ressourcen zu verwenden.

Die Werkzeuge für den Aufbau von Grid Umgebungen, und für das Arbeiten mit selbigen, bieten Grid Middlewares. Zu den bekanntesten gehören Unicore², gLite³ und das Globus Toolkit (GT)⁴, mit dem sich diese Arbeit beschäftigt. Es handelt sich hierbei um eine Sammlung von Open Source Tools, die von der Globus Alliance⁵ entwickelt werden. Aktuell liegt das Toolkit in Version 5 vor.

Durch das wachsende Interesse an Gridlösungen wird natürlich auch das Thema Sicherheit immer wichtiger. Trotz fehlender zentraler Kontrolle über die beteiligten Ressourcen, soll ein größtmögliches Maß an Sicherheit erreicht werden. Zumal es sich bei den benutzten und produzierten Daten, vor allem in der freien Wirtschaft, häufig um sehr sensibles Material handelt. So ist ein Autobauer, der Simulationen für sein neuestes Modell berechnen lassen will, natürlich darauf bedacht, dass kein unbefugter Zugriff auf seine Daten und Ergebnisse möglich ist.

¹www.deisa.eu/science/deci

²www.unicore.eu

³glite.cern.ch

⁴www.globus.org/toolkit/

⁵www.globus.org/alliance/

1 Einleitung

Beim Arbeiten in Grids kommt es zu Szenarien, die eine dynamische Delegation von Nutzerrechten voraussetzen. Dies ist zum Beispiel der Fall, wenn ein Job an eine Ressource geschickt wird, zu dessen Ausführung Daten von anderer Stelle geholt werden müssen. Die Ressource muss in diesem Fall die Möglichkeit haben, als Stellvertreter des Nutzers aufzutreten und in dessen Namen das benötigte Material von anderen Ressourcen zu holen.

Im Globus Toolkit wird die dynamische Delegation mit Hilfe von Proxy Zertifikaten erreicht. Diese ermöglichen außerdem einen Single Sign-On im Grid. In einem Forschungspraktikum [Sch10] wurde die Sicherheit von Proxy Zertifikaten im Globus Toolkit untersucht. Dabei wurden mögliche Sicherheitsschwachstellen gesammelt und überprüft. Als größtes Defizit stellte sich dabei, neben der mangelnden Transparenz, die fehlende Möglichkeit zur feingranularen Beschränkung delegierter Rechte heraus.

Das Globus Toolkit bietet durchaus gewisse Einschränkungsmöglichkeiten. So gibt es neben den 'full' Proxys, die die kompletten Rechte des Ausstellers erben, auch sogenannte 'limited' Proxys, denen das Recht zum Starten neuer Prozesse fehlt. Außerdem kann die maximale Länge einer Proxy Kette und die Gültigkeitsdauer bei der Erstellung des Zertifikats festgelegt werden.

Es ist jedoch bisher nicht möglich, die delegierten Rechte auf bestimmte Ressourcen und mögliche Aktionen zu begrenzen. Dies würde erlauben, ein Proxy Zertifikat auf eine bestimmte Aufgabe, einen Job, zu zuschneiden.

1.2 Problem

In Public Key Infrastrukturen (PKIs) hat die Geheimhaltung des privaten Schlüssels höchste Priorität. Proxy Credentials, also das Proxy Zertifikat und der dazugehörige private Schlüssel, sind dabei aufgrund des fehlenden Passwortschutzes besonders gefährdet.

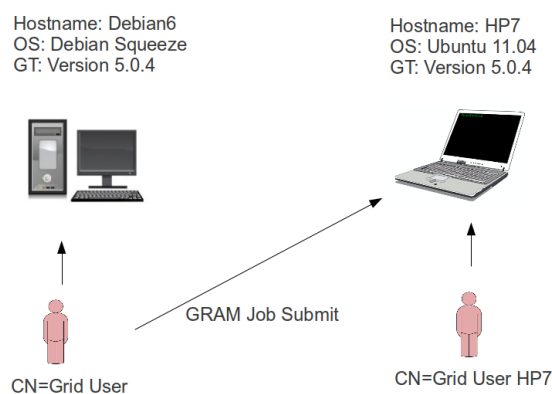


Abbildung 1.1: Problembeschreibung

Um einen Single Sign-On (SSO) zu ermöglichen ist der private Schlüssel nur durch Dateisystemberechtigungen geschützt und daher besonders anfällig für Angriffe. Hier kommt die Problematik vollständig delegierter Rechte ins Spiel. Zwar haben die Proxy Zertifikate kürzere Laufzeiten, als End Entity Certificates (EEC), während dieser Zeit kann ein Angreifer jedoch schon erheblichen Schaden anrichten. Er kann sämtliche Aktionen auf allen Ressourcen im Namen des ursprünglichen Ausstellers des Proxy ausführen, die diesem grundsätzlich erlaubt sind. Dass es hier tatsächlich zu Problemen kommen kann, ist bereits an einem kleinen, stark vereinfachten Beispiel zu erkennen. Anhand der Testumgebung aus Abbildung 1.1 kann es konstruiert werden.

'Grid User HP7' am Rechner 'HP7' erstellt mit dem Befehl `grid-proxy-init` ein neues, selbst-signiertes Proxy Zertifikat, abgeleitet von seinem EEC. Ohne zusätzliche Angaben hat dieses Zertifikat eine Laufzeit von 12 Stunden. Der User will nun auf der entfernten Ressource 'debian6' einen längeren Job ausführen, der wiederum einen anderen job startet. Dafür gibt er dem Job einen 'full Proxy' mit auf den Weg. Der Job wird im Beispiel mit einem kleinen Bash-Skript simuliert, das in Listing 1.1 zu sehen ist. Zuerst wird ein neuer Grid Job gestartet, der lediglich die Informationen über das verwendete Proxy Zertifikat ausgibt, dann wird der Prozess für 2 Stunden schlafen gelegt, um die lange Ausführungsdauer zu simulieren:

```

1 #!/bin/bash
2 /usr/local/globus-5.0.4/bin/globusrun -s -r debian6.fritz.box
   :2119/jobmanager-fork '&(executable=/usr/local/globus-5.0.4/
   bin/grid-proxy-info)';
3 sleep 7200;

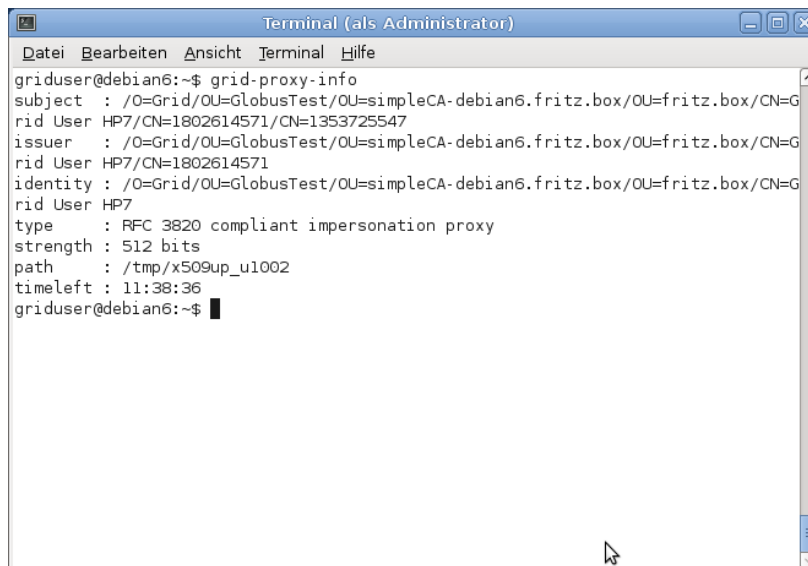
```

Listing 1.1: Job-Skript

Beim Submit des Jobs wird auf 'debian6' ein Proxy erstellt, der im Homeverzeichnis des Nutzers unter `~/.globus/job/<ressource>/<id>/x509_user_proxy` abgelegt wird. Dieser kann aufgrund der Dateisystemrechte nur vom anlegenden Nutzer gelesen und geschrieben werden, andere User haben im Normalfall keinen Zugriff. Wird das Credential allerdings durch einen Angriff gestohlen, beziehungsweise kopiert, so kann es vom Angreifer, auch nach Beendigung des eigentlichen Jobs, bis zum Ablauf seiner Gültigkeitsdauer benutzt werden.

Im Beispiel wird der Proxy vom root-Benutzer nach `/tmp/x509up_u1002` kopiert und der Systemuser 'griduser' mit der uid 1002 als Besitzer der Datei eingetragen. Führt dieser nun den Befehl `grid-proxy-info` aus, so wird das eben kopierte Proxy Zertifikat als sein eigenes angezeigt. Wie in Abbildung 1.2 zu sehen, lautet der Common Name (CN) des Zertifikats 'Grid User HP7', der CN des ursprünglich abschickenden Users.

Mit dem 'griduser'-Account können nun im Namen von 'Grid User HP7' Jobs abgeschickt werden. In Abbildung 1.3 wird ein Job an HP7 geschickt, der einfach Infos über das verwendete Proxy Zertifikat gibt. Der User kann also für den Rest der Gültigkeitsdauer mit dem geklauten Proxy Befehle im Namen eines anderen ausführen. Der ursprüngliche Nutzer hat das Zertifikat mit seinem EEC unterschrieben und ist deshalb auch für die damit verursachten Kosten verantwortlich. Dabei kann es sich um genutzte Rechenzeit handeln, aber auch um finanzielle Mittel im Falle eines derartigen Abrechnungssystems.



```
Terminal (als Administrator)
Datei Bearbeiten Ansicht Terminal Hilfe
griduser@debian6:~$ grid-proxy-info
subject : /O=Grid/OU=GlobusTest/OU=simpleCA-debian6.fritz.box/OU=fritz.box/CN=G
rid User HP7/CN=1802614571/CN=1353725547
issuer  : /O=Grid/OU=GlobusTest/OU=simpleCA-debian6.fritz.box/OU=fritz.box/CN=G
rid User HP7/CN=1802614571
identity : /O=Grid/OU=GlobusTest/OU=simpleCA-debian6.fritz.box/OU=fritz.box/CN=G
rid User HP7
type    : RFC 3820 compliant impersonation proxy
strength : 512 bits
path    : /tmp/x509up_u1002
timeleft : 11:38:36
griduser@debian6:~$
```

Abbildung 1.2: grid-proxy-info: Gestohlenes Zertifikat

1.3 Lösungsansatz

Es wird nun ein Ansatz für Globus Toolkit 5 entwickelt, um die mit Hilfe von Proxy Zertifikaten delegierten Rechte genauer beschränken zu können. Es soll zum Beispiel möglich sein, die delegierten Rechte auf einen speziellen GRAM-Job zuzuschneiden, um die Gefahren, die vom verwendeten Zertifikat ausgehen, zu minimieren. Einschränkungen auf bestimmte Zielressourcen, Executables und andere GRAM-Aktionen sollen dabei möglich sein.

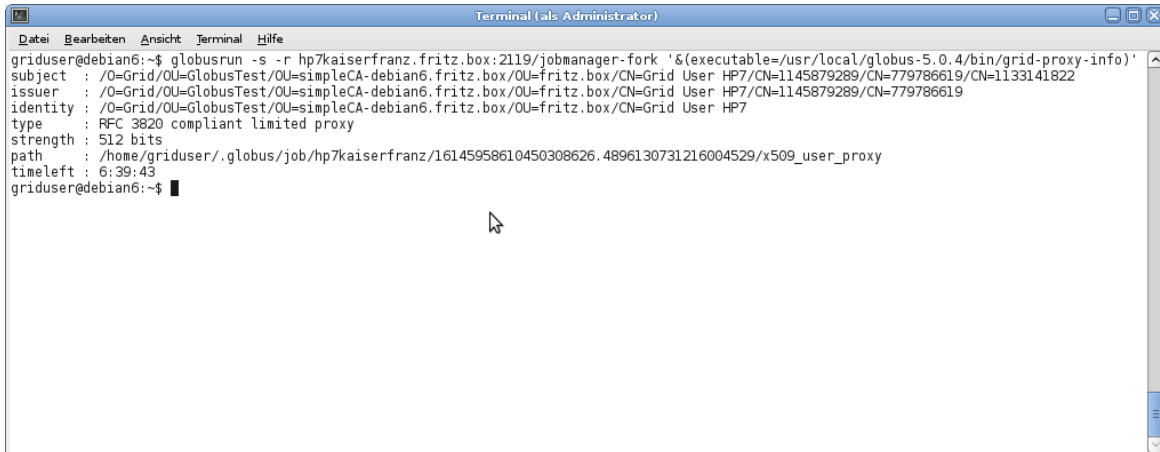
Fehlen die dafür nötigen Informationen zum Zeitpunkt des Job-Submit, so soll weiterhin die Möglichkeit bestehen, einen 'full'-, oder 'limited' Proxy zu verwenden. Die bestehenden Mechanismen zur Delegation von Nutzerrechten sollen also uneingeschränkt erhalten werden.

1.4 Vorgehensweise

Es muss zuerst geklärt werden, welcher Weg grundsätzlich eingeschlagen wird. Soll ein komplett neuer Ansatz zur Delegation von Nutzerrechten im Globus Toolkit entwickelt, oder der bestehende GT Ansatz, um die nötigen Mechanismen zur feingranularen Begrenzung der delegierten Rechte, erweitert werden? Die Entscheidung fiel dabei recht schnell auf die Erweiterung des bestehenden Ansatzes.

Die Verwendung von Proxy Zertifikaten ist in vielen Komponenten des Globus Toolkit fest verankert und ist ein wesentlicher Bestandteil des Globus Sicherheitskonzepts. Mit Version 4 des Toolkit wurden Proxy Zertifikate eingeführt, die Komplet konform zum RFC 3820[TWE⁺] der Internet Society⁶ sind. Seit Version 4.2 sind diese Standard. RFC 3820 Proxy Zertifikate bieten grundsätzlich die Möglichkeit, die delegierten Rechte über Policies zu Beschränken. Erreicht wird dies durch eine zusätzliche Erweiterung, die sogenannte Proxy Certificate Information (PCI) Erweiterung, die in jedem Proxy Zertifikat enthalten sein muss.

⁶<http://www.isoc.org>



```

Terminal (als Administrator)
griduser@debian6:~$ globusrun -s -r hp7kaiserfranz.fritz.box:2119/jobmanager-fork '&(executable=/usr/local/globus-5.0.4/bin/grid-proxy-info)'
subject : /O=Grid/OU=GlobusTest/OU=simpleCA-debian6.fritz.box/OU=fritz.box/CN=Grid User HP7/CN=1145879289/CN=779786619/CN=1133141822
issuer   : /O=Grid/OU=GlobusTest/OU=simpleCA-debian6.fritz.box/OU=fritz.box/CN=Grid User HP7/CN=1145879289/CN=779786619
identity : /O=Grid/OU=GlobusTest/OU=simpleCA-debian6.fritz.box/OU=fritz.box/CN=Grid User HP7
type     : RFC 3820 compliant limited proxy
strength : 512 bits
path     : /home/griduser/.globus/job/hp7kaiserfranz/16145958610450308626.4896130731216004529/x509_user_proxy
timeleft : 6:39:43
griduser@debian6:~$

```

Abbildung 1.3: Job-Submit mit gestohlenem Zertifikat

Sie besteht aus einem Pflichtfeld und 2 optionalen Feldern. Optional sind das *pCPathLen-Constraint* Feld, welches die maximale Länge des Proxy Pfades festlegt und das *proxyPolicy* Feld, dass nur in bestimmten Fällen, abhängig vom Pflichtfeld *policyLanguage*, vorhanden ist. Zur Begrenzung der delegierten Rechte werden die beiden letztgenannte Felder benötigt. Das *policyLanguage* Feld kann auf verschiedene Arten verwendet werden. Entweder kann damit festgelegt werden, dass alle, beziehungsweise keine Rechte delegiert werden, oder es wird der Object Identifier (OID) der Policy Sprache angegeben, mit der im *proxyPolicy* Feld Regeln bezüglich der Delegation der Rechte festgelegt werden sollen.

Die Sprache wird dabei nicht vorgegeben, die Wahl ist den beteiligten Parteien überlassen. Grammatik und zugehörige Semantik müssen vereinbart werden. Im Globus Toolkit wurden Begrenzungen dieser Art aber bisher nicht umgesetzt.

Die Tatsache, dass Proxy Zertifikate ein fester Bestandteil vieler Komponenten des Globus Toolkit sind und sie auch die notwendigen Voraussetzungen für feigranulare, regelbasierte Beschränkungen bieten, führt zur Entscheidung, den in GT bestehenden Ansatz zu erweitern und keinen vollständig neuen zu entwickeln.

Die Policies zur Beschränkung der Rechte können also mit Hilfe einer beliebigen Policy-Sprache über die PCI Erweiterung in die Proxy Zertifikate eingefügt werden und müssen dann auf der Job-Empfänger-Seite, bezüglich der gestellten Anfrage, ausgewertet werden. Dabei müssen die Policies der gesamten Proxy-Kette gesammelt und berücksichtigt werden. Je nach Ergebnis der Auswertung soll der gewünschte Job dann ausgeführt, oder abgelehnt werden.

2 Grundlagen

In diesem Kapitel sollen die wichtigsten Begriffe im Umfeld der Arbeit kurz betrachtet und beschrieben werden.

2.1 Globus Toolkit

Beim Globus Toolkit handelt es sich um eine Sammlung von Tools, die zum Aufbau von Grid Umgebungen und dem Arbeiten mit selbigen genutzt wird. Es handelt sich hierbei um Open Source Software, die unter anderem von der Globus Alliance entwickelt wird. Am 18. Mai 2011 wurde die aktuelle Version 5.0.4 veröffentlicht. Bauten viele Komponenten der Version 4 auf dem Web Service Resource Framework (WSRF)¹ auf, so ist dies in Version 5 nicht mehr der Fall. Das Execution Management wird hier beispielsweise von Grid Resource Allocation and Management 5 (GRAM5) übernommen, das keinen Web-Service-orientierten Ansatz mehr verfolgt, sondern auf dem Code von pre-ws GRAM2 basiert.

In Abbildung 2.1 ist der Aufbau von GT5 zu sehen. Unterteilt wird hier in die Bereiche Security, Data Management, Execution Management und Common Runtime.

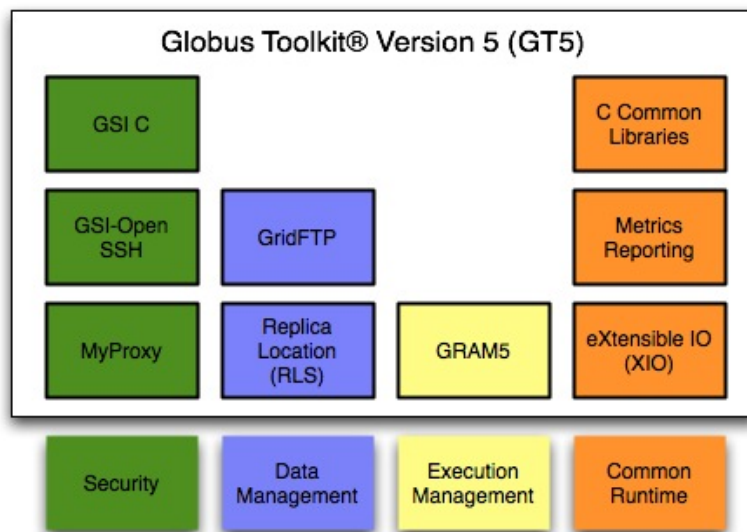


Abbildung 2.1: Globus Toolkit 5 - Komponenten

Das Hauptaugenmerk wird bei der Erweiterung des Delegations-Ansatzes auf den Bereichen Sicherheit und Execution Management liegen, da diese die für die vorliegende Problemstellung relevanten Komponenten enthalten.

¹http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf

2.2 GSI

Der Kern des Globus Sicherheitskonzepts ist die Grid Security Infrastructure (GSI) ², die vom Globus Toolkit implementiert wird. In Abbildung 2.2 ist der schematische Aufbau gezeigt. Die einzelnen Komponenten Secure Socket Layer, Public Key Infrastrukturen und Proxy Zertifikate werden im Folgenden genauer betrachtet.

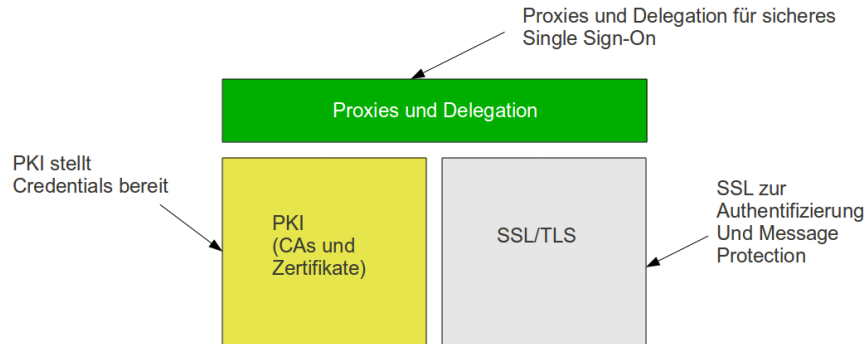


Abbildung 2.2: Grid Security Infrastructure

2.2.1 PKI

Mit Hilfe einer Public Key Infrastruktur (PKI) werden digitale Zertifikate ausgestellt, verteilt und geprüft. Nutzer und Hosts benötigen für Zugang und Interaktion in GT Umgebungen Zertifikate nach dem X.509 Standard. X509 v3 Zertifikate sind in RFC 3280[HLP+] spezifiziert. In Listing 2.1 ist die Textdarstellung eines solchen Zertifikats zu sehen. Wichtige Elemente sind die Distinguished Names von Aussteller und Subjekt des Zertifikats, sowie dessen Gültigkeitsdauer und der zugehörige öffentliche Schlüssel. Außerdem sind Signatur- und Verschlüsselungsalgorithmen, sowie eventuelle Erweiterungen angegeben.

```

1 Certificate:
2   Data:
3     Version: 3 (0x2)
4     Serial Number: 7 (0x7)
5     Signature Algorithm: sha1WithRSAEncryption
6     Issuer: O=Grid, OU=GlobusTest, OU=simpleCA -
7             debian6.fritz.box, CN=Globus Simple CA
8     Validity
9       Not Before: Aug 17 13:43:39 2011 GMT
10      Not After : Aug 16 13:43:39 2012 GMT
11     Subject: O=Grid, OU=GlobusTest, OU=simpleCA -
12             debian6.fritz.box, OU=fritz.box, CN=Grid User HP7

```

²www.globus.org/security/overview.html

```

13     Subject Public Key Info:
14         Public Key Algorithm: rsaEncryption
15             Public-Key: (1024 bit)
16             Modulus:
17                 00:ca:61:82:c6:b8:39...
18             Exponent: 65537 (0x10001)
19     X509v3 extensions:
20         Netscape Cert Type:
21             SSL Client, SSL Server, S/MIME, Object Signing
22     Signature Algorithm: sha1WithRSAEncryption
23         90:d6:f0:58:3e:c6...

```

Listing 2.1: Textdarstellung X509v3 Zertifikat

Derartige Zertifikate müssen bei einer Registration Authority (RA) beantragt und von einer vertrauenswürdigen Certificate Authority (CA) signiert werden. Die RA ist dabei für die Identifizierung des Zertifikatnehmers zuständig. Soll das Zertifikat beispielsweise an eine Person ausgestellt werden, so muss diese bei der Registration Authority einen amtlichen Lichtbildausweis vorlegen. Ist dies geschehen, so gibt die RA den Antrag an die CA weiter, diese stellt ein von ihr signiertes Zertifikat aus. Die so erstellten Zertifikate werden End Entity Certificates (EEC) genannt.

Der in GT verwendete X509 Standard für PKIs setzt einen strikt hierarchischen Aufbau voraus. Es gibt also eine Root-CA, der alle beteiligten Parteien vertrauen und darunter weitere CAs, deren Zertifikate jeweils von der Wurzel-CA signiert sind. In Abbildung 2.3 ist der hierarchische Aufbau der vom DFN-Verein³ betriebenen PKI zu erkennen. Ganz oben steht dabei das 'Deutsche Telekom Root CA 2' Wurzelzertifikat. Darunter liegt die DFN PKI. CA Zertifikate in der folgenden Ebene, auf der sich beispielsweise die CA des Leibniz Rechenzentrums befindet, werden wiederum von DFN CA Zertifikaten signiert.

Ein weiterer Mechanismus einer PKI sind Certificate Revocation Lists (CRL). Diese sind ebenfalls in RFC 3280[HLP⁺] spezifiziert und enthalten Zertifikate, die vor Ablauf ihrer Gültigkeitsdauer zurückgezogen wurden. Dies kann aufgrund kompromittierter Schlüssel, oder auch in Folge von ungültigen Zertifikatsdaten geschehen. Beim Verlassen einer Organisation kann das betroffene Zertifikat ebenfalls auf die CRL gesetzt werden. Bei der Überprüfung der Validität von Zertifikaten werden die Sperrlisten kontrolliert und ungültige Zertifikate somit erkannt.

Certificate Revocation Lists haben festgelegte Laufzeiten. Sie müssen regelmäßig aktualisiert werden, damit auch kürzlich zurückgezogene Zertifikate erkannt werden. Die Aktualität der CRLs ist also ein wichtiger Sicherheitsfaktor in Umgebungen, die auf Public Key Infrastrukturen basieren.

2.2.2 SSL/TLS

Jeder Transaktion in einer GT Umgebung geht eine wechselseitige Authentifizierung der beteiligten Entitäten voraus. Dies geschieht unter Verwendung von SSL und der eben eingeführten EECs, beziehungsweise der von ihnen abgeleiteten Proxy Zertifikate, die in Kapitel 2.2.3 noch genauer besprochen werden.

³<http://www.dfn.de/verein/>

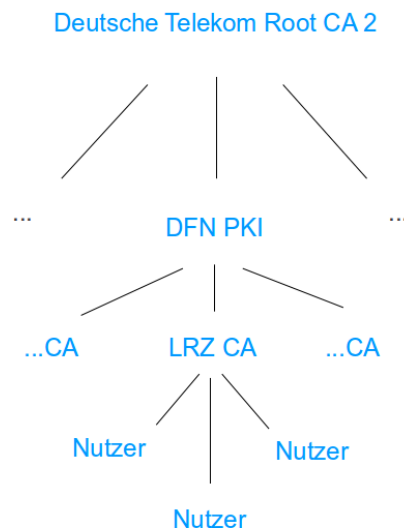


Abbildung 2.3: DFN-PKI

In Abbildung 2.4 ist der Ablauf dieses Vorgangs zu erkennen. Kommunikationspartner X schickt zu Beginn Kommunikationspartner Y sein Zertifikat. Bei einem EEC muss nur die Signatur der CA verifiziert werden. Bei einem Proxy Zertifikat kann die Überprüfung mehrerer Signaturen nötig werden, die gesamte Proxy-Kette entlang und am Ende die Signatur der CA. Bei erfolgreicher Verifizierung schickt Y eine zufällige Phrase, den *Challenge String*, zu X. Auf der anderen Seite wird die Phrase mit dem privaten Schlüssel von X verschlüsselt und an Y zurückgeschickt. Nach erfolgreicher Entschlüsselung mit dem öffentlichen Schlüssel von X wird die Phrase mit dem Original verglichen. Bei positivem Vergleich kann Y sicher sein, mit X zu kommunizieren.

Um die wechselseitige Authentifizierung zu komplettieren, muss die vollständige Prozedur nun noch in die entgegengesetzte Richtung durchgeführt werden. Auch X kann sich dann seines Kommunikationspartners sicher sein und der Prozess ist abgeschlossen.

Das ursprüngliche SSL Protokoll wird für die Nutzung im Globus Toolkit um die Verwendung von Proxy Zertifikaten erweitert. Es muss hier, wie in Kapitel 2.2.3 noch genauer erklärt wird, darauf Rücksicht genommen werden, dass PCs nicht nur von CAs signiert werden können, sondern eben auch von EECs und anderen Proxy Zertifikaten.

2.2.3 Proxy Credentials

Proxy Credentials werden im Globus Toolkit verwendet, um dynamische Delegation und einen sicheren Single Sign-On zu ermöglichen. Der SSO erleichtert dem Nutzer das Arbeiten erheblich. Er wird nur einmal beim Start einer jeden Grid-Sitzung nach dem Passwort des privaten Schlüssels seines EECs gefragt, und von diesem Zeitpunkt an sind sämtliche Transaktionen ohne weitere Passworteingaben möglich.

Es gibt verschiedene Anforderungen an eine Grid Middleware, die die dynamische Delegation von Nutzerrechten erforderlich machen. So werden beim Arbeiten in Grids häufig Meta-Scheduler verwendet, die Jobs, abhängig von Jobbeschreibung und aktueller Verfügbarkeit von Ressourcen, an die endgültig ausführenden Ressourcen schicken.

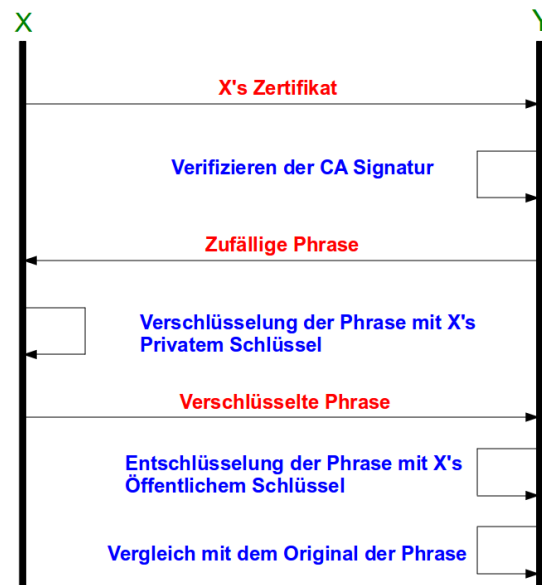


Abbildung 2.4: Wechselseitige Authentifizierung

Dies müssen sie im Auftrag des ursprünglichen Issuers eines Jobs machen können. Sie benötigen dafür einen Teil seiner Rechte.

Außerdem sollen Ressourcen grundsätzlich die Möglichkeit haben, im Namen anderer Entitäten Aktionen durchzuführen. Wird ein Job an die GRAM-Schnittstelle einer Ressource geschickt, so soll die Ressource beispielsweise die Möglichkeit haben, im Namen des Auftraggebers Daten von anderen Ressourcen nachzuladen.

Durch die Verwendung von Proxy Zertifikaten wird der private Schlüssel des End Entity Certificates geschützt. Wie oben erwähnt, wird dieser nur einmal zur Erstellung des Sitzungsproxys verwendet und ansonsten nicht benötigt. Das Passwort des Schlüssels wird zu keiner Zeit, weder verschlüsselt, noch unverschlüsselt, über das Netzwerk geschickt. Für weitere Transaktionen werden Proxy Zertifikate vom Sitzungsproxy abgeleitet.

Aufbau

Proxy Zertifikate repräsentieren den Stellvertreter einer Entität, beziehungsweise des EECs der Entität. In PKI-basierten Systemen kann mit ihrer Hilfe eingeschränktes Proxying und Delegation realisiert werden. Proxy Zertifikate wurden in RFC 3820[TWE⁺] standardisiert und sind eine Erweiterung zu X.509 Zertifikaten. Sie stellen eine eigene Identität dar und können sowohl von EECs, als auch von anderen Proxy Zertifikaten signiert werden. Dies ist einer der entscheidenden Unterschiede zu X.509 EECs, die nur durch eine vertrauenswürdige Certificate Authority signiert werden dürfen. Ein weiterer wichtiger Unterschied ist die stark verkürzte Gültigkeitsdauer. Bei PCs beträgt sie im Standardfall lediglich zwölf Stunden.

In Listing 2.2 ist der Aufbau eines Proxy Zertifikats zu sehen.

Angegeben sind unter anderem der Distinguished Name (DN) des Ausstellers des Zertifikats (Issuer), die Gültigkeitsdauer (Validity), der DN des Subjekts (Subject), also der Identität des Zertifikats, sowie der öffentliche Schlüssel und die Erweiterungen.

```

1 Certificate:
2   Data:
3     Version: 3 (0x2)
4     Serial Number: 118308523 (0x70d3eab)
5     Signature Algorithm: sha1WithRSAEncryption
6     Issuer: O=Grid, OU=GlobusTest, OU=simpleCA-
7             debian6.fritz.box, OU=fritz.box, CN=Grid User HP7
8     Validity
9       Not Before: Dec  1 11:10:13 2011 GMT
10      Not After  : Dec  1 23:15:13 2011 GMT
11     Subject: O=Grid, OU=GlobusTest, OU=simpleCA-
12             debian6.fritz.box, OU=fritz.box, CN=Grid User HP7,
13             CN=118308523
14     Subject Public Key Info:
15       Public Key Algorithm: rsaEncryption
16       Public-Key: (512 bit)
17       Modulus:
18         00:b5:60:a0:e5...
19       Exponent: 65537 (0x10001)
19     X509v3 extensions:
20       Proxy Certificate Information: critical
21       Path Length Constraint: infinite
22       Policy Language: Inherit all
23
24     Signature Algorithm: sha1WithRSAEncryption
25       8a:de:a2:24:3c...

```

Listing 2.2: Textdarstellung X509v3 Zertifikat

Ein PC unterscheidet sich von einem 'normalen' X509v3 Zertifikat im Wesentlichen also nur durch die Bildung des Subject-DN und einer neuen Erweiterung. Der DN des Zertifikats wird, wie im Beispiel zu sehen, vom DN des Issuers abgeleitet. Es wird einfach eine zusätzliche Common Name (CN) Komponente in Form einer Zufallszahl angehängt. Im Beispiel aus Listing 2.2 ist es CN=118308523.

Bei den Erweiterungen findet man die Proxy Certificate Information (PCI) Extension, die jedes X509v3 Proxy Zertifikat haben muss. Im Beispiel handelt es sich um ein Zertifikat, das alle Rechte seines Ausstellers erbt und keine Begrenzung für den Proxy Pfad enthält. Die Bedeutung dieser Erweiterung wird in einem eigenen Abschnitt genauer beschrieben.

Als Proxy Credential versteht man das eben ausführlich erläuterte Zertifikat, den dazugehörigen privaten Schlüssel und die gesamte Zertifikatskette. Diese beinhaltet alle beteiligten Proxy Zertifikate und das EEC des ursprünglichen Ausstellers. Das CA Zertifikat ist ausgenommen, da es direkt bei der CA erhältlich ist. Die komplette Kette ist nötig, damit die Gültigkeit der Zertifikate bis hin zur CA überprüft werden kann. Dies erfordert eine Erweiterung des X.509 Validierungsprozesses. Bisher musste nur die Signatur einer CA überprüft

werden, da kein anderer Aussteller in Frage kam und dadurch auch keine Ketten möglich waren.

Proxy Certificate Information Extension

Die *Proxy Certificate Information (ProxyCertInfo)* Erweiterung ist eine kritische Erweiterung, sie muss also von allen beteiligten Entitäten interpretiert werden können. Sie kennzeichnet Proxy Zertifikate als solche und hat ein Pflichtfeld und zwei optionale Felder. Das *pCPathLenConstraint* Feld ist optional und kann verwendet werden, um die maximale Tiefe des Proxy-Pfades festzulegen. Ist sein Wert 0, so darf damit überhaupt kein PC signiert werden, ist das Feld nicht vorhanden, so darf der Pfad beliebig lang werden.

Das *proxyPolicy* Feld legt Regeln für das Zertifikat im Bezug auf die Autorisierung fest und besteht aus zwei weiteren Feldern. Das *policyLanguage* Feld indiziert die Sprache, die für die Policies verwendet werden soll. Es ist ein Pflichtfeld und kann zwei Spezialwerte annehmen. Der eine (*id-ppl-inheritAll*) dient dazu, den Proxy als unbeschränkt zu kennzeichnen. Dadurch werden alle Rechte des signierenden Zertifikats an den erstellten Proxy delegiert.

Der zweite Wert (*id-ppl-independent*) markiert das Zertifikat als unabhängig, was dazu führt, dass keinerlei Rechte übertragen werden. Das Zertifikat stellt also eine neue Identität dar, die über andere Wege, beispielsweise durch Attribut Zertifikate (AC), explizit mit Rechten ausgestattet werden kann. Hat das Feld einen dieser beiden Werte, so ist das *policy* Feld nicht nötig.

Jeder andere Wert indiziert, dass es sich um ein Proxy Zertifikat mit Beschränkungen handelt. Die zugehörigen Regeln können dann entweder implizit im Wert des *policyLanguage* Feldes enthalten sein, oder Policy Informationen werden im *policy* Feld angegeben. Wird dieses Feld benutzt, so wird im *policyLanguage* Feld die dafür verwendete Sprache, beispielsweise in Form eines Object Identifiers (OID), angegeben.

Mit Hilfe dieser Erweiterung ist es also möglich, die durch Proxy Zertifikate delegierten Rechte zu beschränken. Denkbar ist hier, Anfragen nur auf bestimmte Ressourcen zu begrenzen, oder nur bestimmte Aktionen auf festgelegten Servern zu ermöglichen.

Wie bereits erwähnt, werden in RFC 3820 keinerlei Vorgaben über die zu verwendende Policy-Sprache gemacht. Diese kann also, abhängig vom Anwendungsfall, frei gewählt werden.

2.3 XACML

eXtensible Access Control Markup Language (XACML) bezeichnet einerseits eine XML-basierte Policy-Sprache, die der Beschreibung genereller Anforderungen der Zugriffskontrolle dient und die Definition von Policies ermöglicht. Andererseits handelt es sich um eine, ebenfalls XML-basierte, Request/Response-Sprache für Zugriffskontrollentscheidungen mit einem dazugehörigen Zugriffskontroll-Framework.

2.3.1 Framework

In Abbildung 2.5 sind die Komponenten des Zugriffskontrollsystems zu erkennen.

Im Policy Decision Point (PDP) findet die eigentliche Entscheidung über den Zugriff auf eine Ressource statt. Der PDP hat aber keinerlei Schutzfunktion für die Ressource, sondern bekommt Anfragen und fällt Entscheidungen bezüglich selbiger.

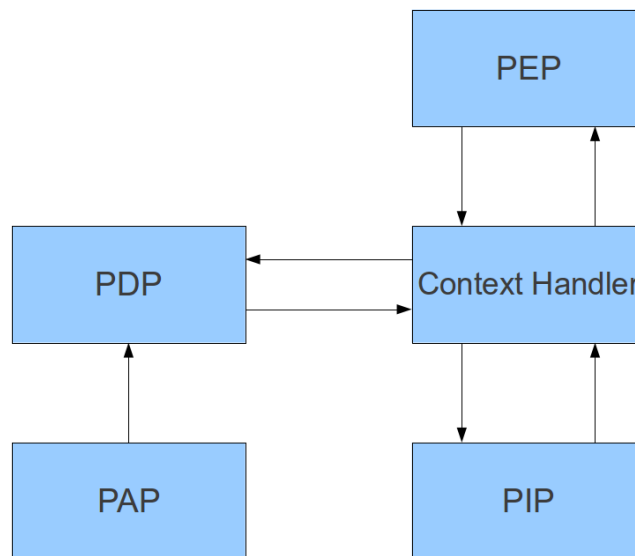


Abbildung 2.5: XACML Framework (in vereinfachter Form)

Der Policy Enforcement Point (PEP) befindet sich als Schutz zwischen dem Subjekt der Anfrage und der angefragten Ressource. Hier wird die Entscheidung nur umgesetzt, warum sie zustande kam, ist für den PEP nicht von Interesse. Im Policy Administration Point (PAP) werden die Regeln festgelegt, nach denen Zugriffe erlaubt, oder Anfragen abgelehnt werden. Es kann sich hier beispielsweise um eine Datenbank handeln, die die festgelegten Regeln enthält. Der PDP bekommt die für die Entscheidungsfindung nötigen Regeln dann vom PAP. Vom Policy Information Point (PIP) bekommt der PEP weitere nötige Informationen. Eine Anfrage enthält in der Regel nur Subjekt, Aktion und Objekt. Policies verwenden aber oft Rollen. Diese fordert der PDP dann beim PIP an. Der Context Handler tritt als Vermittler zwischen den anderen Komponenten auf. Er steht im Mittelpunkt des Systems und verteilt die benötigten Nachrichten.

Eine Anfrage wird also an den PEP gestellt, dann vom Context Handler in das passende XACML-Format umgewandelt, Informationen über Attribute vom PIP geholt und die Anfrage an den PDP weitergeleitet. Dieser trifft, unter Zuhilfenahme weiterer Informationen von PIP und PAP eine Entscheidung und gibt die Antwort über den Context Handler an den PEP zurück. Abhängig vom Ergebnis gewährt oder verweigert dieser den Zugriff.

2.3.2 Datenstrukturen

In XACML gibt es drei wichtige Datenstrukturen. Anfragen und Antworten der Request/Response-Sprache, sowie die Policybeschreibungen der Policy-Sprache.

Anfragen

Eine Anfrage besteht in der Regel aus Attributen in den vier Abschnitten Subject, Action, Resource und Environment. Ein einfaches Beispiel ist in Listing 2.1 zu sehen. Schorsch Mustermann (Subject) will dabei schreibend (Action) auf das Dokument test.txt (Resource) zugreifen. Die ersten drei Abschnitte behandeln also das WER, WAS und WOMIT der Anfrage. Alles, was diesen Abschnitten nicht zugeordnet werden kann, kommt in den Bereich Environment. Es sind auch mehrere Subjekte möglich, die dann jeweils einen Subject-Abschnitt mit zugehöriger SubjectCategory bekommen.

```

1  <Request>
2  <Subject>
3    <Attribute AttributeId="subject-id">
4      <AttributeValue>Schorsch Mustermann</AttributeValue>
5    </Attribute>
6  </Subject>
7  <Action>
8    <Attribute AttributeId="action-id">
9      <AttributeValue>write</AttributeValue>
10   </Attribute>
11 </Action>
12 <Resource>
13   <Attribute AttributeId="resource-id">
14     <AttributeValue>file:///home/griduser/test.txt</
15     AttributeValue>
16   </Attribute>
17 </Resource>
18 </Request>

```

Listing 2.3: Einfache XACML Anfrage

Antworten

Eine einfache XACML-Antwort ist in Listing 2.2 zu sehen. Sie besteht aus einer Entscheidung, in diesem Fall *Permit*, und einem Statuscode, im Beispiel *status:ok*. Eine Response kann außerdem noch Obligationen enthalten. Diese ermöglichen dem PDP, Anforderungen an den PEP zu stellen, die im Falle einer bestimmten Entscheidung erfüllt werden müssen. Hier können beispielsweise Formalitäten, wie das Logging, umgesetzt werden, die schwer in Policies ausgedrückt werden können. Der PDP kann dem PEP beispielsweise auftragen, dass im Falle eines *Permit* ein Log-Eintrag erstellt werden muss. Dadurch kann nachvollzogen werden, wer Zugriff auf welche Ressourcen bekommen hat.

```

1  <Response>
2  <Result>
3    <Decision>
4      Permit
5    </Decision>

```

```

6     <Status>
7         <StatusCode Value="urn:oasis:names:tc:xacml:1.0:status:ok
           "/>
8     </Status>
9 </Result>
10 </Response>

```

Listing 2.4: Einfache XACML Anfrage

Policys

Die dritte und wichtigste Datenstruktur im Bereich XACML ist die Policy selbst. Man unterscheidet zwischen zwei Ausprägungen. Der Policy, die einzelne Regeln (Rules) kombiniert, und dem Policy Set, in dem Policies zu größeren Einheiten kombiniert werden. In beiden Fällen gibt es den Anwendungsbereich (Target), in dem beschrieben wird, für welche Anfragen die Policy zur Auswertung genutzt werden kann. Sowohl Policys, als auch PolicySets können also bezüglich Anfragen ausgewertet werden und liefern als Ergebnis eine Entscheidung, die die Werte *Permit*, *Deny*, *NotApplicable*, oder *Indeterminate* haben kann. Die beiden ersteren erklären sich von selbst, *NotApplicable* wird zurückgegeben, wenn die Policy nicht zur aktuellen Anfrage passt, *Indeterminate*, wenn es während der Auswertung zu einem Fehler kommt. Gemeinsam ist beiden Policy-Ausprägungen auch ein Kombinationsalgorithmus, der die jeweiligen Kindelemente, also Rules oder Policys, als Eingabe erhält, diese auswertet und die Ergebnisse kombiniert. In Listing 2.3 ist das Gerüst einer Policy zu sehen.

```

1 <Policy RuleCombiningAlgId="first-applicable">
2   <Target>
3     ...
4   </Target>
5   <Rule RuleId="Admins" Effect="Permit">
6     <Description>Administrator darf alles.</Description>
7     ...
8   </Rule>
9     ...
10  <Rule RuleId="fallback" Effect="Deny"/>
11 </Policy>

```

Listing 2.5: XACML-Policy-Fragment

Es handelt sich hier um die Kombination zweier Rules, die zu einer Policy zusammengefasst sind und einen bestimmten Anwendungsbereich haben (Target). Kann die erste Regel angewendet werden, so hat sie als Ergebnis ein *Permit*, wird die zweite Regel benutzt, so gibt sie *Deny* zurück. Als Kombinationsalgorithmus für die Regeln wird *first-applicable* angegeben. Das Ergebnis der ersten Regel, die anwendbar ist, ist somit auch das Ergebnis der Policy. Die Elemente *Rule*, *Policy* und *PolicySet* haben einen Anwendungsbereich (Target) als Kindelement. Hier wird festgelegt, auf welche Anfragen das jeweilige Element angewendet werden kann. Der Anwendungsbereich besteht aus den Abschnitten *Subjects*, *Actions*, *Resources* und *Environments*. Ein Anwendungsbereich passt zu einer Anfrage, wenn alle vier Abschnitte passen.

Jeder Abschnitt kann wiederum Unterabschnitte haben. Diese heißen *Subject*, *Action*, *Resource* und *Environment*. Ein Abschnitt passt zu einer Anfrage, wenn mindestens einer der Unterabschnitte passt. Ein Unterabschnitt besteht aus **Match*-Elementen. Beispiele sind hier *SubjectMatch*, *ResourceMatch* und *ActionMatch*. Ein Unterabschnitt passt zu einer Anfrage, wenn alle seine **Match*-Elemente passen.

Mit **Match*-Elementen wird geprüft, ob ein Attribut aus einer Anfrage zu einem Attribut aus einer Policy, oder Regel, passt. Enthalten sind in solchen Elementen eine Funktions-ID, ein Attributwert und ein Verweis auf ein Attribut aus der Anfrage. Funktion-IDs können auf zahlreiche vordefinierte, oder auch durch den Benutzer festgelegte Funktionen verweisen. Diese Funktionen bekommen als erstes Argument das Attribut des **Match*-Elements und als zweites nacheinander alle Attributwerte der Anfrage. Die Anfrage passt also, wenn für eines der Argumentpaare *true* zurückgegeben wird, also wenn das Argument aus **Match* zu einem der Attributwerte aus der Anfrage passt.

XACML besitzt über 200 solcher vordefinierter Funktionen. Darunter sind Funktionen für Mengen, Vergleichsfunktionen, arithmetische Funktionen und boolsche Funktionen. Die Funktionen werden dazu verwendet, komplexe Ausdrücke zu bilden. Dies wird durch die Anwendung der Funktionen auf Listen von Argumenten erreicht, wobei die Argumente wieder Ausdrücke sein müssen.

Es gibt verschiedene Arten von Ausdrücken. Ein *AttributeValue* ist ein Ausdruck eines bestimmten elementaren Typs, mit einem Wert als Literal. In XACML 2.0 werden 16 Datentypen vordefiniert. Darunter ist, neben üblichen Datentypen wie *integer*, *boolean*, *string* und *date*, beispielsweise auch ein Datentyp zur Beschreibung von Emailadressen (*rfc822Name*). Als weitere Ausdrucksart gibt es die **AttributeDesignator*, mit deren Hilfe die Werte eines benannten Attributes aus einer Anfrage ermittelt werden können. Beispiele für die beiden genannten Arten sind in Listing 2.4 abgebildet. Ein **AttributeSelector* dient ebenfalls dazu, Daten aus Anfragen zu gewinnen, nur werden diese nicht über Namen, sondern über XPath-Ausdrücke identifiziert. Ausdrücke der Art *Function* werden ausschließlich als Argumente für Funktionen höherer Ordnung verwendet. *VariableReference* gibt den Wert einer vorher definierten Variable wieder. Dies kann genutzt werden, um häufig benutzte Ausdrücke benennen und wiederverwenden zu können. Ausdrücke der Art *Apply* sind Ergebnisse von Funktionsaufrufen. Damit können beliebig komplex geschachtelte Ausdrücke gebildet werden.

```

1 <AttributeValue DataType="string">Hallihallo</AttributeValue >
2 <SubjectAttributeDesignator AttributeId="subject-id" DataType=
   "integer">
```

Listing 2.6: XACML-Policy-Fragment

Regeln sind, wie oben erwähnt, Bestandteile von Policies, die ausgewertet werden müssen, um Entscheidungen zu treffen. Sie bestehen aus einem Anwendungsbereich (Target), ihrer Auswirkung (Effect) und einer optionalen Bedingung (Condition). Wird in einer Regel kein Anwendungsbereich definiert, so gilt der Bereich der Policy, die diese Regel enthält. Die *Condition* ist ein Ausdruck in Form einer boolschen Funktion. Der Effekt einer Regel ist entweder *Permit*, oder *Deny*. Bei der Auswertung können sich die Werte *Permit*, *Deny*, *NotApplicable* und *Indeterminate* ergeben. Falls das Target nicht zur Anfrage passt, wird *NotApplicable* zurückgegeben. Sonst wird die *Condition* überprüft und für den Fall, dass sie *true* ergibt, lautet das Ergebnis wie in *Effect* angegeben. Ist es *false*, so wird *NotApplicable* ausgegeben.

Bei irgendwelchen Fehlern in einem der Schritte lautet das Ergebnis *Indeterminate*.

2.4 Resource Specification Language

Die Globus Resource Specification Language (RSL)⁴ ist eine verbreitete Sprache zum Austausch von Ressourcen-Beschreibungen. Hier soll eine kurze Einführung in die Syntax der Sprache gegeben und ein kleines Beispiel gezeigt werden. RSL bietet mit ihrer Syntax die Möglichkeiten, um komplexe Beschreibungen zu erstellen. Die verschiedenen Resource Management Komponenten führen dabei spezielle ;Attribut, Wert; Paare in die allgemeine Struktur ein. Jedes dieser Attribute dient als Parameter, um das Verhalten einer oder mehrerer Komponenten im Resource Management System zu kontrollieren.

Das Herzstück der RSL Syntax ist die Relation. Sie verbindet einen Attributnamen mit einem Wert. Außerdem gibt es zwei generative Strukturen in der RSL, die es ermöglichen, aus Relationen komplexere Ressourcen-Beschreibungen zu bilden. Es handelt sich dabei um *compound requests* und *value sequences*. Außerdem beinhaltet die Resource Specification Language die Option String *substitution variables* sowohl einzuführen, als auch zu dereferenzieren.

Die einfachste Form eines *compound request*, die auch von allen Resource Management Komponenten verwendet wird, ist ein *conjunction request*. Dabei handelt es sich um die Verknüpfung einfacher Relationen, oder *compound requests*. Damit können beispielsweise mehrere Relationen, wie *executable name*, *node count*, *executable arguments* und *output files* einer üblichen GRAM Job-Anfrage kombiniert werden. Zusätzlich sind auch noch *multi requests* möglich. Damit können mehrere parallele Ressourcen beschrieben werden, die eine Ressourcen-Beschreibung ausmachen. *multi requests* führen neue Gültigkeitsbereiche für Variablen ein. Variablen aus einem Abschnitt sind in anderen Abschnitten des *requests* nicht sichtbar. Genauere Informationen über *multi requests* sind in [All] zu finden.

Die einfachste Form eines Werts in der RSL Syntax ist das String Literal. Explizit angegebene Literale können dabei sämtliche Zeichen enthalten. Ein Wert kann außerdem eine Referenz auf eine Variable sein. Die Referenz wird dabei im Endeffekt durch den Wert ersetzt, der für die Variable festgelegt wurde. Die Verkettung von String-Werten wird von RSL ebenfalls unterstützt. Auch Wertsequenzen sind möglich und drücken geordnete Folgen von Werten aus. Damit können beispielsweise die Argumentlisten einer Anwendung dargestellt werden. Einige der häufig verwendeten RSL-Attribute sind *arguments*, *count*, *environment*, *executable*, *file_stage_in*, *file_stage_out*, *job_type* und *max_cpu_time*. Weitere Attribute und ihre genaue Bedeutung, sowie die komplette Spezifikation der RSL v.1.1, sind in [All] zu finden. Listing 2.7 zeigt eine typische GRAM-Job-Beschreibung.

```

1  (* this is a comment *)
2  & (executable = a.out (* <-- that is an unquoted literal *))
3  (directory = /home/nobody )
4  (arguments = arg1 "arg_2")
5  (count = 1)

```

Listing 2.7: RSL Ressourcen-Beschreibung

⁴<http://globus.org/toolkit/docs/5.0/5.0.0/execution/gram5/pi/gram5PublicInterfacesGuide.pdf>, Chapter 2

Angegeben sind hier das Executable, das Verzeichnis, in dem die Ausführung stattfinden soll, sowie zwei zu verwendende Kommandozeilenparameter. Zusätzlich wird noch angegeben, wie oft die angegebene Datei ausgeführt werden soll.

3 Analyse

In diesem Abschnitt sollen die Anforderungen an den Ansatz zur feingranularen Delegation von Nutzerrechten erarbeitet werden. Als Grundlage dafür dienen die in einem Fortgeschrittenenpraktikum gesammelten Informationen, sowie ein typischer Ablauf eines GRAM Job-Submit, wie er im aktuellen Ansatz vorkommt.

3.1 Authentifizierung und Autorisierung - Ist-Zustand

3.1.1 Erkenntnisse aus dem Fortgeschrittenenpraktikum

Der Titel der in diesem Zusammenhang erstellten Praktikumsarbeit lautet 'Beurteilung der Sicherheit bei der Verwendung von Proxy Zertifikaten im Globus Toolkit' [Sch10]. Darin wird hauptsächlich Version 4 des Toolkit behandelt. Diese Version unterscheidet sich aber in der Verwendung von Proxy Zertifikaten nicht wesentlich von Version 5, die als Grundlage für die aktuelle Arbeit zur Rechtebegrenzung dient.

In angesprochenem Praktikum wurden potentielle Schwachstellen bei der Verwendung von Proxy Zertifikaten in GT gesammelt und überprüft. Bei den Recherchen dazu wurden unter anderem auch Personen aus dem Industrieumfeld befragt, um die dort sehr hoch angesetzten Sicherheitsstandards mit zu berücksichtigen. Als eine der vermeintlichen Schwachstellen wurde das Fehlen adäquater Begrenzungsmöglichkeiten delegierter Rechte bei der Verwendung von Proxy Zertifikaten ausgemacht.

Wie in [Sch10] beschrieben, bieten das Globus Toolkit, beziehungsweise das Proxy-Konzept an sich, sehr wohl Beschränkungsmöglichkeiten an. Es kann sowohl die Gültigkeitsdauer eines Zertifikats, als auch die Länge eines möglichen Proxy-Pfads frei bestimmt werden. Außerdem wird im Globus Toolkit das Konzept eines `limited` Proxy angeboten. Derartigen Zertifikaten fehlt das Recht, neue Prozesse zu starten. Das abschicken neuer GRAM Jobs wird somit verhindert, sie können jedoch problemlos zur Authentifizierung gegenüber einer Ressource, oder auch zum Datentransfer mittels GridFTP, verwendet werden. Außerdem existieren noch `independent` Proxies, denen per se überhaupt keine Rechte des Ausstellers übertragen werden.

All diese Möglichkeiten bieten jedoch nur grobe Abstufungen der delegierten Rechte. Soll es einem Job beispielsweise möglich sein, weitere Jobs zu starten, so ist die einzige Option dies zu gewährleisten die Übertragung eines `full` Proxy. Diesem werden also sämtliche Rechte des Issuers übertragen, obwohl er nur einen Teil davon wirklich braucht. Der Zugang zu allen Ressourcen, auf denen der Issuer autorisiert ist, ist möglich, obwohl unter Umständen nur ein Bruchteil davon für die aktuelle Aufgabe benötigt wird.

Diese Notwendigkeit einer vollständigen Delegation wurde in den Interviews häufig als ein großes Problem genannt, das die Nutzung von GT für das Arbeiten mit sensitiven Daten häufig ausschließt. Tatsächlich fehlt in GT die Option, feingranulare, auf bestimmte Tätigkeiten zugeschnittene Begrenzungen zu erreichen.

3.1.2 Standardszenario

In folgendem Abschnitt soll anhand eines repräsentativen Beispiels gezeigt werden, wie Authentifizierung und Autorisierung im Moment im Globus Toolkit gelöst sind. In Abbildung 3.1 ist eine sehr vereinfachte Darstellung der ablaufenden Mechanismen in GT5, beim Submit eines GRAM-Jobs durch einen Nutzer, dargestellt. Zu sehen ist der Standardfall, der ohne Anpassungen an ein spezielles Grid vorgefunden wird.

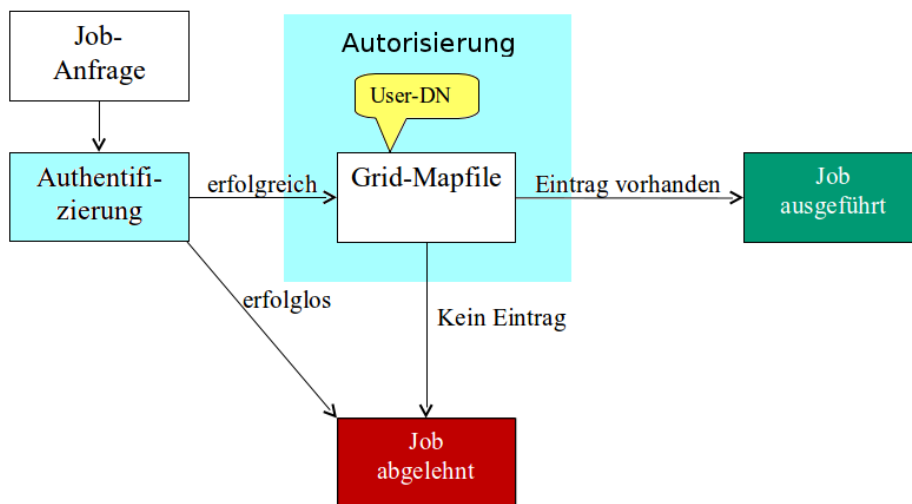


Abbildung 3.1: Sicherheitskonzept GT (stark vereinfachte Form)

Der Ablauf beginnt mit einer Job-Anfrage durch einen Nutzer. Es kann sich hierbei sowohl um eine Person, als auch um eine andere Ressource, die im Auftrag einer Person agiert, handeln. Vor jeder Transaktion kommt es in GT-basierten Grids zu einer wechselseitigen Authentifizierung der beteiligten Parteien. Diese wird, wie in Kapitel 2.2.2 beschrieben, unter Verwendung der Zertifikate beider Seiten durchgeführt. In der Regel handelt es sich beim Zertifikat der anfragenden Partei um ein Proxy Zertifikat, beim Zertifikat des Globus Gatekeepers auf der Zielseite um ein End Entity Certificate. Verläuft die Authentifizierung erfolglos, so wird der angefragte Job abgelehnt, verläuft sie erfolgreich, so ist der nächste Schritt die Autorisierung.

Das Toolkit bietet Entwicklern die Möglichkeit, eigene Autorisierungsmechanismen zu implementieren, im Standardfall wird jedoch ein grobgranularer Ansatz unter Verwendung eines Grid-Mapfiles verwendet. Die Einträge in dieser Datei bestehen aus einem Distinguished Name und einer ihm zugeordneten Systemkennung. Ein Beispiel dafür ist in Listing 5.1 zu sehen. Dem Nutzer mit der Common Name (CN) Komponente /CN=Grid User wird hier die Systemkennung `griduserdebian6` zugeordnet.

```

1  "/O=Grid/OU=GlobusTest/OU=simpleCA-debian6.fritz.box/OU=fritz.
   box/CN=Grid_□User" griduserdebian6
2  "/O=Grid/OU=GlobusTest/OU=simpleCA-debian6.fritz.box/OU=fritz.
   box/CN=Grid_□User_□HP7" griduser
  
```

Listing 3.1: Grid-Mapfile

Beim Autorisierungsvorgang wird nun der Distinguished Name aus dem Zertifikat des Users genommen und nach dem entsprechenden Eintrag im Grid-Mapfile gesucht. Ist ein Eintrag vorhanden, so ist die Autorisierung erfolgreich und es findet ein Mapping auf die angegebene Systemkennung statt, mit der der angefragte Job dann ausgeführt wird. Wird kein passender Eintrag gefunden, so hat der Nutzer nicht das Recht, auf der Ressource zu arbeiten und die Anfrage wird abgelehnt.

Es wird also nach erfolgreicher Authentifizierung überprüft, ob der anfragende Nutzer (kann auch eine Ressource sein) autorisiert ist, auf der angefragten Ressource zu arbeiten. Die momentanen Mechanismen sehen keine Möglichkeit vor, die es dem Nutzer erlaubt, feingranulare Einschränkungen bezüglich der Verwendung des vorgezeigten Zertifikats zu tätigen, die dann von der Ressource überprüft und gegebenenfalls umgesetzt werden könnten.

3.2 Anforderungen

In diesem Abschnitt werden Ziele, Systemgrenzen und daraus abgeleitet funktionale und nicht-funktionale Anforderungen des erweiterten Delegationsansatzes erarbeitet. Grundlage für diesen Prozess ist die Definition einer Anforderung in [Rup07]. Diese wird hier als „eine Aussage über eine Eigenschaft oder Leistung eines Produktes, eines Prozesses oder der am Prozess beteiligten Personen“ beschrieben. Eine solche Definition beinhaltet beide Kategorien von Anforderungen. Funktionale Anforderungen beschreiben laut [Rup07] „Aktionen, die von einem System selbstständig ausgeführt werden sollen, Interaktionen des Systems mit menschlichen Nutzern oder Systemen (Eingaben, Ausgaben) und Anforderungen zu allgemeinen, funktionalen Vereinbarungen und Einschränkungen.“ Nicht-funktionale Anforderungen werden dort als „alle Anforderungen, die nicht funktional sind“ zusammengefasst.

3.2.1 Ziele und Systemgrenzen

Ziel der Entwicklung ist es, den in GT5 bestehenden Delegationsansatz zu erweitern und eine feingranulare, regelbasierte Beschränkung der bei der Nutzung von Proxy Zertifikaten übertragenen Rechte zu ermöglichen. Delegierte Rechte sollen dabei auf spezielle Aufgaben zugeschnitten werden können, damit die Delegation der vollständigen Rechte, unter Verwendung von Proxy Zertifikaten, möglichst verhindert wird. Die Möglichkeiten des bestehenden Ansatzes sollen außerdem in vollem Umfang erhalten bleiben.

Der Kontext des zu entwickelnden Systems beinhaltet die Grid Security Infrastructure(GSI), das Globus Toolkit 5, das diese implementiert, sowie die enthaltenen Authentifizierungs- und Autorisierungsmechanismen und die Ausführung von Jobs. Hier sind vor allem Grid Resource Allocation and Management (GRAM), sowie das Konzept der Proxy Zertifikate und das damit verbundene Delegationskonzept, explizit zu nennen. All diese Bereiche werden vom zu entwickelnden System benutzt, beziehungsweise manipuliert, sind aber nicht Teil davon.

3.2.2 Informelle Anforderungsbeschreibungen

Dieses Kapitel zeigt die informelle Beschreibung der Anforderungen an die zu entwickelnde Erweiterung des Delegationsansatzes des Globus Toolkit 5. Diese Anforderungen basieren hauptsächlich auf den Erkenntnissen des in Kapitel 3.1.1 behandelten Fortgeschrittenenpraktikums [Sch10], sowie den grundsätzlichen Eigenschaften eines Grids und einer Literaturrecherche zum Thema dynamische Delegation von Nutzerrechten in Grid-Umgebungen.

Außerdem wurden natürlich die in Kapitel 3.2.1 angegebenen Ziele einbezogen. Die Anforderungen sind von A1 bis A7 durchnummeriert. A1 bis A6 beschreiben dabei funktionale und nicht-funktionale Anforderungen an das zu entwickelnde System. A7 betrifft den Erhalt von Mechanismen des alten, zu erweiternden Systems. Die Anforderungen werden gemäß des Anforderungs-Templates aus [Rup07] dargestellt. Zuerst wird die Begründung für eine Anforderung gegeben, dann werden daraus konkrete Anforderungsdefinitionen hergeleitet und am Schluss Angaben zur Messbarkeit der Erfüllung einer Anforderung gemacht und damit festgelegt, wann ein System diese erfüllt.

A1 - Erstellung/Definition von Beschränkungen

Der Nutzer sollte für die Erstellung, beziehungsweise Definition, von gewünschten Beschränkungen von der Middleware unterstützt werden. Das Hauptaugenmerk liegt bei der Nutzung eines Grids auf den zu erledigenden Berechnungen, Simulationen und so weiter. Aus Sicht der Effizienz sollten organisatorische und sicherheitstechnische Aufgaben deshalb so schnell wie möglich erledigt werden können, ohne dabei an Qualität einzubüßen. Sowohl die Einarbeitungszeit in den gegebenen Delegationsmechanismus, als auch der Zeitaufwand beim tatsächliche Arbeiten mit ihm, sollte daher so gering wie möglich gehalten werden.

Können die Beschränkungen computergestützt erstellt werden, so ist es ebenfalls möglich, syntaktische Fehler des Nutzers abzufangen und ihn gegebenenfalls auf sie hinzuweisen. Außerdem kann eine einheitliche Form, in einer noch zu definierenden Syntax, gewährleistet werden.

A1.1 Das System soll den Globus User bei der Erstellung/Definition von feingranularen Beschränkungen unterstützen. Diese Anforderung ist erfüllt, wenn die Definition von Einschränkungen computergestützt durchgeführt werden kann.

A2 - Beschränkung der Rechte auf begrenzte Aufgaben

Der zu entwickelnde Delegationsansatz soll eine feingranulare Delegation von Nutzerrechten, mit Hilfe von Proxy Zertifikaten, ermöglichen. Die bisherigen Mechanismen lassen lediglich eine Unterscheidung zwischen full, limited und independent Proxy zu. Independent Proxies werden in Globus-Umgebungen kaum eingesetzt, die anderen beiden Optionen delegieren alle Rechte, beziehungsweise alle, außer diejenigen zum Starten neuer Prozesse. Der bisherige Ansatz ist also sehr unflexibel und lässt sich nur sehr geringfügig an aktuelle Bedürfnisse anpassen. Es ist nicht möglich, delegierte Rechte auf eine bevorstehende Aufgabe zuzuschneiden.

Wird beispielsweise ein GRAM-Job an eine entfernte Ressource abgeschickt, so wird standardmäßig ein limited Proxy für seine Durchführung auf dieser Ressource erstellt. Benötigt der Job allerdings mehr Rechte, als er durch ein limited PC erhält, so muss ein full Proxy delegiert werden. Diesem werden dann alle Rechte des Nutzers übertragen und es sind keine weiteren Einschränkungen möglich.

Im Rahmen des in dieser Arbeit schon häufiger erwähnten Fortgeschrittenenpraktikums zur Verwendung von Proxy Zertifikaten in Globus Toolkit 4 [Sch10] wurden, unter Einbeziehung von Interviews im industrienahen Umfeld, einige Defizite der vorhandenen Delegationsmöglichkeiten erarbeitet. Auch hier wird vor allem die fehlende Möglichkeit, delegierte

Rechte auf bestimmte Workflows oder GRAM-Jobs abzustimmen, genannt.

Wie in Kapitel 2.2.3 erläutert, kann die PCI Erweiterung von Proxy Zertifikaten zur Begrenzung delegierter Rechte mittels Policies genutzt werden. Es muss also eine Policy-Sprache entwickelt werden, die es dem Nutzer ermöglicht, die Rechte flexibel und feingranular zu begrenzen. Dafür sind Einschränkungsmöglichkeiten in verschiedenen Kategorien nötig:

Die möglichen **Ziele** von Anfragen müssen begrenzt werden können. Ein großes Risiko bei der Verwendung von Proxy Zertifikaten mit vollen Rechten, besteht darin, dass dem Besitzer des Proxy Zertifikats damit alle Ressourcen, zu denen der ursprüngliche Issuer grundsätzlich Zugang hat, offen stehen. Häufig ist für die Erledigung einer Aufgabe aber nur eine begrenzte, feststehende Menge von Ressourcen nötig. Hier könnten Sicherheits-, beziehungsweise finanzielle Risiken durch eine Begrenzung möglicher Ziele minimiert werden.

Um eine weitere Eingrenzung der für eine Aufgabe tatsächlich benötigten Rechte zu gewährleisten, muss auch eine Option geschaffen werden, um erlaubte **Aktionen** anzugeben. Dabei sollten sowohl ausführbare Dateien, als auch andere Grid-Aktionen, wie zum Beispiel Stage-in, Stage-out, oder auch GridFTP-Filetransfers bei GRAM-Jobs, berücksichtigt werden. Sind die Informationen über die benötigten Aktionen beim Job-Submit bekannt, so kann auch hier eine starke Verringerung des Sicherheitsrisikos erreicht werden, da ein gestohlenen Zertifikat eben nicht für beliebige Aktionen missbraucht werden kann, sondern nur ganz bestimmte Aufgaben damit möglich sind. Damit wäre dann eine Begrenzung sowohl der Ziele von Anfragen, als auch der dabei angefragten Aktionen zu erreichen und somit eine Anpassung der delegierten Rechte auf spezifische Aufgaben möglich.

A 2.1 Es muss eine Policy-Sprache für die Definition von Begrenzungen angeboten werden. Diese Anforderung ist erfüllt, wenn Syntax und Semantik einer für die feingranulare Delegation von Rechten geeigneten Policy-Sprache definiert wurden.

A2.2 Das System muss dem Nutzer die Option bieten, delegierte Rechte auf bestimmte, erlaubte Ziele von Anfragen zu begrenzen. Diese Anforderung ist erfüllt, wenn die Ziele für Anfragen beim zu entwickelnden Delegationsansatz explizit festgelegt werden können.

A2.3 Das System muss dem Nutzer bei der Delegation von Rechten für Grid-Jobs eine Begrenzung erlaubter ausführbarer Dateien ermöglichen. Die Anforderung ist erfüllt, wenn erlaubte ausführbare Dateien explizit angegeben werden können.

A2.4 Das System muss dem Nutzer eine Möglichkeit zur Begrenzung erlaubter Aktionen bei Grid-Anfragen zur Verfügung stellen. Diese Anforderung ist erfüllt, wenn die Menge erlaubter Grid-Aktionen explizit, auf Aktionen wie Stage-in, Stage-out, job-submit oder job-cancel begrenzt werden kann.

A3 - Integration der Regeln in Proxy Zertifikate

Da das zu entwickelnde System zur Rechtedelegation auf Proxy Zertifikaten, und damit auf der Integration von Regeln in die Proxy Certificate Information Erweiterung, basiert, müssen Optionen geschaffen werden, um eben diese Integration zu ermöglichen. Erst wenn die Regeln in der PCI Erweiterung sind, ist damit eine feingranulare Delegation der Nutzerrechte

möglich. Dies betrifft sowohl die Erstellung von Proxy Zertifikaten mit Hilfe des `grid-proxy-init` Befehls, als auch die Integration von Policies in Job-Proxies, welche bei der Nutzung der GRAM-Schnittstelle auf entfernten Ressourcen erstellt werden.

A3.1 Das System muss dem Nutzer die Integration von Regeln in Proxy Zertifikate ermöglichen Die Anforderung ist erfüllt, wenn Policies, die in der angebotenen Policy-Sprache definiert sind und in passendem Format vorliegen, mit Hilfe eines Tools in Proxy Zertifikate integriert werden können.

A3.2 Das System muss die Integration von Policies in Job-Proxies ermöglichen. Diese Anforderung ist erfüllt, wenn in Job-Proxies, die bei der Verwendung der GRAM-Schnittstelle auf entfernten Ressourcen erstellt werden, Policies integriert werden können.

A4 - GRAM-Erweiterung

Der für das Abschicken von Anfragen an die GRAM-Schnittstelle verwendete GRAM-Client muss für die Umsetzung der feingranularen Delegation mit Proxy Zertifikaten geändert, beziehungsweise erweitert werden. Der Submit eines Jobs ist momentan nur mit einem full Proxy möglich. Ein limited Proxy hat dazu keine Berechtigung. Beim Abschicken eines Jobs mit einem begrenzten Zertifikat gibt es nun zwei Möglichkeiten:

1. Das aktuelle Zertifikat enthält keine Policies, sondern diese sind in der Proxy-Kette. In diesem Fall muss nichts geändert werden. Der aktuelle Proxy ist ein full Proxy ohne Beschränkungen und das Abschicken des Jobs damit möglich. Die Policies aus der Kette werden auf Ressourcen-Seite trotzdem gesammelt und durchgesetzt.
2. Es befinden sich Policies im aktuellen Zertifikat. GRAM muss hier also so geändert werden, dass auch restricted Proxies akzeptiert werden. Es geht dabei nichts an Sicherheit verloren, da alle Policies durch den neuen Ansatz auf Ressourcen-Seite umgesetzt werden.

Eine weitere Änderung ist nötig, um, nach der endgültigen Autorisierung der Anfrage, einen Passenden Job Proxy zu erstellen. Im aktuellen Ansatz wird hier ein limited Proxy auf der Ressource erstellt. Wurde allerdings ein Begrenzter Proxy zum Abschicken des Jobs verwendet, so ist dies nicht möglich. In diesem Fall muss ein full Proxy erstellt werden, der ja dann die Begrenzungen des Sitzungsproxies über seine Proxy-Kette erbt.

A4.1 GRAM muss so geändert werden, dass restricted Proxies beim Submit von Jobs verwendet werden können. Diese Anforderung ist erfüllt, wenn zum Abschicken eines Jobs neben limited Proxies auch restricted Proxies verwendet werden können und die GRAM-Schnittstelle dies akzeptiert.

A4.2 Bei der Verwendung eines begrenzten Proxy beim Submit, muss anstatt eines limited ein full Proxy als Job-Proxy erstellt werden. Diese Anforderung ist erfüllt, wenn, je nach verwendetem Zertifikat beim Submit, der passende Job-Proxy erstellt wird.

A5 - Umsetzung von Begrenzungen auf Ressourcenseite

Auf Client-Seite müssen, wie in den vorhergehenden Anforderungen festgelegt, Möglichkeiten zur Beschränkung der delegierten Rechte auf explizite Aufgaben gegeben sein und die Definition dieser Begrenzungen soll computergestützt durchgeführt werden können. Sind diese Anforderungen erfüllt, so ist es ebenfalls notwendig, dass die so erreichte feingranulare Delegation von Nutzerrechten auch entsprechend umgesetzt wird. Dies muss auf Ressourcen-/Zielseite geschehen. Wurden gewisse Rechte mittels Proxy Zertifikaten weitergegeben, um es Entitäten zu erlauben, im Namen anderer Aktionen durchzuführen, so muss beim Eingang einer Anfrage auf einer Ressource überprüft werden, ob die aktuelle Entität tatsächlich vom ursprünglichen Halter der Rechte vorgesehen wurde, um eben die angefragten Aktionen durchzuführen.

Die Ziele müssen also in der Lage sein, die definierten Begrenzungen syntaktisch zu verstehen, und deren Semantik beim Autorisierungsvorgang mit einzubeziehen. Die im Proxy Zertifikat enthaltenen Regeln müssen zuerst gesammelt werden. Dies betrifft, wie in Kapitel 2.2.3 bereits erwähnt, nicht nur das Zertifikat selbst, sondern die komplette Zertifikats-Kette. Im Anschluss müssen die Regeln, entsprechend der verwendeten Policy-Sprache, zusammengefasst werden können. Dann muss die Semantik der Regeln verwendet werden, um die Anfrage zu überprüfen und eine Entscheidung zu treffen. Wurden der anfragenden Entität beispielsweise die Rechte für Anfragen an eine Ressource B von einer Entität A übertragen, nicht aber die Rechte für die Durchführung von Stage-in und Stage-out, so muss eine Anfrage für einen GRAM-Job auf Ressource A mit vorangehendem Stage-in abgelehnt werden. Soll allerdings nur ein erlaubtes Skript ausgeführt werden, und ist A autorisiert, Anfragen an B zu schicken, so muss diese Anfrage angenommen werden.

A5.1 Wurden im Rahmen des feingranularen Delegationsansatzes auf Client-Seite Beschränkungen definiert, so muss das System diese auf Ziel-/Ressourcen-Seite syntaktisch verstehen und entsprechend der verwendeten Policy-Sprache zusammenfassen können. Diese Anforderung ist erfüllt, wenn bei einer Anfrage an eine Ressource die Policies der kompletten Proxy-Kette des zur Authentifizierung benutzten Zertifikats gesammelt und zusammengefasst werden.

A5.2 Das System muss die für die Autorisierungsentscheidung nötigen Informationen aus der Anfrage extrahieren können. Diese Anforderung ist erfüllt, wenn der Entscheidungsfunktion des Autorisierungsansatzes alle nötigen Informationen bereitstellt und diese darauf aufbauend eine Entscheidung treffen kann.

A5.3 Das System muss, aufgrund der gesammelten Policies, Anfragen überprüfen und Autorisierungsentscheidungen treffen können. Diese Anforderung ist erfüllt, wenn die auf Client-Seite aufgestellten Policies berücksichtigt und semantisch korrekt umgesetzt werden.

A6 - Benutzbarkeit des zu entwickelnden Systems

In dieser Anforderung geht es darum, ein gewisses Maß an Benutzbarkeit des zu entwickelnden Systems zu garantieren. Wichtige Stichworte sind dabei Systemverständlichkeit, Sys-

temlernbarkeit und Systembedienbarkeit. Der zu entwickelnde Ansatz soll derart gestaltet werden, dass er leicht verständlich ist und die Einarbeitung für einen Benutzer, der davor noch nicht mit den bereitgestellten Mechanismen zur feingranularen Begrenzung delegierter Rechte in Kontakt gekommen ist, in kurzer Zeit ermöglicht wird. Wie in Anforderung A1 bereits erwähnt, soll der Zeitaufwand gering gehalten werden, um ein effizientes Arbeiten in Globus-Umgebungen weiterhin zu gewährleisten. Deshalb ist es wichtig, nicht nur die Einarbeitungszeit in die grundsätzlichen Mechanismen, sondern auch das tatsächliche Hantieren mit ihnen, kurz und unkompliziert zu halten. Die Definition von Einschränkungen soll bei der Zeitplanung keine wichtige Rolle spielen müssen.

A6.1 Das System soll es einem Grid-User, der noch nicht mit den Mechanismen zur feingranularen Delegation von Rechten gearbeitet hat, ermöglichen, sich in kurzer Zeit einzuarbeiten und sowohl Syntax, als auch Semantik der Begrenzungsmöglichkeiten zu verstehen. Diese Anforderung ist erfüllt, wenn ein Nutzer mit Globus-Erfahrung innerhalb weniger Minuten sowohl die Syntax von zu erstellenden Regeln, als auch die Semantik des Ansatzes, nachvollziehen kann.

A6.2 Die Erstellung/Definition von Beschränkungen soll für einen eingearbeiteten Globus User in kurzer Zeit durchgeführt werden können. Diese Anforderung ist erfüllt, wenn das Erstellen von Begrenzungen delegierter Rechte, für eine bestimmte Aufgabe, innerhalb weniger Minuten möglich ist.

A7 - Integration der Erweiterungen in bestehenden Ansatz

Die Erweiterung des Delegationsansatzes in GT5 um Mechanismen zur feingranularen Delegation sollen in den bestehenden Ansatz integriert werden. Die bereits vorhandenen Delegationsoptionen sollen dabei weiterhin nutzbar bleiben. Nutzer, die bislang mit den Delegationsstufen full und limited gearbeitet haben, sollen diese Option auch weiterhin haben. Jeder sollte im Fall von Rechtedelegation die Wahl haben, wie hoch er die Sicherheitsstandards für sich anlegt. In [Sch10] wird die Nichtwiderlegbarkeit bei der Verwendung von Proxy Zertifikaten besprochen. Wichtig ist dabei der uneingeschränkte Sitzungsproxy, der für die weiteren Aktionen eines Nutzers im Grid verwendet und in der eigenen Domäne mit dem eigenen EEC signiert wird. Der Nutzer gibt bei der Verwendung von Proxy Zertifikaten seine Rechte weiter und mit diesen können Aktionen durchgeführt werden, die er zu verantworten hat. Ist er dabei weniger restriktiv, so erhöht er zwar sein eigenes Risiko, schadet damit aber keinem anderen.

Aus Gründen der Rückwärtskompatibilität sollen außerdem auch Anwendungen und Dienste, die auf den bisherigen Mechanismen basieren, weiterhin funktionsfähig bleiben. Die Erweiterungen sollen dabei so integriert werden, dass der User bei der Verwendung der bisherigen Mechanismen keinerlei Beeinflussung erfährt. Damit kann die Rückwärtskompatibilität vollständig erreicht werden.

A7.1 Das System muss die Delegationsmöglichkeiten full- und limited Proxy weiterhin zur Verfügung stellen. Diese Anforderung ist erfüllt, wenn beim Abschicken eines Grid Jobs an eine Ressource nach wie vor limited und full Proxies delegiert werden

können.

A7.2 Das System muss bisherige Standardvorgänge der Delegationsmechanismen erhalten. Diese Anforderung ist erfüllt, wenn das System bei Nicht-Verwendung der Erweiterung das gewohnte Verhalten zeigt. Wird beispielsweise ein GRAM-Job an eine Ressource geschickt, so soll dort, wie bisher ein limited Job-Proxy erstellt werden.

3.2.3 funktionale Anforderungen und Use Cases

In diesem Abschnitt sollen nun speziell die funktionalen Anforderungen des Ansatzes weiter spezifiziert und unterteilt werden. Unter funktionalen Anforderungen versteht man, wie zu Beginn des Kapitels erwähnt, gewünschte, konkrete Funktionalitäten, die in einer Implementierung umgesetzt werden können.

Mit Hilfe von Use-Case-Diagrammen, und darauf aufbauenden Use-Case-Beschreibungen, sollen erarbeitete Anforderungen geordnet und genauer beleuchtet werden.

Methodik

Um Softwaresysteme besser beschreiben zu können, bietet die Unified Modeling Language (UML), welche von der Object Management Group (OMG) spezifiziert wird¹, verschiedene Arten von Diagrammen. Die UML eignet sich daher sehr gut dazu, die erarbeiteten Anforderungen zu modellieren und in die Dokumentation einzuarbeiten.

Eine der Diagrammart ist das Use Case-Diagramm, das später, zusammen mit zugehörigen Use-Case-Beschreibungen, verwendet wird, um die aufgestellten Anforderungen zu konkretisieren und zu ordnen. Die angesprochenen Diagramme bestehen aus den Elementen **Use Case**, **Akteur** und **System**. Ein Use Case beschreibt eine Abfolge von Aktionen, die, wenn sie nacheinander durchgeführt werden, ein bestimmtes Ergebnis liefern. Damit kann beispielsweise eine 'typische Interaktion eines Anwenders mit einem System' ([Rup07]) gezeigt werden.

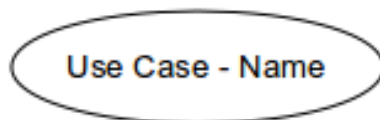


Abbildung 3.2: Darstellung eines Use Case

Ein Teil der Anforderungen, nämlich diejenigen, die das 'gewünschte externe Systemverhalten aus Sicht des Anwenders' ([Rup07]) betreffen, können damit beschrieben werden. Dargestellt werden Use Cases, wie in Abbildung 3.2 zu sehen, durch eine Ellipse, mit dem Namen als Inhalt.

Die Use Cases beschreiben also das Verhalten, das durch das System realisiert wird. Das System selbst wird dabei durch ein Rechteck, welches die Systemgrenzen repräsentiert und zentriert am oberen Rand den System-Namen enthält, dargestellt (Abbildung 3.3).

Der letzte wichtige Bestandteil eines Use Case-Diagramms ist der Akteur.

¹<http://www.omg.org/spec/UML/>

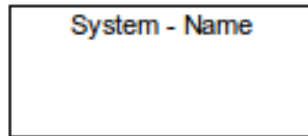


Abbildung 3.3: Darstellung des Systems

Er spezifiziert eine Rolle, die mit dem System interagiert und entweder von einem Nutzer, oder auch von einem anderen System, aus dem System-Kontext, gespielt werden kann. Dargestellt wird ein Akteur, in der Regel, durch ein Strichmännchen mit dem Namen darunter, wie es in Abbildung 3.4 zu sehen ist.



Abbildung 3.4: Darstellung eines Akteurs

Die bereits erwähnte Use Case-Beschreibung soll die Anwendungsfälle weiter konkretisieren. Diese Notation ist, laut [Rup07], 'eine gute, semi-formale Notation, um grobe Anwenderforderungen zu dokumentieren'. Es gibt dafür jede Menge Formularvorlagen, welche zum Beispiel in [Coc01], zusammen mit vielen anderen Informationen zur Use Case-Beschreibung, zu finden sind. In dieser Arbeit wird eine Vorlage mit den Feldern *ID*, *Beschreibung*, *Vorbedingung*, *Ergebnis*, *Normalablauf*, *Nachbedingung* und *Hergeleitet aus* verwendet. Die Felder, mit den jeweiligen Erklärungen, sind nochmals in Abbildung 3.5 zu sehen.

Feld	Inhalt
ID	Eindeutige ID aus Nummerierung und Titel des Use Case
Beschreibung	Kurze Beschreibung
Vorbedingungen	Optionale Bedingungen, die zur Ausführung des Use Case erfüllt sein müssen
Normalablauf	Abfolge der einzelnen Schritte des Use Case
Nachbedingungen	Optionale Bedingungen, die nach der Ausführung des Use Case gelten müssen
Hergeleitet aus	Informelle Anforderungen, die die Grundlage für diesen Use Case bilden

Durchführung

Aus den erarbeiteten informellen Anforderungen (Kapitel 3.2.2) und den zuvor festgelegten Zielen (Kapitel 3.2.1) werden in diesem Abschnitt Use Cases und beteiligte Akteure abgeleitet.

AKTEURE

Es gibt nur eine Rolle, die mit dem zu entwickelnden System interagiert. Es handelt sich dabei um den *Grid-User*, welcher im Folgenden kurz beschrieben wird.

Grid-User - Dieser Akteur beschreibt einen Anwender des Globus Toolkit. Er arbeitet also in einer Grid-Umgebung und nimmt dafür die dynamische Delegation von Nutzerrechten in Anspruch. Um die von ihm delegierten Rechte zu Begrenzen, interagiert er also mit dem System zur feingranularen Delegation mit Hilfe von Policies in Proxy Zertifikaten.

USE CASES

In Abbildung 3.5 ist das System *Feingranulare Delegation mit Proxy Zertifikaten* mit seinem Akteur, den vier Use Cases und seinen Systemgrenzen zu sehen.

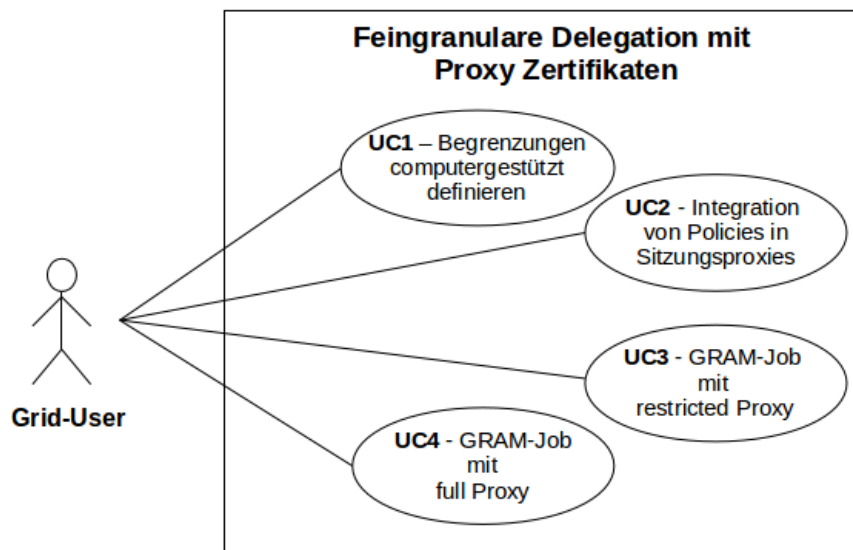


Abbildung 3.5: System mit Akteuren und Use-Cases

Im Folgenden werden die Anforderungen, anhand von Anforderungsbeschreibungen nach der präsentierten Vorlage, weiter konkretisiert.

ID	UC1 - Begrenzungen computergestützt definieren
Beschreibung	Computergestützte Erstellung von Policies in vorgegebener Policy-Sprache
Vorbedingungen	Nutzer hat sich für bestimmte Begrenzungen entschieden
Normalablauf	1. Nutzer startet Tool zur Erstellung. 2. Das System gibt Anweisungen und der Nutzer tätigt Eingaben, entsprechend der gewünschten Begrenzungen. Er spezifiziert dabei Begrenzungen in verschiedenen Kategorien 3. Policies werden aus Eingaben des Nutzers in der vorgegebenen Sprache erzeugt und in vorgegebenem Format gespeichert.
Nachbedingungen	Policies liegen in vorgegebener Sprache und vorgegebenem Speicherformat vor
Hergeleitet aus	A1.1, A2.1, A2.2, A2.3

ID	UC2 - Integration von Policies in Sitzungsproxies
Beschreibung	Proxy Zertifikat wird erstellt und zuvor definierte Policies zur Begrenzung delegierter Rechte in PCI Erweiterung integriert
Vorbedingungen	Policies wurden erstellt und liegen in vorgegebenem Format vor
Normalablauf	1. Nutzer ruft Befehl zur Integration, unter Angabe der erstellten Policies, auf 2. Proxy Zertifikat wird erzeugt und Policies entsprechend RFC 3820 in PCI Erweiterung eingebaut
Nachbedingungen	Proxy Zertifikat mit gewünschten Begrenzungen liegt vor
Hergeleitet aus	A3.1

ID	UC3 - GRAM-Job mit restricted Proxy
Beschreibung	Abschicken eines GRAM-Jobs an eine entfernte Ressource unter Verwendung eines Proxys mit integrierten Policies
Vorbedingungen	Proxy mit integrierten Policies liegt an entsprechendem Speicherort
Normalablauf	1. User benutzt GRAM zum Submit eines Jobs an eine entfernte Ressource 2. Der Client und der Gatekeeper der entfernten Ressource führen wechselseitige Authentifizierung erfolgreich durch. Dabei wird komplette Proxy-Kette zum Gatekeeper übertragen und zur Validierung des User-Zertifikats benutzt 3. System sammelt sämtliche Policies aus den PCI Erweiterungen der Zertifikate der zuvor übertragenen Kette 4. GT5 Autorisierung mit Grid-Mapfile wird erfolgreich durchgeführt 5. System extrahiert die nötigen Informationen aus der Anfrage des Nutzers 6. System trifft, unter Beachtung gesammelter Policies, positive Zugriffsentscheidung über die eingegangene Anfrage 7. Erstellung eines neuen full Job-Proxy auf entfernter Ressource. Policies sind in dessen Proxy-Kette enthalten. 8. Durchführung des Jobs
Nachbedingungen	Endgültige Zugriffsentscheidung ist getroffen und Job ist angenommen
Hergeleitet aus	A4.1, A3.2, A4.2, A5.1, A5.2, A5.3

Es gibt zu Use Case UC3 einige wichtige Sekundärszenarien. Scheitert in Schritt zwei die wechselseitige Authentifizierung, so wird der Job nicht angenommen und die Transaktion ist beendet. Dies passiert ebenso, wenn in Schritt vier kein passender Eintrag für den Distinguished Name des User-Zertifikats gefunden wird und das Mapping auf einen Systemuser somit nicht möglich ist. Der User hat auf der Ressource nicht die Berechtigungen zum Durchführen der angefragten Aufgabe. Verhindert eine, aus der Proxy-Kette extrahierte, Policy die Durchführung der Anfrage, so wird diese ebenfalls, aufgrund fehlender Berechtigung, durch die Ressource abgelehnt und die Transaktion abgebrochen.

ID	UC4 - GRAM-Job mit full Proxy
Beschreibung	Abschicken eines GRAM-Jobs an eine entfernte Ressource unter Verwendung eines Proxys ohne Policies
Vorbedingungen	Full Proxy liegt an entsprechendem Speicherort
Normalablauf	1. und 2. siehe UC3 3. System registriert full Proxy und führt eine normale GT5 Autorisierung durch 4. siehe UC3 5. Erstellung eines neuen Job-Proxy auf der entfernten Ressource unter Beachtung bei der Anfrage verwendeter Optionen. Standard ist ein limited Proxy 8. Durchführung des Jobs
Nachbedingungen	Autorisierung abgeschlossen und Job angenommen
Hergeleitet aus	A7.1, A7.2

Auch in Use Case UC4 gibt Sekundärszenarien. Scheitert in Schritt zwei die wechselseitige Authentifizierung, so wird der Job nicht angenommen und die Transaktion ist beendet. Dies passiert ebenso, wenn in Schritt vier kein passender Eintrag für den Distinguished Name des User-Zertifikats gefunden wird und das Mapping auf einen Systemuser somit nicht möglich ist. Der User hat auf der Ressource nicht die Berechtigungen zum Durchführen der angefragten Aufgabe und die Transaktion wird abgebrochen.

3.2.4 Nicht-funktionale Anforderungen

Nicht-funktionale Anforderungen sind Anforderungen an die Qualität, in der die geforderte Funktionalität erbracht werden soll. Inbegriffen sind beispielsweise auch Randbedingungen für das zu entwickelnde System, wie unterstützte Plattformen, oder auch relevante Gesetze und Verordnungen.

Die Nicht-funktionale Anforderung an das System kann nicht mit Hilfe eines Use Cases dargestellt werden und ist deshalb hier gesondert aufgelistet.

NFA1 - *Benutzbarkeit des Systems*

Ein Globus-User soll durch die Verwendung des feingranularen Delegationsansatzes nicht störend lange von der Umsetzung seiner eigentlichen Aufgaben abgehalten werden. Dies betrifft sowohl die Einarbeitungszeit, und damit das Verstehen der Policy-Sprache und der damit verbundenen Mechanismen, als auch das aktive Arbeiten mit den Begrenzungsmöglichkeiten.

→ abgeleitet aus A6

4 Verwandte Arbeiten

In diesem Kapitel soll gezeigt werden, wie die Delegation in wichtigen Middlewares organisiert ist und vor allem, ob eine feingranulare, auf spezielle Jobs zugeschnittene Weitergabe von Rechten, ermöglicht wird. Der erste Teil behandelt dabei Ansätze, die auf Proxy Zertifikaten basieren, im zweiten Teil wird die Delegation in Unicore genauer betrachtet.

4.1 Delegation mit Proxies in anderen wichtigen Middlewares

Ein Mittel zur dynamischen Delegation von Nutzerrechten sind die in Kapitel 2.2.3 vorgestellten Proxy Zertifikate. Sie werden von einigen wichtigen Middlewares eingesetzt und sind im Grid-Umfeld sehr verbreitet. Es soll nun gezeigt werden, wie das Konzept in verschiedenen wichtigen Middlewares eingesetzt ist und ob dabei die PCI Erweiterung zur regelbasierten Begrenzung übertragener Rechte eingesetzt wird.

4.1.1 Globus Toolkit 4/5

Ein Großteil der Globus Toolkit 4 (GT4) Komponenten basiert auf Web Services und dem Web Services Resource Framework (WSRF)¹, einer Sammlung von OASIS-Spezifikationen für Web Services zur Realisierung statusbehafteter Ressourcen. In Abbildung 4.1 ist ein Überblick zu sehen, der eine Trennung von ws-basierten Komponenten und solchen, die nicht auf Web Services basieren, vornimmt. Wichtige Komponenten wie GRAM, oder Mechanismen für Authentifizierung und Autorisierung, sowie die Delegation, sind hier als Web Services umgesetzt. Teile davon sind zwar auch in Pre-WS-Versionen vorhanden, dies dient aber lediglich der Kompatibilität mit früheren Versionen. In Globus Toolkit 5 sieht das grundlegend anders aus. In Abbildung 2.1 sind die Komponenten zu sehen. Basis für Version 5 ist der Code der pre-ws Version 2 des Toolkits, was zur Folge hat, dass sämtliche wichtigen Komponenten nicht mehr auf Web Services aufbauen.

Die grundlegenden Mechanismen zur dynamischen Delegation von Nutzerrechten unterscheiden sich in den beiden Versionen hingegen kaum. Die dynamische Delegation von Nutzerrechten wird mit Hilfe von Proxy Zertifikaten umgesetzt. Der zu Beginn jeder Sitzung erstellte Sitzungsproxy wird für alle weiteren Transaktionen verwendet und ein Single Sign-On dadurch garantiert. Die seit Version 4.2 standardmäßig eingesetzten *RFC 3820 compliant Proxies*, die in vollem Umfang RFC 3820-konform sind, bilden auch in Version 5 des Toolkits den Standard. Sie beinhalten die in Kapitel 2.2.3 beschriebene Proxy Certificate Information Erweiterung und erlauben somit die Begrenzung der mit den Zertifikaten delegierten Rechte über Policies. Wie in [Sch10] beschrieben, sind in Globus Toolkit 4 durchaus Möglichkeiten zur Beschränkung delegierter Rechte umgesetzt. Gültigkeitsdauer und Pfadlänge können begrenzt und limited Proxies verwendet werden, aber eine feingranulare Beschränkung auf bestimmte Aufgaben, um nur die tatsächlich benötigten Rechte zu delegieren, ist nicht möglich.

¹http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf#technical

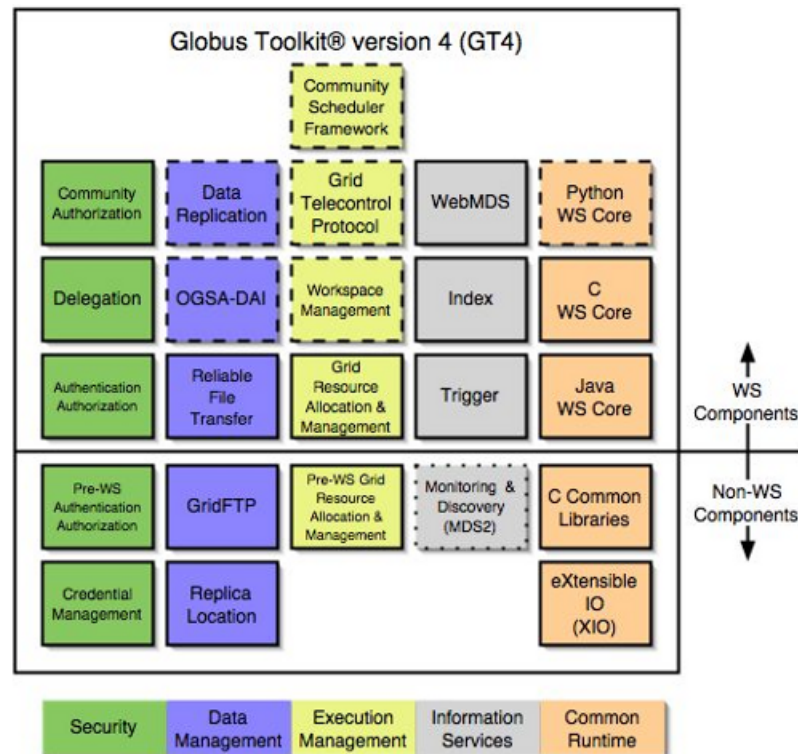


Abbildung 4.1: Globus Toolkit 4 - Überblick

Die in RFC 3820-konformen Proxies enthaltene Option, Regeln zu integrieren, wird nicht genutzt. Dies ist in Version 5 weiterhin der Fall. Es gibt hier einige kleine Änderungen, beispielsweise beim Erneuern von PCs, aber die Begrenzungsmöglichkeiten sind dieselben, wie in GT4 und regelbasierte Begrenzungsmöglichkeiten **nicht** umgesetzt.

4.1.2 gLite

gLite - *Lightweight Middleware for Grid Computing*² ist ebenfalls eine Middleware zum Aufbau von Grid-Umgebungen. Sie wurde vom Enabling Grids for E-science (EGEE) Projekt³ ins Leben gerufen und wird aktuell von der European Middleware Initiative (EMI)⁴ weiterentwickelt. Entstanden ist gLite im Zusammenhang mit der Entstehung des Large Hadron Colliders (LHC)⁵ am CERN, der Europäischen Organisation für Kernforschung⁶. Die dort durchgeführten Simulationen und Berechnungen produzieren riesige Datenmengen, die dann analysiert werden müssen. Bedarf an Rechenkraft und Speicher sind enorm, und um die nötigen Ressourcen besser verteilen zu können, wird das ganze in einer Grid-Infrastruktur organisiert, dem Worldwide LHC Computing Grid (WLCG)⁷. Als Middleware wird dort gLite verwendet.

²<http://glite.cern.ch/>

³<http://www.eu-egee.org/>

⁴<http://www.eu-emi.eu/>

⁵<http://www.lhc-facts.ch/>

⁶www.cern.ch

⁷<http://lcg.web.cern.ch/lcg/public/>

In einem gLite Grid existieren Storage Elements (SE), welche einen einheitlichen Zugriff auf Datenspeicherungs-Ressourcen ermöglichen. Computing Elements hingegen stellen Ressourcen dar, welche Berechnungen ausführen können und über ein GridGate(GG) mit dem Cluster kommunizieren, in das sie eingefügt sind. Eine weitere wichtige Komponente der Middleware ist das Workload Management System (WMS). Es akzeptiert Job-Anfragen und ermittelt die für die Ausführung am besten geeigneten Computing Elements mit Hilfe des sogenannten *match-making*-Prozesses. Ausgeführt wird WMS auf dem Resource Broker (RB). Zur Job-Beschreibung wird die Job Description Language (JDL) verwendet.

Wie das Globus Toolkit implementiert gLite die Grid Security Infrastructure und basiert auf einer PKI, die Zertifikate zur Verfügung stellt. Für sichere Kommunikation sorgt SSL/TLS und Proxy Zertifikate ermöglichen einen Single Sign-On und dynamische Delegation. Dabei gibt es zwei Ausprägungen von Proxies. Normale Proxy Zertifikate, und solche, die Informationen über Virtuelle Organisationen (VO) enthalten, sogenannte VOMS-Proxies. Zur Erstellung wird das Tool `voms-proxy-init` verwendet. Dabei werden bei einem, oder mehreren, VO Membership Service (VOMS) Informationen über Gruppenzugehörigkeit und Rollen eingeholt, und über eine nicht-kritische Erweiterung in ein normales PC integriert. Diese Form der Proxies existiert auch in GT. Sie ermöglicht es Ressourcen, den Zugriff auf sie genauer festzulegen. Nutzer können damit in verschiedenen VOs verschiedenen Gruppen angehören und unterschiedliche Rollen bekleiden. Im WLCG ist beispielsweise ausschließlich diese Art der Proxies zugelassen. Autorisierung kann hier auf zwei verschiedenen Wegen durchgeführt werden. Zum einen über den starren, unflexiblen Ansatz, der Entscheidungen auf Grundlage des Grid-Mapfiles fällt und ein Mapping auf Systemkennungen durchführt, um die Durchführung von Jobs zu ermöglichen. Der andere Ansatz trifft seine Entscheidungen anhand der im VOMS-Proxy enthaltenen VO-Informationen. Zugriff wird hier also anhand von Rollen und Gruppenzugehörigkeiten gestattet, oder verboten. Ermöglicht wird dadurch allerdings nicht die Begrenzung delegierter Rechte durch den Nutzer, sondern lediglich eine flexiblere Autorisierung durch die Ressourcen und eine bessere Handhabung von komplexen VOs. Auch in gLite ist somit keine feingranulare Begrenzung der Rechte durch den Nutzer möglich. Durch die Verwendung von Proxy Zertifikaten und der Implementierung der Grid Security Infrastructure, wären die Möglichkeiten zur feingranularen Begrenzung über Policies auch hier gegeben, sie sind aber nicht umgesetzt.

4.1.3 ARC Middleware

Die Entwicklung der Advanced Resource Connector (ARC) Middleware⁸ wird von Nordugrid⁹ koordiniert. Momentan liegt sie in Version 11.05 vor. Verschiedene Services daraus sind auch in EMI 1 Kebnekaise¹⁰ der European Middleware Initiative (EMI)¹¹ enthalten, welche Komponenten der vier größten Middlewareanbieter Europas (ARC, dCache, gLite, Unicore) vereint.

ARC implementierte früher die GSI, mit SSL für sichere Kommunikation. Dies wurde aber mittlerweile in den Sicherheitskomponenten geändert. Teile der Sicherheitsfunktionalität werden über Plug-ins implementiert, die konfigurierbar sind und dynamisch geladen werden können. Eine Erweiterung der Sicherheitsmechanismen ist daher leicht umsetzbar.

⁸www.nordugrid.org/arc/

⁹<http://www.nordugrid.org/>

¹⁰<http://www.eu-emi.eu/emi-1-kebnekaise>

¹¹<http://www.eu-emi.eu/>

GSI-basierte Mechanismen werden, aus Kompatibilitätsgründen, weiterhin unterstützt. Viele der etablierten Dienste wurden weiterentwickelt und nutzen nun eine auf Web Services basierende Architektur.

Wichtige Komponenten eines ARC Grids sind Speicherelemente (ARC Storage Elements), sowie Berechnungselemente (ARC Computing Elements). Letztere beinhalten den ARC Resource-coupled EXecution service (A-REX), der Anfragen samt Job-Beschreibungen akzeptiert und diese auf darunterliegenden Batch Systemen ausführt. Als Beschreibungssprachen werden die Extended Resource Specification Language (XRSL), sowie die Job Submission Description Language (JSDL) akzeptiert. Der Client erlaubt auch die Job Description Language, welche dann in XRSL oder JSDL übersetzt wird.

ARC bietet außerdem einen Service Container mit dem Namen Hosting Environment Daemon (HED). Dort können verschiedene Services auf Applikations- und Protokoll-Ebene gehostet werden. Es ist außerdem ein Framework für die Implementierung und Umsetzung von Authentifizierung und Autorisierung vorhanden. Die Funktionalitäten werden, wie bereits erwähnt, über Plug-ins realisiert, die sogenannten Security Handlers (SecHandler). Diese können wiederum auf andere Teil-Module zugreifen, welche verschiedene Sicherheitsfunktionen, wie Authentifizierung und Autorisierung, handeln. Beispiele dafür sind verschiedene Policy Decision Point (PDP) Komponenten ([QK]).

Die Authentifizierung basiert auch beim ARC, wie in vielen anderen Grid-Umgebungen, auf einer Public Key Infrastruktur und dem Konzept der Proxy Zertifikate. Neben den üblichen zeitlichen Begrenzungen kann allerdings in feinerem Maße bestimmt werden, welche Rechte delegiert werden. Die in RFC 3820 spezifizierte Möglichkeit zur Integration von Policies in PCs ist hier umgesetzt und die Regeln können angehängt werden. Eine Begrenzungsmöglichkeit, wie sie im Globus Toolkit durch limited Proxies existiert, ist in ARC nicht vorgesehen. Auch independent PCs sind nicht möglich, es gibt lediglich Proxies, die alles erben, und solche, die durch Policies begrenzt sind.

Um die Policies aus Zertifikaten zu sammeln, die zur Authentifizierung verwendet werden und um diese anschließend zu verarbeiten und darauf basierend Entscheidungen zu treffen, müssen Ressourcen den Security Handler *delegation.collector* und das PDP-Plugin *delegation.pdp* verwenden. Wird ein Service im HED geladen, so werden sowohl die SecHandler der Komponente geladen, als auch die dazu angegebenen Plug-ins. Der Delegation Collector holt die Policy-Informationen aus dem entfernten Proxy und steckt sie in ein Sicherheitsattribut, um sie Komponenten wie dem Delegation PDP zur Verfügung zu stellen. Dieser nimmt die Attribute, ruft die *Policy evaluation engine* auf, die wiederum die Anforderung bezüglich der Delegations-Policies auswertet, und gibt abschließend das Resultat dieser Auswertung zurück ([QK]).

Die Delegation wird in ARC über das *Delegation Interface* realisiert. Will ein Service im Namen von Client-Identitäten handeln, so muss er dieses Interface implementieren. Der Client kontaktiert den Service mit einer dafür vorgesehenen Message. Dieser antwortet mit einem Proxy Request und einer ID. Der Client schickt daraufhin ein serialisiertes, signiertes Zertifikat, sowie die ID, um die Verknüpfung mit dem Request herzustellen. Der Service bestätigt abschließend mit einer Nachricht. Die ID kann vom Client mehrmals wiederverwendet werden, um das Proxy Zertifikat aufzufrischen und damit einen Abbruch des Jobs aufgrund eines abgelaufenen Zertifikats zu verhindern ([QK]).

Zum Erstellen von X509 Proxies stellt ARC ein Kommandozeilenwerkzeug zur Verfügung. *arcproxy* erlaubt dem Nutzer sowohl die Erstellung von normale PCs, als auch von VOMS Proxies und vereint somit die Funktionalitäten, die in GT in *grid-proxy-init* und

voms-proxy-init aufgeteilt sind. Mit Hilfe des Tools können sowohl Zertifikate mit, als auch solche ohne Policies erstellt werden. Mit der Option *-c constraint* können dabei Beschränkungen angegeben werden. Mögliche Constraints sind dabei *validityStart*, *validityEnd*, *validityPeriod* zur Begrenzung der Gültigkeitsdauer, sowie *proxyPolicy* und *proxyPolicyFile* für die direkte Angabe der zu integrierenden Policies, beziehungsweise dem Zeiger auf die Datei, die die Regeln enthält. Unterstützt werden momentan ARC Policies, grundsätzlich kann aber jede Sprache verwendet werden, die auf der umsetzenden Ressource verstanden wird. ARC Policies sind XML-basiert, ihr Schema ist in [Waa] zu sehen. Der Aufbau ähnelt stark dem Aufbau der XACML Policies, die in Kapitel 2.3 vorgestellt wurden.

```

1
2   Policy (1)
3     Rule (1-)
4       Subjects (1)
5         Subject (1-)
6           Attribute (1-)
7     Resources (0-1)
8       Resource (1-)
9     Actions (0-1)
10      Action (1-)
11     Conditions (0-1)
12      Condition (1-)
13      Attirbute (1-)

```

Listing 4.1: Aufbau einer ARC Policy

Eine Policy besteht aus einer oder mehreren *Rules*. Diese wiederum aus einem Tupel aus *Subjects*, *Resources*, *Actions*, *Conditions*. Jede Regel hat dabei als Entscheidung entweder *Permit* oder *Deny*. Im Baum in Listing 4.1 sieht man die Zusammenhänge der Hauptelemente und die jeweiligen Multiplizitäten. Von den Hauptelementen einer Rule muss nur Subjects zwingend vorkommen. Die restlichen drei sind optional. Jedes Hauptelement muss, wenn vorhanden, mindestens ein Unterelement haben. Ist beispielsweise Actions angegeben, so muss mindestens ein Action-Element darin enthalten sein.

ARC Policies sind XACML Policies also sehr ähnlich. Sie sind allerdings weniger komplex und dafür aber auch weniger allgemein und flexibel. Eine ARC Policy kann also zum Beispiel nicht von einer XACML Evaluation-Engine verstanden werden, sondern ausschließlich von der *ARC policy evaluation engine*.

Der Ansatz des Advanced Resource Connectors setzt also, im Gegensatz zu GT und gLite, eine feingranulare Beschränkung delegierter Rechte durch den Nutzer um. Nativ unterstützt wird dabei eine eigens für das Projekt definierte, an XACML angelehnte Policy-Sprache. Sie ist zwar weniger komplex als XACML, bedarf aber trotzdem einer gewisse Einarbeitungszeit. Außerdem bietet ARC kein Tool an, das den Nutzer bei der Erstellung von Policies unterstützt. Erst die Integration kann dann mit dem *arcpolicy* Kommando vorgenommen werden. Es sind lediglich PCs mit vollen Rechten, und solche mit regelbasierter Begrenzung zugelassen. Ein einfacher Mechanismus, wie die limited Proxies des Globus Toolkit, die das Starten neuer Prozesse verbieten, sonst aber den Stellvertreter mit umfassenden Rechten ausstatten, ist nicht vorgesehen.

Einige der in Kapitel 3.2 aufgestellten Anforderungen werden also nicht erfüllt. Außerdem

haben sich die ARC-Entwickler von der Grid Security Infrastructure abgewandt und ARC Services sind größtenteils WS-basiert. Beim Globus Toolkit ist eine gegenläufige Entwicklung zu erkennen. Basierten in GT4 noch viele wichtige Dienste auf Web Services, so ist man davon in Version 5 wieder abgekommen und hat auf den Code der pre-ws Version 2 des Toolkits aufgebaut. Die zugrunde liegenden Ansätze sind also sehr verschieden. Das zu erstellende System wird von der grundsätzlichen Umsetzung der feingranularen Delegation her einen ähnlichen Ablauf haben, wie er in ARC zu finden ist. Es sollen eben auch regelbasierte Begrenzungen in RFC 3820-basierte PCs integriert, und diese auf Ressourcenseite ausgewertet werden. Die Umsetzung muss sich aber an den Gegebenheiten orientieren, die im Globus Toolkit 5 vorzufinden sind. Außerdem sollen die Anforderungen aus Kapitel 3.2 vom zu entwickelnden System möglichst vollständig erfüllt werden.

4.2 Unicore und feingranulare Delegation

Wie bereits in Kapitel 4.1.3 erwähnt, ist Uniform Interface to Computing Resources (Unicore)¹² Teil der European Middleware Initiative und eine der wichtigsten europäischen Middlewares. Unicore wird am Forschungszentrum Jülich¹³ entwickelt und betreut und setzt auf Standards und Erweiterbarkeit. Als Grundlage dient die Open Grid Services Architecture (OGSA), eine Service orientierte Architektur, die statusbehaftete Web Services voraussetzt. Diese werden vom Web Service Resource Framework (WSRF) spezifiziert. Den Authentifizierungs- und Autorisierungsmechanismen liegen auch hier, wie in den anderen vorgestellten Middlewares, X509 Zertifikate zugrunde. Außerdem werden Erweiterungen für die Zugriffskontrolle, basierend auf Proxy Zertifikaten und VOs, bereitgestellt.

In Abbildung 4.2 ist die Unicore-Architektur dargestellt. Wie zu sehen ist, gibt es für den User verschiedene Möglichkeiten. Es existiert ein Kommandozeilen-Client (UCC), sowie ein auf Eclipse basierender und ein Rich Client (URC). Außerdem kann über Portale, wie beispielsweise Grid Sphere, in einem Unicore-Grid gearbeitet werden. Außerdem wird Entwicklern die High Level API for Grid Applications (HiLA) bereitgestellt, die es ermöglicht, eigene Clients zu schreiben und Unicore 6 in Nutzer-Anwendungen zu integrieren.

Bei Transaktionen im Unicore-Grid, kontaktiert der Client grundsätzlich den Gateway seiner Anlaufstelle. Es findet eine Authentifizierung statt, die ausschließlich mit End User Zertifikaten durchgeführt wird. Dies unterscheidet Unicore vom Globus Toolkit und anderen Middlewares, bei denen Proxy Zertifikate zur Authentifizierung verwendet werden. Wie in [Sch10] diskutiert, bringt dies eine Steigerung der Transparenz, da eine Unicore-Ressource immer weiß, mit wem sie es gerade zu tun hat. Eine Globus-Ressource kann sich lediglich sicher sein, wer der ursprüngliche Aussteller des zur Authentifizierung genutzten Zertifikats ist, nicht aber, mit wem er es aktuell zu tun hat. Dem User stehen verschiedene Dienste wie *Service Registries*, *Workflow Engines*, oder *Service Orchestrators* zur Verfügung. Hinter dem Gateway einer Unicore-Site, befindet sich die WSRF-basierte Hosting-Umgebung. Diese ist wiederum über ein sogenanntes *Target System Interface* mit den lokalen Resource Managern, wie Torque oder LSF, verbunden. Als Job-Management und -Ausführungs-Engine wird XJNS verwendet. Sie erfüllt eine ähnliche Aufgabe, wie WMS in gLite und ist Grundlage vieler Unicore-Services.

¹²<http://www.unicore.eu/unicore>

¹³<http://www.fz-juelich.de>

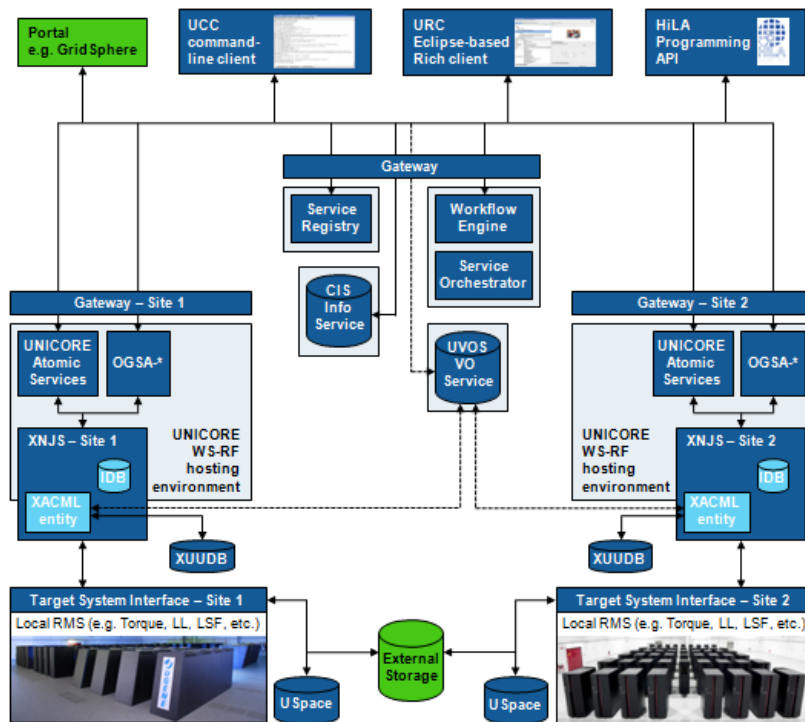


Abbildung 4.2: Unicore - Überblick

Zugriff darauf ist über die *Atomic Services*, das Unicore-proprietäre Interface zu XNJS, sowie über einen Satz standardisierter Interfaces, die als OGSA-* bezeichnet werden, möglich. Zwei dieser standardisierten Interfaces sind OGSA-BES und HPC-P. Als Beschreibungssprache für Jobs wird in Unicore JSDL verwendet.

Dynamische Delegation wird in Unicore nicht über Proxy Zertifikate, sondern über die sogenannte Explicit Trust Delegation (ETD) zur Weitergabe von Rechten umgesetzt. Genutzt werden hier Security Assertion Markup Language (SAML)¹⁴ Assertions, welche im Header von Simple Object Access Protocol (SOAP)¹⁵ Nachrichten transportiert werden können und so gut in den WS-basierten Ansatz von Unicore integrierbar sind.

SAML ist ein xml-basierter Standard, der zum Austausch von Authentifizierungs- und Autorisierungsdaten genutzt werden kann. Informationen über eine Identität werden über sogenannte Assertions weitergegeben. Diese enthalten unter anderem einen *Issuer*, ein *Subject* und einen *Custodian*. Der Issuer gibt dabei die Rechte an das Subject weiter, der Custodian ist bei einer einfachen Rechtedelegierung gleich dem Issuer. SAML Assertions können aber auch verkettet werden, um die Rechte über mehrere Stationen weiterzugeben. In diesem Fall ist der Custodian die Identität, die die erste Assertion der Kette ausgestellt hat.

Dieser Ansatz hat, gegenüber der Delegation mit Proxy Zertifikaten, den Vorteil, dass sämtliche an der Kette beteiligten Stationen bekannt sind. Der Ansatz bietet also ein Plus an Transparenz. Eine feingranulare Weitergabe von Rechten wäre hier über XACML Policies in den Assertions durchaus möglich. Doch in der aktuellen Version 6 von Unicore ist dies nicht vorgesehen. Mit jeder neuen Assertion in der Kette werden die kompletten Rechte des

¹⁴<http://saml.xml.org/saml-specifications>

¹⁵<http://www.w3.org/TR/soap12-part1/>

Subjects weitergegeben. Auch hier, wie beispielsweise in GT, ist also keine Einschränkung auf bestimmte Aufgaben und Jobs möglich.

4.3 Evaluation

Bei den hier vorgestellten Middlewares handelt es sich um die bekanntesten Vertreter im Bereich Grid Computing. Drei der Ansätze verfolgen eine Delegation unter Verwendung von Proxy Zertifikaten. Im Globus Toolkit gibt es dabei, zusätzlich zu den üblichen Proxy-spezifischen Beschränkungen, noch die Möglichkeit von limited PCs. Außer ARC, erlaubt aber keine der erwähnten Middlewares eine feingranulare Delegation von Rechten, welche ein Abstimmen der Rechte auf jeweils bevorstehende Aufgaben ermöglicht. Der Advanced Resource Connector basiert auf Web Services und verfolgt deshalb einen grundsätzlich anderen Ansatz, als das Globus Toolkit 5. Das in dieser Arbeit zu entwickelnde System soll aber eben gerade einen feingranularen Delegationsmechanismus für GT5 bereitstellen.

Aufgrund fehlender Delegationsmöglichkeiten in den anderen Middlewares, oder der Verwendung anderer Grundlagen, sind viele der in Kapitel 3.2 aufgestellten Anforderungen entweder nicht erfüllt, oder es kann keine Aussage über die Erfüllung gemacht werden. Es soll im Folgenden trotzdem eine Evaluation durchgeführt werden.

Es wird dafür die in Abbildung 4.3 gezeigte Schablone verwendet. Links steht der Name des ausgewerteten Ansatzes, die unterschiedlichen Kästchen stehen für die verschiedenen Use Cases, sowie die nicht-funktionalen Anforderungen. Ein Fragezeichen bedeutet hier, dass über die Erfüllung der entsprechenden Anforderungen keine Aussage gemacht werden kann. Ein grüner Haken bedeutet *Anforderung erfüllt*, das rote Kreuz steht hingegen für eine Nichterfüllung. Ist ein Fragezeichen zu sehen, das rechts oben einen kleinen grünen Haken hat, so ist die Anforderung grundsätzlich erfüllt, es werden aber andere technische Mechanismen und Grundlagen verwendet. Dies ist beispielsweise der Fall, wenn zum Abschicken eines Jobs vom entsprechenden Ansatz nicht GRAM verwendet wird, der Submit aber mit einem Begrenzten Zertifikat möglich ist.

Ansatz	UC1	UC2	UC3	UC4	NFA1
	✗	✔	✗	? <small>✔</small>	?

Abbildung 4.3: Evaluations-Schablone

4.3.1 Globus Toolkit 4/5

In Abbildung 4.4 ist die Zusammenfassung der Auswertung der Anforderungen bezüglich Globus Toolkit Version 4 und 5 zu sehen. Da sich die beiden Versionen in ihren Delegationsmöglichkeiten nicht grundlegend unterscheiden, können sie hier zusammengefasst werden.

Da die Definition feingranularer Beschränkungen im Globus Toolkit grundsätzlich nicht gegeben ist, kann das Erstellen natürlich auch nicht computergestützt vorgenommen werden. Die Integration von Policies mit Hilfe des `grid-proxy-init` Befehls wird allerdings unterstützt. Die Optionen `-pl` und `-policy` lassen sowohl die Angabe eines Object Identifiers, als auch

einer Datei, die die gewünschten Policies enthält, zu. Ein Beispiel für die Durchführung ist in einem späteren Kapitel zu sehen (Abbildung 5.1). Der Submit eines GRAM-Jobs ist aufgrund der fehlenden Mechanismen zur Auswertung integrierter Policies auf Ressourcen-Seite nicht möglich, das Abschicken eines Jobs mit einem full Proxy natürlich schon, da dies das Standardvorgehen in GT4/5 ist. Da kein System zur feingranularen Delegation vorhanden ist, können auch keine Aussagen über die Benutzbarkeit des selbigen gemacht werden. Über die Erfüllung von NFA1 ist also keine Aussage möglich.

	UC1	UC2	UC3	UC4	NFA1
GT4/5	✗	✔	✗	✔	?

Abbildung 4.4: Globus Toolkit 4/5 - Evaluation

4.3.2 gLite

Abbildung 4.5 zeigt die Auswertung für die Middleware gLite. Auch hier ist kein Mechanismus zur feingranularen Delegation von Rechten durch den Nutzer vorgesehen. UC1 ist deshalb logischerweise nicht erfüllt. Auch das *voms-proxy-init* Tool, zur Erstellung von Proxies mit VO Informationen, bietet die Möglichkeit, Policies in Zertifikate zu integrieren. Dafür werden, wie bei *grid-proxy-init*, die Kommandozeilenoptionen *-pl* und *-policy* für die Policy Language und die zu integrierende Policy-Datei benutzt. Da keine feingranulare Delegation möglich ist und die Nutzer in gLite die WMS-Schnittstelle nutzen, um ihre Jobs an die entsprechenden Ressourcen zu schicken, können keine Aussagen über UC3 und UC4 gemacht werden. GRAM wird nicht eingesetzt. Allerdings ist der Submit eines Jobs mit einem full Proxy möglich. Über die Benutzbarkeit kann auch hier keine Aussage gemacht werden, da die Mechanismen nicht vorhanden sind.

	UC1	UC2	UC3	UC4	NFA1
gLite	✗	✔	✗	? ✔	?

Abbildung 4.5: gLite - Evaluation

4.3.3 ARC

Beim Advanced Ressource Connector sieht die Auswertung etwas anders aus. Eine feingranulare Begrenzung der delegierten Rechte ist hier, im Gegensatz zu den anderen Ansätzen, möglich. Es können also Jobs mit begrenzten Zertifikaten abgeschickt werden. Policies werden mit Hilfe des *arcproxy* Tools in die Zertifikate integriert, und können dann mit dem *arcsub* Tool an A-REX Computing Elements geschickt werden. Es wird also nicht die GRAM-Schnittstelle verwendet. Es ist natürlich ebenfalls möglich, auf diese Weise Jobs unter Verwendung eines full Proxy abzuschicken. NFA1 ist nicht erfüllt. Die Einarbeitungszeit sollte für Nutzer mit XACML-Kenntnissen zwar recht schnell gehen, alle anderen müssen sich

aber erst orientieren. Außerdem ist kein Tool zur Unterstützung beim Erstellen von Policies vorgesehen. Dies muss also manuell geschehen, wofür mehr Zeit eingerechnet werden muss.

	UC1	UC2	UC3	UC4	NFA1
ARC	✗	✓	? ✓	? ✓	✗

Abbildung 4.6: ARC - Evaluation

4.3.4 Unicore

In Abbildung 4.7 sieht man nun abschließend die Evaluation für Unicore. Dort wird grundsätzlich ein anderer Delegationsansatz verwendet. Hier spielen nicht Proxy Zertifikate, sondern SAML Trust Delegation Assertions, die in SOAP-Headern transportiert werden, die entscheidende Rolle. Feingranulare Begrenzungen sind aber auch hier nicht möglich, obwohl SAML im Zusammenspiel mit XACML diese Option im Prinzip bieten würde. Jobs werden über den Unicore Commandline Client (UCC) an Ressourcen geschickt. GRAM ist auch hier nicht im Einsatz.

	UC1	UC2	UC3	UC4	NFA1
Unicore	✗	✗	✗	? ✓	?

Abbildung 4.7: Unicore - Evaluation

Keiner der vorgestellten Ansätze erfüllt also alle Anforderungen, die in Kapitel 3.2 gesammelt wurden. Nur ARC bietet überhaupt Mechanismen zur feingranularen Delegation von Nutzerrechten. Hier sind aber trotzdem einige Forderungen nicht erfüllt, außerdem sind die technischen Grundlagen gänzlich anders, als beim Globus Toolkit. In GT ist in der aktuellen Version eine feingranulare Delegation nicht umgesetzt. Es muss also eine Erweiterung durchgeführt werden, um die durchaus vorhandenen Möglichkeiten nutzen zu können.

5 Entwurf

Das Authentifizierungs- und Autorisierungskonzept des Globus Toolkit soll um die feingranulare Begrenzung der mit PCs delegierten Rechte erweitert werden. Dafür wird die Eigenschaft von Proxy Zertifikaten, Policies in einer Extension, der sogenannten PCI Extension, aufnehmen zu können, genutzt. Der erste Teil dieses Kapitels zeigt die Stellen des aktuellen Ansatzes, an denen Änderungen vorgenommen werden müssen, um die gestellten Anforderungen zu erfüllen. Dies geschieht vorerst unabhängig von der verwendeten Policy-Sprache, welche dann im zweiten Teil des Kapitels behandelt wird. Dort werden Bereiche des Konzepts noch einmal aufgegriffen, falls sie sprachspezifische Eigenschaften haben.

5.1 Sicherheitskonzept - nötige Änderungen (sprachunabhängig)

In Kapitel 3.1.2 wurde erläutert, wie Authentifizierung und Autorisierung momentan im Standardfall in globus-basierten Grids aussehen. Abbildung 3.1 soll nun verwendet werden, um genau zu zeigen, an welchen Stellen Änderungen nötig sind. In der Zusammenfassung zu diesem Abschnitt wird dann eine erweiterte Version der Abbildung zu finden sein.

5.1.1 Client - Erstellung und Einbringung von Policies

Bei der Nutzung regelbasiert beschränkter Proxy Zertifikate ist der erste Schritt des Gesamtvorgangs die Erstellung einer Policy durch den Nutzer. Diese muss syntaktisch korrekt sein, sich also an die Vorgaben der verwendeten Policy-Sprache halten, und soll semantisch die gewollten Beschränkungen ausdrücken.

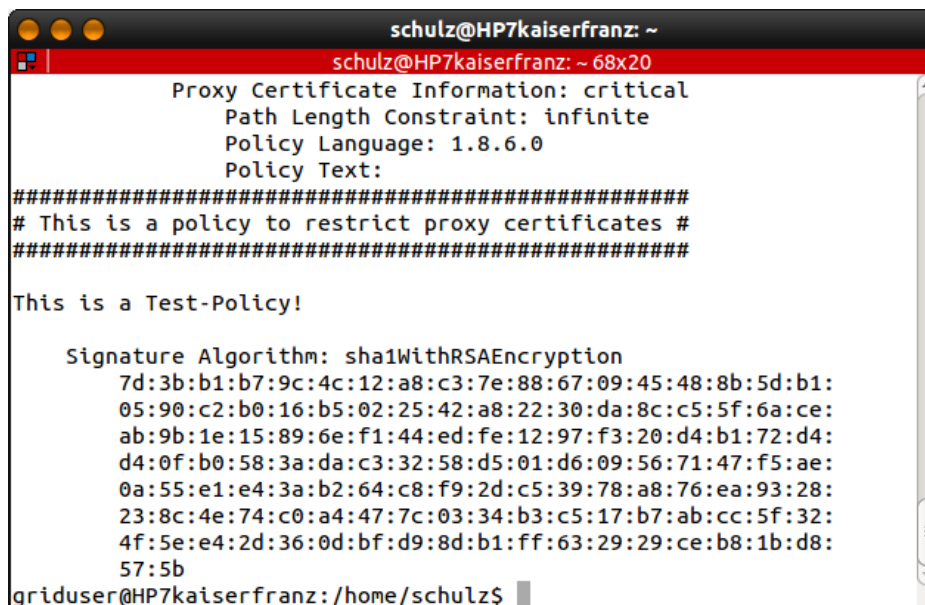
Damit die Nutzer sich nicht zu ausführlich mit der Syntax beschäftigen müssen und formale Fehler möglichst verhindert werden können, soll ein Werkzeug für die automatisierte Erstellung von Regeln zur Verfügung gestellt werden. Hierbei wäre entweder ein Kommandozeilen-Tool, oder aber eine Anwendung mit graphischer Oberfläche denkbar. Da ein Globus-User bei der Arbeit mit dem Toolkit grundsätzlich mit der Kommandozeile vertraut sein muss und die wichtigsten Globus Komponenten auch über diese gesteuert werden, ist ein Konsolen-Skript der richtige Weg. Wie dieses genau aufgebaut wird und wie User und System dabei miteinander interagieren, ist wiederum sehr stark abhängig von der gewählten Policy-Sprache.

Ist die Policy syntaktisch richtig und den semantischen Wünschen entsprechend erstellt, so muss sie im Anschluss in ein PC eingebracht werden, um den gewünschten Effekt zu erreichen. Es gibt verschiedene Szenarien, die ein durch Policies beschränktes Proxy Zertifikat beinhalten. Eine Möglichkeit ist die Integration mit Hilfe des *grid-proxy-init* Tools, das standardmäßig in Globus Toolkit 5 enthalten ist. Mit diesem Werkzeug werden Proxy Zertifikate erstellt. Es wird beispielsweise benutzt, um beim Start einer Grid Sitzung einen Sitzungsproxy zu erstellen. Dieser wird dann für alle weiteren Aktionen des Nutzers verwendet und ermöglicht so einen Single Sign-On. Da der private Schlüssel eines Proxy Credentials grundsätzlich nicht passwortgeschützt ist und ein Diebstahl nie ausgeschlossen werden kann, ist natürlich auch hier eine Begrenzung sinnvoll. Noch wichtiger ist jedoch eine Begrenzung

von Zertifikaten, die weitergegeben werden und somit die eigene Domäne, wie auch die eigene Kontrolle verlassen.

Der *grid-proxy-init* Befehl kann mit vielen Optionen ausgeführt werden. Es kann zwischen den verschiedenen Ausprägungen von Proxy Zertifikaten gewählt werden, limited- und independent Proxies erstellt, sowie Laufzeit und Pfadlänge des Zertifikats festgelegt werden. Außerdem gibt es Optionen für die OID einer Policy-Sprache (-oid) und für die Angabe einer Policy-Datei, die über die PCI Erweiterung in das zu generierende Zertifikat integriert wird (-policy). Bei der Verwendung von *grid-proxy-init* sind die in RFC 3820 festgelegten Möglichkeiten zur policy-basierten Beschränkung von Proxies also umgesetzt und können standardmäßig genutzt werden.

Als Beispiel wird mit dem Befehl *grid-proxy-init -pl 1.8.6.0 -policy /home/schulz/Desktop/-Diplomarbeit/sourcen/policy.txt* ein neues Proxy Zertifikat erstellt. Die OID ist hier zufällig gewählt, die Policy-Datei besteht aus einigen Zeilen Text. Ein Teil des erstellten Proxies, mit Hilfe von openssl in Textform ausgegeben, ist in Abbildung 5.1 zu sehen. Die kritische PCI Erweiterung hat ein Feld für die im Befehl angegebene OID und auch die Policy-Datei ist integriert. Lässt man sich mit dem Befehl *grid-proxy-info* die Eigenschaften des Zertifikats anzeigen, so ist dort unter anderem der Typ des Proxys angegeben. Die Bezeichnung lautet *RFC 3820 compliant restricted proxy* und zeigt einerseits, dass in GT5 RFC 3820 konforme PCs Standard sind, andererseits, dass das Zertifikat Beschränkungen anhand einer Policy Datei enthält.



```

schulz@HP7kaiserfranz: ~
schulz@HP7kaiserfranz: ~ 68x20
Proxy Certificate Information: critical
Path Length Constraint: infinite
Policy Language: 1.8.6.0
Policy Text:
#####
# This is a policy to restrict proxy certificates #
#####
This is a Test-Policy!

Signature Algorithm: sha1WithRSAEncryption
7d:3b:b1:b7:9c:4c:12:a8:c3:7e:88:67:09:45:48:8b:5d:b1:
05:90:c2:b0:16:b5:02:25:42:a8:22:30:da:8c:c5:5f:6a:ce:
ab:9b:1e:15:89:6e:f1:44:ed:fe:12:97:f3:20:d4:b1:72:d4:
d4:0f:b0:58:3a:da:c3:32:58:d5:01:d6:09:56:71:47:f5:ae:
0a:55:e1:e4:3a:b2:64:c8:f9:2d:c5:39:78:a8:76:ea:93:28:
23:8c:4e:74:c0:a4:47:7c:03:34:b3:c5:17:b7:ab:cc:5f:32:
4f:5e:e4:2d:36:0d:bf:d9:8d:b1:ff:63:29:29:ce:b8:1b:d8:
57:5b
griduser@HP7kaiserfranz:/home/schulz$

```

Abbildung 5.1: Policy mit OID und Policy

Wird ein so erzeugtes Zertifikat zum Abschicken eines Jobs verwendet, so sind die eingebrachten Begrenzungen auch für das neu erzeugte Zertifikat auf der entfernten Ressource gültig und der Nutzer hat so die Möglichkeit, einem Job genau die Rechte zu delegieren, die dieser für die Durchführung benötigt.

Um dies zu ermöglichen und gleichzeitig bei Bedarf auch eine Trennung von Sitzungs-Proxy und Job-Proxy zu erlauben, müssen Änderungen am GRAM-Client, beziehungsweise der

Kommunikation des selbigen mit der entfernten GRAM-Schnittstelle vorgenommen werden. Zuallererst muss der Delegationsmechanismus geändert werden, der beim Abschicken eines Jobs ein neues Proxy Credential auf der angefragten Ressource erstellt. Das zugehörige Zertifikat wird dabei natürlich vom Auftraggeber signiert. Bisher wird im Standardfall ein limited Proxy erstellt, das dem Job das Erstellen neuer Prozesse unmöglich macht. Sind die Rechte dafür allerdings erforderlich, so hat der Nutzer die Möglichkeit, bei der Verwendung von *globusrun* oder darauf basierender Befehle, wie *globus-job-submit* und *globus-job-run*, mit der Option *-full-proxy* alle seine Rechte zu übertragen. Außerdem ist zum Abschicken eines Jobs ein full Proxy nötig, Anfragen mit limited Proxies werden abgelehnt.

Die GRAM-Mechanismen müssen erweitert, beziehungsweise geändert werden, um die Handhabung von restricted Proxies zu ermöglichen. Mit limited Proxies soll das Abschicken von Jobs weiterhin nicht möglich sein. Wird ein full Proxy verwendet, so muss für die Erstellung des Job-Proxies eine Unterscheidung getroffen werden. Auf Ressourcenseite muss überprüft werden, ob in der Proxy-Kette des verwendeten Zertifikats Policies enthalten sind. Ist dies der Fall, so soll ein Full Proxy abgeleitet und auf der Ressource erstellt werden. Die Begrenzungen sind im neuen Zertifikat über die Kette ebenfalls enthalten. Wird beim Submit des Jobs ein full Proxy verwendet und es befinden sich keine Beschränkungen in der Kette, so soll standardmäßig, wie bisher, ein limited Proxy übertragen werden. Bei der Verwendung der Option *-full-proxy* werden weiterhin alle Rechte weitergegeben. Wird beim Abschicken ein restricted Proxy verwendet, so kann einfach ein full Proxy abgeleitet werden, da die Begrenzungen auch hier in der Kette des neuen Zertifikats enthalten sind und deshalb beachtet werden.

Um zusätzlich die Trennung von Sitzungsproxy und Job-Proxies zu ermöglichen, wird eine zusätzliche Option geschaffen, um auch dem *globusrun* Befehl die OID einer Sprache und eine Policy-Datei übergeben zu können. Es muss dafür die Kommunikation zwischen Client und Ziel-Gatekeeper so abgeändert werden, dass beim erstellen des neuen Proxys die nötigen Informationen, also OID und Datei, zur Verfügung stehen und dann integriert werden können. Da dies ja auch beim *grid-proxy-init* Befehl geschieht, können die hier nötigen Funktionen davon abgeleitet werden, beziehungsweise sind in der C-API des Globus Toolkit vorhanden. Wird diese Option dann beim Job-Submit verwendet, so werden übergebene Policies integriert und gelten zusätzlich zu denen, die schon in der Proxy-Kette vorhanden waren. In Abbildung 5.2 ist nochmal zusammengefasst, welche Rechte ein Job-Proxy in Abhängigkeit vom verwendeten Sitzungsproxy und gewählter Optionen erhält.

5.1.2 Authentifizierung und Autorsierung

Zusätzlich zu den Änderungen am Client, die gerade eben beschrieben wurden, müssen auch Änderungen am Globus Gatekeeper vorgenommen werden. Der zu entwickelnde Ansatz muss die wichtigsten Komponenten des in RFC 2904¹ vorgestellten Autorisierungsframeworks realisieren, um damit Zugriffskontrollentscheidungen, basierend auf den von Nutzern definierten Begrenzungen in Proxy Zertifikaten, treffen zu können.

Die Komponenten und ihre Beziehungen zueinander sind in Abbildung 5.3 zu sehen.

¹<http://www.ietf.org/rfc/rfc2904.txt>

Verwedeter Proxy	Optionen	Abgeleiteter Proxy	Ergebnis
limited	/	/	Anfrage abgelehnt
full (Begrenzung in der Kette)	/	full (Begrenzung über Kette)	Abhängig von Gesamtautorisierung
	full		
	pl, policy	restricted	
full (Keine Begrenzung in der Kette)	/	limited	
	full	full	
	pl, policy	restricted	
restricted	/	Full (Begrenzung über Kette)	
	full		
	pl, policy	restricted (zusätzliche Regeln)	

Abbildung 5.2: Delegationsmöglichkeiten für Job-Proxy

Die Aufgaben, die die einzelnen Teile dabei verrichten müssen, sind denen der Komponenten des XACML Frameworks, das in Kapitel 2.3.1 beschrieben wurde und auf RFC 2904 aufbaut, sehr ähnlich. Die Zugriffsentscheidung wird vom PDP getroffen. Er hat keinen Kontakt zur Ressource und deshalb auch keinerlei Schutzfunktion für sie. Er bekommt Anfragen und fällt Entscheidungen über Annahme oder Ablehnung selbiger. Der PEP steht hingegen schützend zwischen angefragter Ressource und der anfragenden Entität. Die vom PDP getroffenen Entscheidungen werden also hier umgesetzt. Warum der PDP die eine, oder andere Entscheidung trifft, ist für den PEP nicht von Interesse. Der Policy Retrieval Point sammelt die Policies, die der PDP für seine Entscheidung benötigt und der Policy Information Point liefert weitere, für die Entscheidungen nötige Informationen.

Gatekeeper - Authentifizierung: Sammeln der Policies

Eine der Besonderheiten von Proxy Zertifikaten ist, dass sie sowohl von EECs, als auch von anderen Proxy Zertifikaten signiert werden können. End Entity Zertifikate hingegen können nur von einer vertrauenswürdigen CA signiert werden. Ein Effekt, der sich daraus ergibt, ist die Kettenbildung bei PCs. Bei der Validierung eines EECs muss lediglich das Zertifikat der signierenden CA vorhanden sein und dieser CA, beziehungsweise der darüberliegenden RootCA, vertraut werden.

In Abbildung 5.4 ist eine Proxy Kette zu sehen. Das EEC von Bob ist hier vom CA Zertifikat der CA 'rootCA' signiert.

5.1 Sicherheitskonzept - nötige Änderungen (sprachunabhängig)

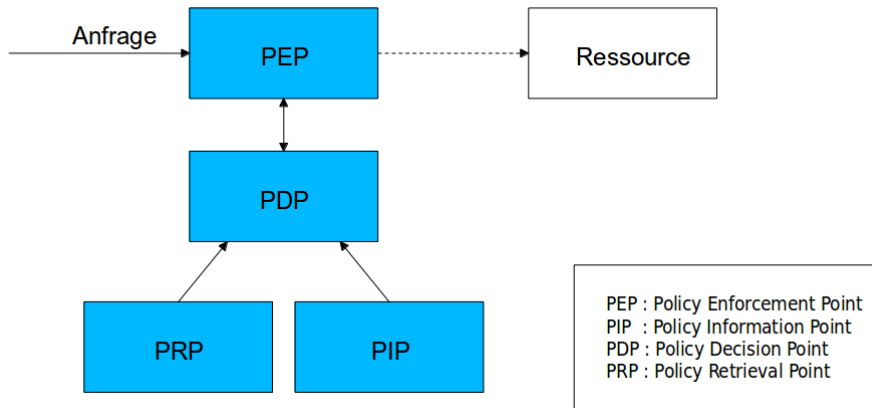


Abbildung 5.3: Komponenten des umzusetzenden Autorisierungsframeworks

Bob erstellt dann ein von diesem End Entity Zertifikat signiertes Proxy Zertifikat, Bobs PC 1, von dem dann ein weiteres Proxy Zertifikat, Bobs PC 2, abgeleitet wird. Die dadurch entstandene Proxy Kette hat einige Besonderheiten bezüglich der Validierung der einzelnen PCs. Um ein Proxy Zertifikat zu validieren benötigt man alle vorherigen PCs der Proxy Kette, das EEC, welches das erste PC signiert hat und das dazugehörige CA Zertifikat. Wurden damit alle Signaturen, bis hin zur CA, validiert und wird der CA vertraut, so wird das Proxy Zertifikat akzeptiert.

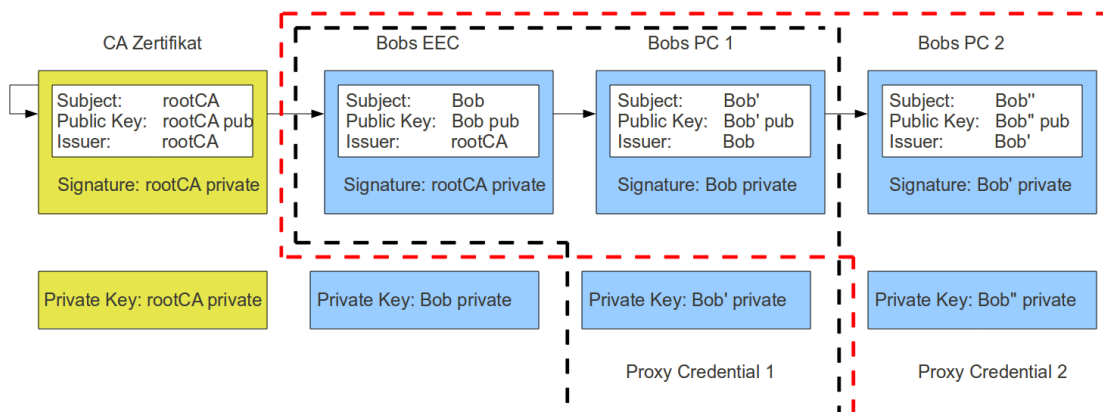


Abbildung 5.4: Bob's Proxy-Kette

Im Beispiel wird die Signatur von PC 2 mit dem öffentlichen Schlüssel von PC 1 überprüft. Die Signatur von PC 1 wiederum mit dem öffentlichen Schlüssel von Bob's EEC. Dann verläuft die Validierung weiter wie bei einem EEC üblich. Die Signatur wird mit Hilfe des CA Zertifikats überprüft und die Certificate Revocation Lists kontrolliert.

Die in Kapitel 2.2 vorgestellte Grid Security Infrastructure wird vom Globus Toolkit implementiert und SSL/TLS wird dabei um die Validierung von Proxy Zertifikaten erweitert. Die

Tatsache, dass für die Validierung eines PCs sämtliche Proxy Zertifikate einer Kette benötigt werden, kann für den Mechanismus der regelbasierten Beschränkung von PCs genutzt werden. Um eine sinnvolle Durchsetzung der Policies zu gewährleisten, müssen natürlich sämtliche Regeln der gesamten Proxy Kette beachtet werden. Beschränkungen in vorhergehenden Zertifikaten müssen, zusätzlich zu denen des aktuell verwendeten, durchgesetzt werden, um einen sinnvollen Mechanismus zu erhalten.

Legt ein Nutzer fest, dass mit einem Zertifikat A nur auf die Ressource XY zugegriffen werden darf, und wird von diesem Zertifikat ein weiteres Zertifikat B abgeleitet, das keine zusätzlichen Beschränkungen erhält, so reicht es nicht, wenn von einer angefragten Ressource nur die Regeln in B überprüft und durchgesetzt werden. Damit wäre B nicht auf XY beschränkt, sondern könnte Anfragen an alle Ressourcen schicken, zu denen der Nutzer grundsätzlich Zugriff hat. Begrenzungen könnten somit durch eine erneute Delegation gelöscht werden. Dies ist natürlich nicht erwünscht. Werden die Regeln der kompletten Kette beachtet, so ist ein Löschen von Regeln, aufgrund der nötigen digitalen Signatur des jeweiligen Ausstellers, nicht möglich.

Da die komplette Kette von Proxy Zertifikaten also zur Authentifizierung eines Nutzers vor Ausführung eines Jobs sowieso bei der Gatekeeper-Komponente auf Ressourcenseite vorhanden sein muss, bietet es sich an, hier auch den Mechanismus zum Sammeln aller in der Proxy-Kette vorhandenen Policies einzubauen. Wie beim Validierungsprozess müssen alle Zertifikate durchgegangen, und falls vorhanden, die jeweiligen Policies extrahiert werden.

Sind diese gesammelt und verfügbar, so muss gewährleistet werden, dass sie auf die eingehende Anfrage angewendet und durchgesetzt werden können. Ob die einzelnen Policies dabei nacheinander angewendet, oder unter Verwendung eines passenden Algorithmus zu einer Policy zusammengefasst werden können, hängt stark von der verwendeten Policy-Sprache ab. Wie das von dieser Arbeit genau umgesetzt wird, soll in Kapitel 5.2 endgültig besprochen werden.

Der Mechanismus zum Sammeln und Zusammenfassen der Policies der kompletten Proxy-Kette, sowie der Bereitstellung für den Prozess der Entscheidungsfindung im weiteren Verlauf, realisiert für den vorgestellten Ansatz also die Policy Retrieval Point Komponente aus RFC 2904. Es werden die Regeln bereitgestellt, anhand derer der PDP dann Entscheidungen trifft.

Gatekeeper - Autorisierung: Durchsetzung der Policies

Sowohl das Erstellen und Einbringen von Policies auf Client-Seite, als auch das Sammeln und Zusammenfassen aller Policies einer Kette im Globus Gatekeeper auf Ressourcenseite, wurden bereits besprochen. Im dritten großen Bereich des Ansatzes geht es nun darum, die Anfragen derart aufzuarbeiten, dass sie anhand gesammelter Policies geprüft werden können und Access Control Entscheidungen möglich sind. Der Algorithmus für diese Entscheidungen muss ebenfalls noch festgelegt werden.

Verwendet ein User eine in den Anforderungen in Kapitel 3.2 enthaltene und durch das hier entwickelte Konzept umgesetzte Beschränkung eines Proxy Zertifikats, so muss diese Beschränkung auf der Seite der Ziel-Ressource auch durchgesetzt werden können. Um das zu gewährleisten, müssen dafür nötige Informationen gesammelt werden. Als Hauptquelle dient hier die von GRAM genutzte und in Kapitel 2.4 kurz angesprochene Resource Specification Language (RSL). Eine an eine Ressource gesendete GRAM Job-Anfrage ist in RSL beschrieben. Die Eigenschaften einer solchen Anfrage können also aus dieser Beschreibung extrahiert

und für den Autorisierungsprozess genutzt werden. Grid-Aktionen und Executables, im Rahmen einer GRAM-Anfrage, können und müssen auf diesem Weg gewonnen werden.

Die Beschränkung der Rechte eines Proxy Zertifikats auf bestimmte Ziele von Anfragen, kann mittels des Distinguished Names der angefragten Ressource überprüft werden. Dieser kann aus dem Host-Zertifikat der Site gewonnen werden.

Für die Durchsetzung von Beschränkungen bezüglich der Quelle von Anfragen, sind das für die Authentifizierung verwendete Zertifikat und der damit verbundene Distinguished Name nicht von Nutzen. Es handelt sich hierbei um ein Proxy Zertifikat, dessen Subjekt, entsprechend RFC 3820, keinerlei Information über die Maschine, auf der es verwendet wird enthält. Weder der aktuelle Auftraggeber, noch eventuelle Zwischenstationen seit der Delegation der Rechte eines Nutzer-EECs an ein Proxy Zertifikat sind nachvollziehbar. Lediglich der ursprüngliche Issuer, beziehungsweise dessen EEC, ist auf diesem Weg zu ermitteln. Welchen Weg die delegierten Rechte danach gegangen sind, ist aufgrund fehlender Informationen über die Besitzer von Proxy Zertifikaten nicht nachvollziehbar.

Die fehlende Transparenz bei der Verwendung von PCs im Globus Toolkit wurde auch in einem Fortgeschrittenenpraktikum zu diesem Thema als ein Hauptdefizit erkannt[Sch10]. Als mögliche Lösung wird dort die Einführung eines neuen Feldes in der PCI Erweiterung erwähnt. Dort könnten Informationen über den Halter eines Proxys eingearbeitet werden. Diese Erweiterung ist allerdings nicht bestand der Aufgabenstellung dieser Arbeit.

Um die Menge möglicher Anfrage-Quellen begrenzen zu können, müssen also andere Informationen genutzt werden. Zur eindeutigen Identifikation einer Entität im Netz bietet sich der Fully Qualified Domain Name (FQDN) an. Dieser identifiziert Objekte im Domain Name System (DNS) weltweit eindeutig und ist deshalb ideal zur Beschränkung der Proxy-Rechte auf bestimmte Anfrage-Quellen.

All diese Informationen sind nötig, um eine gestellte Anfrage so zu beschreiben, dass der PDP anhand aufgestellter Policies entscheiden kann, ob der Zugriff gewährt oder abgelehnt wird. Der ausführende Mechanismus realisiert somit die Aufgaben des PEP aus RFC 2904. Wie eine Anfrage dann konkret aussieht ist von der Policy-Sprache abhängig und wird in Kapitel 5.3 genauer erläutert. Ist die Anfrage derartig aufgearbeitet, so wird der PDP beauftragt, eine Entscheidung zu treffen. Diese wird dann im Anschluss umgesetzt. Dabei müssen keine weiteren Veränderungen am Gatekeeper vorgenommen werden. Die Autorisierung durch die Site, anhand eines Grid-Mapfiles oder eines anderen Mechanismus, findet ja bereits davor statt. Trifft der PDP nun eine positive Zugriffsentscheidung, so ist der weitere Ablauf mit dem Start des Job-Managers und dem eigentlichen Starten des Jobs identisch zum bisherigen Ablauf nach der Autorisierung durch die Site im Globus Toolkit 5.

Der Algorithmus, mit dessen Hilfe Zugriffskontrollentscheidungen unter Berücksichtigung der vorhandenen Policies getroffen werden, ist stark von der verwendeten Policy-Sprache abhängig und wird daher in Kapitel 5.2 genauer behandelt. Dieser Teil des Ansatzes realisiert dann den Policy Decision Point aus RFC 2904. Er erhält die Anfragen vom PEP aufbereitet und die jeweiligen Policies vom PRP bereitgestellt und trifft dann anhand eines Algorithmus eine Access Control Entscheidung.

5.1.3 Zusammenfassung

In diesem Kapitel wurden nötige Änderungen des momentan in GT5 vorzufindenden Ansatzes gezeigt. Da der Ansatz unter Verwendung verschiedener Policy-Sprachen umgesetzt werden kann, wurde dabei auf Sprachunabhängigkeit geachtet. Sprachspezifische Punkte

werden im nächsten Kapitel aufgegriffen.

Wie in Abbildung 5.5 zu sehen, sind für die Realisierung der regelbasierten Beschränkung von Proxy Zertifikaten Änderungen sowohl am Authentifizierungs-, als auch am Autorisierungsvorgang vorzunehmen. Während der Authentifizierung werden zusätzlich die in der Proxy-Kette des verwendeten Zertifikats enthaltenen Regeln gesammelt. Die Autorisierung, wie sie bisher in GT5 stattgefunden hat, bleibt erhalten und wird um die Überprüfung der im verwendeten Zertifikat vorhandenen Policies erweitert. Der endgültige Zugriff ist also sowohl von der Zustimmung der Site, als auch der des Nutzers anhängig. Außerdem schafft das System Möglichkeiten zur Erstellung von PCs mit Begrenzung und für das Einbringen der Policies.

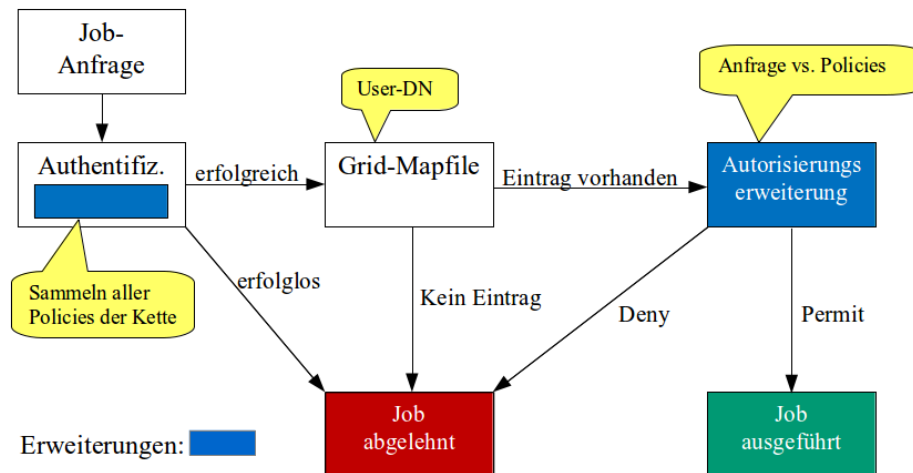


Abbildung 5.5: Sicherheitskonzept GT - Soll-Zustand (stark vereinfachte Form)

5.2 Policy-Sprache

Es wird ein Konzept zur Erweiterung des Globus Toolkit 5 entwickelt, das die in Kapitel 3.2 erarbeiteten Anforderungen nach Möglichkeit vollständig erfüllt und in einer prototypischen Implementierung in einer GT5-Umgebung umgesetzt werden kann. RFC 3820-konforme Proxy Zertifikate sind in vielen GT Komponenten fest verankert. Wie in Kapitel 2.2.3 beschrieben, bieten sie die Möglichkeit einer regelbasierten Beschränkung delegierter Rechte und dienen daher als Grundlage für die geforderten Erweiterungen.

Wo im Ablauf Erweiterungen nötig sind und wie diese theoretisch umgesetzt werden können, wurde in Kapitel 5.1.1 erläutert. Die Policy-Sprache ist dabei eine tragende Säule des erweiterten Ansatzes und einige der behandelten Punkte konnten ohne die Wahl einer konkreten Sprache nicht vollständig besprochen werden. Dies soll nun nachgeholt werden.

5.2.1 Grundlagen

Um eine auf Regeln basierte Beschränkung delegierter Rechte zu ermöglichen, geht es zuallererst darum, eine geeignete Policy-Sprache zu wählen. Es existieren jede Menge Möglichkeiten die dafür in Frage kommen. RFC 3820 macht dazu keine Vorgaben. Es muss nur sicherge-

stellt werden, dass alle beteiligten Entitäten die Sprache sowohl syntaktisch, als auch semantisch handhaben können und verstehen. Auf Nutzerseite müssen die Regeln also syntaktisch korrekt erstellt und in die PCs eingebracht werden. Auf der Zielseite müssen die Regeln verstanden und dann auch korrekt durchgesetzt werden können.

Bei der Wahl der Sprache sind nun gewisse Kriterien zu beachten. Liegt das Hauptaugenmerk auf einem möglichst großen Funktionsumfang, oder sollen hauptsächlich einfaches Verständnis und leichte Handhabung für den Nutzer im Vordergrund stehen? Soll eine einfache, leicht verständliche Implementierung möglich sein, oder lohnt es sich, für mehr Grundfunktionalität und Erweiterungsmöglichkeiten, einen aufwändigeren Weg zu gehen.

Das angestrebte Ziel, die regelbasierte Beschränkung der mit Proxy Zertifikaten delegierten Rechte, kann also über viele verschiedene Wege erreicht werden. Wichtig sollte hier jedoch sein, dass ein Nutzer, der in einem GT-basierten Grid arbeitet, sich nicht erst aufwändig in den Delegations-Mechanismus einarbeiten, oder eine komplizierte Policy-Sprache lernen muss. Es sollte ihm möglich sein, in kurzer Zeit, ohne größeren Aufwand, mit der Arbeit beginnen zu können und trotzdem ein hohes Maß an Sicherheit zu erreichen. Eine Anwendung, die den Nutzer beim einfachen und korrekten Erstellen einer solchen Policy unterstützt, soll im Implementierungsteil der Arbeit umgesetzt werden.

XACML ist ein weit verbreiteter und anerkannter OASIS Standard, der im Grid-Umfeld in verschiedenen Projekten zur Anwendung kommt. Die kurze Einführung in Kapitel 2.3 hat gezeigt, dass es sich dabei um eine XML-basierte Policy-Sprache handelt, die eine ebenfalls XML-basierte Request-/Response-Sprache und ein dazu passendes Zugriffskontroll-Framework mitbringt. Die nötigen Voraussetzungen wären also grundsätzlich gegeben.

Allerdings sind sowohl das Erstellen von Policies, als auch deren Auswertung und Abgleich gegen Zugriffsanfragen keine trivialen Prozesse. Um Policies zu erhalten, die den eigenen Wünschen entsprechen, ist mehr als nur ein oberflächliches Wissen über XACML nötig. Für einen Grid-User, dessen Hauptaugenmerk auf dem effektiven Arbeiten im Grid liegt, ist der damit verbundene Zeitaufwand zu groß. Es müsste ihm eine Anwendung, Kommandozeilen-Tool, oder am besten ein Tool mit graphischer Oberfläche zur Verfügung gestellt werden, mit der er, ohne große Vorkenntnisse und mit erträglichem Zeitaufwand, zu arbeiten beginnen könnte. Die Implementierung dieser Anwendung wäre hier natürlich wegen des komplexen Hintergrundes sehr aufwändig.

Der Komplexe Standard bietet zwar sehr viele Möglichkeiten und die Erweiterung des Delegationsansatzes um zusätzliche Funktionalitäten wäre leicht möglich. Es soll in dieser Arbeit allerdings primär gezeigt werden, wie der Mechanismus zur Beschränkung delegierter Rechte in GT5 aussehen kann und dann eine prototypische Implementierung mit möglichst einfachen Mitteln vorgenommen werden. Die Integration einer XACML Engine würde das ganze Projekt sehr aufblasen und ist für den Nachweis der Machbarkeit nicht zwingend erforderlich. Durch die relative Unabhängigkeit des Konzepts von der verwendeten Policy-Sprache, wäre eine Anpassung an die Nutzung von XACML später relativ leicht machbar und ein Ausbau der Funktionalität wäre so nach wie vor möglich.

Eine weitere Option wäre die Entwicklung einer eigenen, XML-basierten Policy-Sprache. Diese könnte genau auf das Projekt zugeschnitten werden und hätte den Vorteil, dass nicht benötigte Funktionalität gar nicht erst eingeführt werden muss und die Sprache somit schlanker und unkomplizierter bleiben würde, als beispielsweise XACML. Dieser Ansatz wurde in der ARC Middleware mit den ARC Policies gewählt. Diese sind an XACML angelehnt, beinhalten allerdings nur die für das Projekt nötigen Funktionalitäten.

Ein Großteil der Nutzer in Grid-Umgebungen ist mit XML vertraut und die Einarbeitung

würde relativ zügig von statten gehen. Man müsste hier allerdings, anders als bei XACML, wo es bereits verschiedene Implementierungen gibt, sowohl die Sprache definieren, als auch die benötigten Mechanismen zur Auswertung von Anfragen bezüglich bestehender Policies.

Die unkomplizierteste Lösung ist daher die Verwendung einfacher Listen in einer Textdatei, um die verschiedenen Beschränkungen zu realisieren. Auch hier muss die Logik zur Umsetzung der Regeln selbst implementiert werden. Die Erstellung der Policies ist aber denkbar einfach und ohne großen Aufwand machbar. Auch das Zusammenfassen mehrerer Regeln zu einer einzigen, die dann angewendet werden kann, ist, wie sich zeigen wird, mit einem relativ einfachen Algorithmus zu lösen. Es muss allerdings noch gezeigt werden, dass die gestellten Anforderungen damit erfüllt werden können.

Im Folgenden wird die Erstellung von Policies auf der Basis verschiedener Kategorien von Einschränkungen erläutert. Sowohl auf die Kombination aller Policies einer Proxy-Kette, als auch auf die Mechanismen zum Durchsetzen derartiger Regeln auf Ressourcenseite wird dabei eingegangen.

5.2.2 Kategorien von Beschränkungen

Das zu entwickelnde System soll also die feingranulare Begrenzung delegierter Rechte in Globus Toolkit 5-basierten Grids ermöglichen. GT bietet mit dem auf Proxy Zertifikaten basierenden Delegationsansatz bereits die Grundlagen für Beschränkungen. Es soll jetzt also bestimmt werden, welche Begrenzungen nötig sind, und in dieser Arbeit umgesetzt werden sollen. In Kapitel 3.2.2 wurden in der informellen Anforderung A1 die nötigen Begrenzungsmöglichkeiten für einen derartigen Ansatz festgelegt. Es muss sowohl die Möglichkeit geben, Ziele von Anfragen festzulegen, als auch Beschränkungen bezüglich der ausführbaren Dateien und möglicher Grid-Aktionen innerhalb eines GRAM-Jobs.

Diese drei Kategorien von Begrenzungen lassen eine Einschränkung der delegierten Rechte auf eine bestimmte Aufgabe, beziehungsweise einen bestimmten Job zu. Soll beispielsweise ein Job an Ressource A geschickt werden, der Daten von einer Ressource B holen und diese dann als Eingabe für eine ausführbare Datei `/pfad/zum/executable` nutzen soll, so können die Rechte mit den vorgeschlagenen Kategorien sehr genau auf diesen Fall abgestimmt werden. Als Sicherheitsrisiko, das von einem gestohlenen Proxy Zertifikat ausgeht, kann somit stark verringert werden. Anfragen an andere, nicht angegebene Ressourcen sind genauso wenig möglich, wie Aktionen, die für die eigentliche Aufgabe nicht notwendig sind.

Zusätzlich zu den in den Anforderungen festgelegten Begrenzungsmöglichkeiten soll noch eine weitere kommen, um die Sicherheit bei der Verwendung von Proxy Zertifikaten zur dynamischen Delegation in GT5-Umgebungen weiter zu erhöhen. Eine zusätzliche Kategorie soll die Option bieten, Quellen von Anfragen festzulegen. Es geht also darum, von wo aus Anfragen mit einem PC geschickt werden dürfen. Damit kann es Angreifern erschwert werden, gestohlene Zertifikate zu Nutzen und Schaden anzurichten. Bekommt er Zugriff auf ein Proxy Zertifikat, so kann er bisher für den Rest der Gültigkeitsdauer alles machen, wozu der ursprüngliche Nutzer grundsätzlich autorisiert ist. Mit den drei bereits erwähnten Kategorien lassen sich die Rechte auf bestimmte Aufgaben zuschneiden. Kommt diese vierte Kategorie zum Einsatz, so muss das Zertifikat von einer der genannten Maschine aus genutzt werden. Einen Proxy abzufangen und dann einfach von einem beliebigen Ort aus zu nutzen, ist also nicht mehr möglich.

Ein zusätzlicher Effekt bei der Angabe von möglichen Quellen von Anfragen ist, dass ein Zertifikat `limited` ist für alle Sites, die in den möglichen Zielen, nicht aber in den möglichen

Quellen sind. Man kann sich dort mit dem Zertifikat authentifizieren, eine Delegation der Rechte dorthin macht aber keinen Sinn, da das Zertifikat nicht weiter genutzt werden kann. Die vier Kategorien zur Begrenzung delegierter Rechte sind also Ziele von Anfragen (Targets), Quellen von Anfragen (Hosts), sowie erlaubte ausführbare Dateien (Executables) und Grid-Aktionen (Actions) bei GRAM-Anfragen. Diese Kategorien müssen mit der Policy-Sprache, die im nächsten Kapitel (5.2.3) spezifiziert wird, umgesetzt werden können.

5.2.3 Eigene Policy-Sprache

Wie in Kapitel 5.2.1 erläutert, soll der zu entwickelnde Ansatz als Policy-Sprache einfache Listen verwenden. Der Ansatz wird dadurch einfach gehalten. Dabei soll es pro Kategorie eine Liste mit Items geben. Durch die unterschiedliche Semantik, die mit den einzelnen Kategorien beschrieben werden soll, unterscheiden sich natürlich auch die Items.

Formate der Items

Die Liste für die möglichen Targets ist eine Aufzählung der Subject Names der Host-Zertifikate der erlaubten Sites. Es handelt sich hierbei um die Distinguished Names der jeweiligen Entitäten.

Die Liste erlaubter Quellen (Hosts) soll aus Full Qualified Domain Names (FQDN) bestehen. FQDNs sind weltweit eindeutige Bezeichner für Objekte im Domain Name System (DNS). Subject Names, wie bei den Targets, können hier nicht verwendet werden, da es in dieser Kategorie um die Begrenzung der Maschinen geht, von denen aus ein PC genutzt werden kann, und nicht um Identitäten die agieren. Der Subject Name wäre, aufgrund der fehlenden Transparenz bei der Verwendung von Proxy Zertifikaten, auch in diesem Fall nicht geeignet, da der Subject Name eines Ps nichts über die aktuell agierende Entität aussagt, sondern lediglich über den ursprünglichen Issuer eines Zertifikats. Seinem Subject Name wird bei jeder Weitergabe lediglich eine Common Name Komponente mit einer Zufallszahl als Wert angehängt. Eine Lösung für die fehlende Transparenz wäre zwar wünschenswert, ist aber kein Thema dieser Arbeit.

Die Liste der Executables besteht aus absoluten Pfaden zu den ausführbaren Dateien. Würden keine absoluten Pfade verlangt, so könnten Dateien mit identischem Namen im Account des Nutzers abgelegt werden und diese würden anstatt der tatsächlich vorgesehenen ausgeführt. Das Problem bei der Benutzung von absoluten Pfaden besteht wiederum darin, dass sich diese auf verschiedenen Maschinen unterscheiden können. Soll mit einem Proxy ein Job an verschiedene Ressourcen geschickt werden, so müssen eben alle möglichen Pfade angegeben werden.

Grid-Aktionen beschränken sich in dieser Arbeit auf Aktionen im Zusammenhang mit dem Abschicken eines Jobs. Es geht hier ja hauptsächlich um die Machbarkeit eines derartigen Ansatzes und eine prototypische Implementierung. Später können die möglichen Aktionen natürlich erweitert werden, um beispielsweise GridFTP Dateitransfers, oder das Einloggen mit Hilfe von Gsissh. Auch eine Einschränkung bezüglich dem Abbrechen von Jobs oder dem Abfragen von Informationen über solche, wäre denkbar. Es gibt also einen bestimmten Satz an möglichen Aktionen, aus denen der Nutzer die für ihn nötigen aussuchen kann. Zur Verfügung stehen sollen in diesem Ansatz *stage-in*, *stage-out* und *job-submit*, sowie *dryrun* und *authenticate-only*. 'authenticate-only' und 'dryrun' sind Optionen, die vom *globusrun* Tool angeboten werden. Erstere prüft, ob ein Gatekeeper an der adressierten Stelle vorhanden

ist. Es wird eine wechselseitige Authentifizierung durchgeführt und kontrolliert, ob dem verwendeten Client-Credential der Zugriff zum Service gewährt wird. Ist dies der Fall, so wird die Meldung 'GRAM Authentication test successful' zurückgegeben. Falls nicht, so wird eine Erklärung des aufgetretenen Problems ausgegeben. Beim 'dryrun' wird zusätzlich der Job Manager gestartet und alles bis kurz vor dem eigentlichen Submit des Jobs zum Service ausgeführt. Dadurch können Probleme mit der RSL Spezifikation erkannt werden. Der Nutzer hat natürlich auch die Möglichkeit, die erlaubten Aktionen nicht einzuschränken. Dies geschieht, wenn er einfach keine Angaben zu dieser Kategorie macht.

Targets, Hosts und Executables sind also vom Nutzer frei auszuwählen, bei der Wahl der Aktionen, muss er aus einem vorgegebenem Pool wählen, oder keine Beschränkung durchführen. In Abbildung 5.6 sind nochmal die vier Kategorien mit den Formaten ihrer jeweiligen Items aufgeführt. Zu jeder Kategorie ist außerdem ein Beispiel angegeben.

Kategorie	Items
Targets	Subject Names der Host-Zertifikate von Ziel-Sites Beispiel: /O=Grid/OU=GlobusTest/OU=simpleCA-hp7kaiserfranz/CN=host/hp7kaiserfranz.fritz.box
Hosts	FQDN der Quell-Maschinen Beispiel: test1.nm.ifi.lmu.de
Executables	Absolute Pfade zu ausführbaren Dateien Beispiel: /bin/lis
Actions	Möglichkeiten: job-submit, authenticate-only, dryrun, stage-in, stage-out

Abbildung 5.6: Kategorien und ihre Items

Listensemantik

Nun muss noch festgelegt werden, was die verschiedenen möglichen Ausprägungen einer Liste in den unterschiedlichen Kategorien für Bedeutungen haben. Eine Liste, egal welcher Kategorie, kann entweder nicht angegeben sein, vorhanden sein und Items enthalten, oder zwar vorhanden, aber leer sein. Im folgenden soll erklärt werden, was das im jeweiligen Fall für Begrenzungen in einer Kategorie bedeutet.

Ist die Liste einer Kategorie nicht in einer Policy enthalten, so bedeutet dies, dass der oder die Nutzer, je nach dem, wie lange die Proxy Kette ist, keine Begrenzungen für diesen Bereich festgelegt haben. Fehlt beispielsweise die Liste der möglichen Targets, so können Anfragen damit an alle Sites geschickt werden, zu denen der Aussteller des ursprünglichen Proxies Zugang hat. Ist er also durch die Ressourcen autorisiert, so schränkt die Policy die Wahl der Ziele nicht weiter ein.

Ist die Liste einer Kategorie in einer Policy vorhanden, und befinden sich Items darin, so hat der Nutzer, oder eine der Zwischenstationen, Beschränkungen für den jeweiligen Bereich festgelegt. Die Listen ermöglichen ein *Whitelisting* in den verschiedenen Kategorien. Es ist also immer genau das erlaubt, was angegeben wird. *Blacklisting* wäre grundsätzlich auch als Begrenzungsform denkbar, würde aber der Anforderung, delegierte Rechte exakt auf bestimmte Aufgaben beschränken zu können, nicht so sehr entsprechen wie das *Whitelisting*. Hier wird

eben nur genau das erlaubt, was tatsächlich benötigt wird. Ist in der Hosts-List als einziger FQDN beispiel.für.einen.fqdn.de angegeben, so darf das Proxy Zertifikat nur von diesem Objekt aus verwendet werden. Anfragen von anderen Maschinen werden abgelehnt. Für die Kategorien Targets, Hosts und Executables ist das Verhalten recht einfach ersichtlich. Die Action-List stellt eine kleine Ausnahme dar. Job-Submit beschreibt hier das Abschicken eines GRAM-Jobs und beinhaltet natürlich auch die Optionen *dryrun* und *authenticate-only*. Dies ist notwendig, weil es sonst bei der, im nächsten Abschnitt beschriebenen, Zusammenfassung von Policies zu Verlusten kommen kann. Erlaubt eine Policy das Abschicken von Jobs und die andere nur *dryrun* und *authenticate only*, so sollen bei der Anwendung beider Policies *dryrun* und *authenticate-only* möglich sein, und nicht keine der Optionen. Dieser Sachverhalt wird dann im nächsten Abschnitt, der die Zusammenfassung beschreibt, nochmal aufgegriffen. Ansonsten verhält sich die Action-List wie die anderen auch. Was angegeben ist, ist erlaubt, der Rest verboten.

Eine Liste kann Bestandteil einer Policy sein und keine Items enthalten. Dies kann passieren, wenn unterschiedliche Policies die möglichen Items so begrenzen, dass bei der Anwendung beider Policies kein Item mehr erlaubt ist. Wie das genau zustande kommt wird im nächsten Abschnitt genauer erläutert. Abhängig von der Kategorie, kann eine leere Liste ein Proxy Zertifikat unbrauchbar machen. Sind in einer Target-List keine Einträge, so ist keine weitere Anfrage möglich. In der Regel werden aber die Policies vom Aussteller eines ersten Proxys festgelegt, um die Verwendung seiner Rechte zu begrenzen und es kommt zu keinen Überschneidungen mit anderen Policies.

Listenkombination

In diesem Abschnitt geht es darum, wie mit mehreren durchzusetzenden Policies umgegangen wird. Sollen die Policies einfach nacheinander angewendet werden, oder ist es möglich sie geeignet, ohne Bedeutungsverlust zusammenzufassen? Durch die Definition der Policies als Listen mit Items für die verschiedenen Kategorien von Beschränkungen, und das Konzept des Whitelisting, ergibt sich tatsächlich ein einfaches Vorgehen für die Kombination von Regeln.

Jede Liste enthält die erlaubten Items. Die Listen, die Policies müssen aus der kompletten Policy-Kette gesammelt werden. Unter Umständen sind also für eine Kategorie mehrere Listen aus mehreren Policies vorhanden, die durchgesetzt werden müssen. Jede Liste verbietet dabei alles, was sie nicht ausdrücklich beinhaltet. Nur die Elemente, die in allen relevanten Listen enthalten sind, sind also erlaubt. Eine Zusammenfassung ist also durch das Bilden der Schnittmengen aus den vorhandenen Listen möglich. Liegen also die Policies aus einer kompletten Proxy-Kette vor, so kann daraus eine einzige Policy gebildet werden, die alle Beschränkungen beinhaltet. Die Policies können der Reihe nach durchgegangen werden, die Listen extrahiert, und jeweils die aktuelle Liste einer Kategorie mit der Gesamtliste, die aus vorherigen Schnitt-Operationen hervorging, geschnitten werden. Zu Beginn ist die Gesamtliste leer, die Liste aus der ersten Policy wird dann komplett übernommen. Dieses Vorgehen liefert ein richtiges Ergebnis, das alle angegebenen Begrenzungen widerspiegelt, weil für die Bildung von Schnittmengen das Kommutativgesetz gilt. Die Reihenfolge, in der einzelne Mengen geschnitten werden, ist für die daraus im Endeffekt resultierende Menge unerheblich.

Nach dem Durchlaufen aller Policies einer Kette, bleiben dann vier Listen, eine pro Kategorie, übrig, die die Semantik aller Policies vereinen. Mit diesen Listen kann dann bei der

Auswertung von Anfragen weitergearbeitet werden.

Listensyntax

Die Semantik des verwendeten Listenansatzes, wie auch die Kombination dieser wurden in den vorhergehenden Abschnitten beschrieben. Nun soll noch definiert werden, wie genau eine solche Liste aussehen soll. Es soll darauf geachtet werden, die Syntax für den Nutzer so einfach wie möglich zu halten. Es wird zwar ein Tool zur Erstellung von Begrenzungen geben, dass die Anwender unterstützen und die Arbeit mit beschränkter Delegation im Globus Toolkit 5 erleichtern soll. Trotzdem soll es auch möglich sein, ohne große Umstände, Policies selbst, ohne Unterstützung, zu erstellen.

Außerdem muss darauf geachtet werden, dass die angegebene Syntax auch großen Einfluss auf die Implementierung des Ansatzes hat. Die Regeln müssen auf Ressourcen-Seite bezüglich der einzelnen Kategorien aufgespalten und die einzelnen Items ermittelt werden. Dieser Vorgang soll durch die Syntax nicht unnötig kompliziert, beziehungsweise im besten Fall vereinfacht werden.

Die Listen für die verschiedenen Kategorien sollen eine einheitliche Syntax erhalten. Eine Liste wird von einem Paar von Geschweiften Klammern umfasst und beginnt mit der Angabe der Kategorie, für die die Begrenzungen definiert werden sollen. Es handelt sich dabei um eine der vier Kategorien *targets*, *hosts*, *executables* oder *actions*. Die Kategorie wird klein geschrieben und durch ein Semikolon abgeschlossen. Im Anschluss folgt die eigentlich Item-Liste. Jedes Item wird dabei wieder durch ein Semikolon abgeschlossen. Dies gilt auch für das letzte einer Kategorie. Ein Beispiel für eine solche Kategorie ist in Listing 5.1 zu sehen. Es handelt sich hier um die Kategorie *Hosts*, die die Items *test1.nm.ifi.lmu.de* und *test2.nm.ifi.lmu.de* enthält, und somit Anfragen von diesen beiden Maschinen aus erlaubt. Wie beschrieben sind alle Einträge, auch der letzte, mit einem Semikolon abgeschlossen.

```
1 {hosts;test1.nm.ifi.lmu.de;test2.nm.ifi.lmu.de;}
```

Listing 5.1: Liste für die Kategorie Hosts

Polisyntax

Nachdem festgelegt wurde, wie eine einzelne Liste auszusehen hat, geht es in diesem Abschnitt darum, wie eine komplette Policy definiert wird.

Eine Policy besteht aus einer Reihe von Kategoriellisten. Es ist dabei keine spezielle Reihenfolge vorgegeben. Durch die Angabe der jeweiligen Kategorie zu Beginn einer Liste ist dies auch nicht nötig. Außerdem ist es, wie bereits erwähnt, möglich, einzelne Kategorien wegzulassen. Eine nicht angegebene Liste bedeutet, dass in der jeweiligen Kategorie keine Einschränkungen vorgenommen wurden. Jede Kategorie darf nur einmal angegeben werden. Die Listen der verschiedenen Kategorien werden durch Kommata getrennt. Es ergibt sich also eine Aufzählung von ein bis vier Listen, abhängig von den gewünschten Beschränkungen. Die Policy wird in einer Textdatei gespeichert, um dann in einen Proxy integriert werden zu können. Leerzeichen und Zeilenumbrüche zwischen den Listen spielen dabei keine Rolle.

Ein Beispiel für eine Policy ist in Listing 5.2 zu sehen. Hier wurden Beschränkungen der Rechte in allen vier Kategorien vorgenommen. Es wird ermöglicht, von zwei bestimmten Maschinen aus, auf einer bestimmten Site Jobs auszuführen. */usr/bin/l*s und */usr/bin/l*ess sind dabei die einzigen zugelassenen ausführbaren Dateien.

```

1 {hosts;test1.nm.ifi.lmu.de;test2.nm.ifi.lmu.de;},
2 {targets;/O=Grid/OU=GlobusTest/OU=simpleCA-hp7kaiserfranz/CN=
   host/hp7kaiserfranz.fritz.box;},
3 {executables;/usr/bin/ls;/usr/bin/less;},
4 {actions;dryrun;authenticate-only;job-submit;stage-in}

```

Listing 5.2: Policy mit Beschränkungen in drei Kategorien

5.3 PDP - Entscheidungsalgorithmus

Der Algorithmus, mit dessen Hilfe Zugriffskontrollentscheidungen bezüglich aufgestellter Policies getroffen werden sollen, konnte nicht vor der Wahl der Policy-Sprache festgelegt werden. Diese bestimmt nämlich in großem Maße das Vorgehen dabei. Nun wurde die in Kapitel 5.2.3 beschriebene, auf einfachen Listen basierende Policy-Sprache ausgewählt. Darauf aufbauend wird nun das vorgehen des PDP auf dem Weg zu seinen Entscheidungen erarbeitet.

Die Anfrage-Beschreibung, die der PDP-Komponente zur Verfügung gestellt wird, enthält Informationen zu den vier Kategorien, die in Kapitel 5.2.2 festgelegt wurden. Angegeben ist also der Subject Name des Host-Zertifikats der angefragten Site, sowie den Full Qualified Domain Name der Maschine, von der aus die Anfrage geschickt wird. Außerdem erhält er, je nach Art der Anfrage, den absoluten Pfad zur Datei die ausgeführt werden soll und zusätzlich die benötigten Grid-Aktionen. Die Infos in den letzten beiden Kategorie wurden dabei aus der Job-Beschreibung in der Resource Specification Language, die zur Beschreibung von GRAM-Jobs verwendet wird, extrahiert. Eine für den PDP aufbereitete Anfrage besteht also auch aus einer Liste pro Kategorie. Ist eine der Listen leer, so benötigt die Anfrage keine Autorisierung in dieser Kategorie. Dies unterscheidet Anfragen von den Policy-Listen. Dort signalisiert eine leere Liste, dass in dieser Kategorie nichts erlaubt ist. In jeder Liste der Anfrage sind also die für die Erledigung der jeweiligen Aufgabe benötigten Items.

Durch die sehr einfach gehaltene Sprache zur Definition von Policies und der Möglichkeit diese zusammenzufassen, ist auch der Algorithmus zur Entscheidungsfindung, der im PDP umgesetzt wird ziemlich einfach und leicht zu erklären.

Es gibt verschiedene Möglichkeiten die Entscheidungen zu verschiedenen Regeln zu einer Gesamtentscheidung zu kombinieren. Einige der bekanntesten sind hier *permit-overrides*, *first-applicable*, sowie *deny-overrides*. Der erstgenannte prüft die Regeln nacheinander, ein *permit* setzt dabei alle anderen Entscheidungen außer Kraft und die Gesamtentscheidung ist *permit*, der Zugriff wird also gewährt. *first-applicable* hingegen sucht die erste anwendbare Regel und die resultierende Entscheidung wird zur Gesamtentscheidung aller zu kombinierender Regeln. *deny-overrides* verhält sich wie *permit-overrides*, nur dass hier ein *deny* alle anderen Entscheidungen überschreibt und das Gesamtergebnis *deny* lautet.

Im Fall des hier vorgestellten Ansatzes heißt dies, dass die Entscheidungen in allen vier Kategorien *permit* lauten müssen, um ein positives Ergebnis zu erhalten. Ein einziges *deny* reicht, um die Anfrage abzulehnen. Die in der Proxy-Kette transportierten Regeln, die festlegen, was genau mit einem Zertifikat gemacht werden darf, sind hier in einem Bereich verletzt und das reicht aus, um den Zugriff nicht zu gestatten.

Die Kategorien werden jetzt der Reihe nach durchgegangen und Entscheidungen in jedem Bereich getroffen. Sind alle Items aus der Anfrage-Beschreibung in einer Kategorie auch in der Kategorie-Liste der Policy enthalten, so lautet die Entscheidung *permit*. Ist ein Item

nicht vorhanden, so lautet das Ergebnis *deny*. Die Gesamtentscheidung ist positiv, falls alle Einzelentscheidungen ein *permit* liefern. Wird ein *deny* zurückgegeben, so lautet die Gesamtentscheidung ebenfalls 'deny'.

In Abbildung 5.7 ist ein Gesamtbeispiel für den Entscheidungsalgorithmus zu sehen. In drei der vier Kategorien lautet die Einzelentscheidung dabei *permit*, nur in der Kategorie *Executables* kommt es zu einem *deny*. Der angefragte Job beinhaltet die Ausführung des *grid-proxy-info* Tools zur Ausgabe von Informationen über das Zertifikat des aktuellen Sitzungsproxy. Allerdings erlauben die Begrenzungen, die im verwendeten Proxy enthalten sind, nur die Ausführung der Dateien */bin/ls*, */bin/less* und */usr/bin/who*. Die Einzelentscheidung ist in dieser Kategorie ein *deny*. Der *deny-overrides* Algorithmus greift und das Gesamtergebnis der Auswertung lautet *deny*, der Job wird also abgelehnt.

Kategorie	Anfrage	Kategorie-Liste	Entscheidung
Targets	/O=Grid/OU=GlobusTest/OU=simpleCA-hp7kaiserfranz/CN=host/debian6	/O=Grid/OU=GlobusTest/OU=simpleCA-hp7kaiserfranz/CN=host/debian6	<i>permit</i>
Hosts	test1.nm.ifi.lmu.de	test1.nm.ifi.lmu.de test2.nm.ifi.lmu.de	<i>permit</i>
Executables	GLOBUS_LOCATION/bin/grid-proxy-info	/bin/ls /bin/less /usr/bin/who	<i>deny</i>
Actions	job-submit	job-submit authenticate-only dryrun	<i>permit</i>
Gesamtentscheidung:			<i>deny</i>

Abbildung 5.7: Beispiel einer Zugriffskontrollentscheidung

5.4 Programmablaufpläne

Ein Programmablaufdiagramm (PAP), auch Flussdiagramm genannt, wird verwendet, um die Umsetzung eines Algorithmus in einer Anwendung grafisch darzustellen. Dabei werden aufeinander folgende Operationen zur Lösung einer Aufgabe beschrieben. Die Grafiken führen dabei einen Schritt näher zur eigentlichen Umsetzung und dienen als Vorbereitung zur Implementierung der Algorithmen.

Da der hier vorgestellte Ansatz einen bereits bestehenden erweitern soll, hat die Einarbeitung in die Konzepte und dann auch in den Code des bereits bestehenden Ansatzes einige Zeit in Anspruch genommen. Aus diesem Grund können im Rahmen dieser Arbeit nicht alle vorgestellten Komponenten tatsächlich implementiert werden. Das Vorgehen wurde bereits in den vorhergehenden Abschnitten erläutert, aber die tatsächliche Umsetzung in lauffähigen Code muss für einige Teile des neuen Systems von Folgearbeiten übernommen werden, die auf der hier gelegten Grundlage aufbauen können. Für die Komponenten, die umgesetzt werden können, sollen im Anschluss Programmablaufpläne erarbeitet werden, um diese dann bei der Implementierung, die in Kapitel 6 beschrieben wird, als Grundlage zu verwenden.

5.4.1 Methodik

In diesem Abschnitt wird gezeigt, aus welchen Elementen die verwendeten PAPs bestehen, und welche Aufgaben diese erfüllen. Ein Überblick ist in Abbildung 5.8 zu sehen. Das *Start* und *Stop* Element markieren Beginn, beziehungsweise Ende des dargestellten Lösungsweges. Die Parallelogramme werden genutzt um Ein- und Ausgaben darzustellen. Dabei kann jeweils eine Variable angegeben werden, die entweder den Input aufnimmt, oder den Wert der Variable ausgibt. Als Output ist auch ein String möglich. Dies wird beispielsweise verwendet, um eine Interaktion mit dem Nutzer zu zeigen. Die Raute stellt eine Verzweigung im Ablauf dar. Je nachdem, ob eine aufgestellte Bedingung erfüllt ist, oder nicht, werden unterschiedliche Flussrichtungen eingeschlagen. Das einfache Rechteck stellt eine vom System durchgeführte Operation dar, während dasjenige mit den doppelten vertikalen Seiten ein Unterprogramm repräsentiert, das an dieser Stelle aufgerufen wird. Dieses Unterprogramm kann dann wieder mittels eines PAPs gezeigt werden. Die Diagramme komplexer Abläufe können so in einfachere, kleinere aufgespalten werden. Verbunden werden die einzelnen Elemente jeweils durch Pfeile, die die Flussrichtung anzeigen.

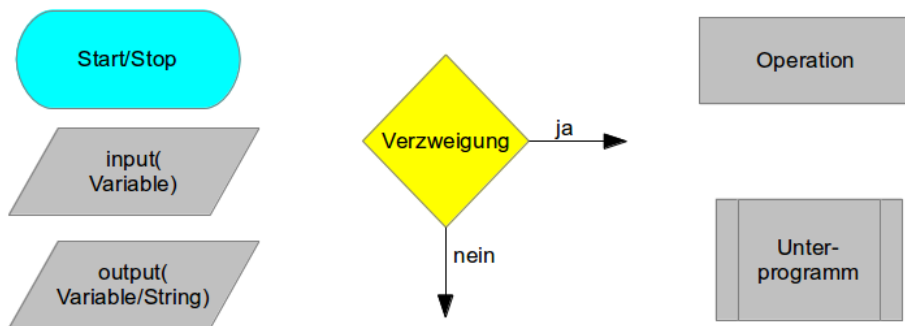


Abbildung 5.8: Flussdiagramm: Verwendete Elemente

5.4.2 Anwendung

Im Folgenden werden Programmablaufdiagramme zu denjenigen Komponenten gezeigt und beschrieben, die darauf aufbauend implementiert werden. Teilweise beinhalten die Grafiken Unterprogrammaufrufe, die dann wiederum in weiteren Diagrammen gezeigt werden.

Computergestützte Erstellung von Policies

In Kapitel 3.2 wurde eine Anforderung an das zu entwickelnde System gestellt, die die Möglichkeit der computergestützten Definition von Regeln beinhaltet. Der Ablauf eines Programms, das diese Anforderung erfüllen soll, ist in Abbildung 5.9 zu sehen. Ein Kommandozeilen-Tool, das mit dem Nutzer interagiert, wird dafür entworfen. Der Nutzer hat zu Beginn die Wahl zwischen der Beendigung des Tools und Festlegung von Begrenzungen für eine bestimmte Kategorie. Gibt er eine Kategorie an, so wird überprüft, ob es sich um eine der erlaubten handelt. Im Anschluss muss noch überprüft werden, ob bereits eine Liste zu dieser Kategorie vorhanden ist. Ist das nicht der Fall, so muss die Struktur dafür geschaffen werden

und dann der Nutzer aufgefordert, ein Item zu wählen, oder die Liste leer zu lassen, und damit in dieser Kategorie nichts zu erlauben. Soll die Liste leer bleiben, so wird die Liste abgeschlossen und es geht wieder mit der Auswahl einer Kategorie weiter. Soll ein Item hinzugefügt werden, so wird dieses vom Nutzer eingegeben und es wird überprüft, ob es der grundsätzlichen Form eines Items dieser Kategorie entspricht. Wenn ja, so wird es in die Liste eingetragen. Dann kann man wieder wählen, ob ein weiteres Item hinzugefügt werden soll, oder die Liste komplett ist. Auf diese Weise können Angaben zu den verschiedenen Kategorien getätigt werden. Die Reihenfolge der Kategorien ist dabei irrelevant und es können natürlich auch Listen komplett weggelassen werden. Dies ist genau das gewollte Verhalten, da dies einfach bedeutet, dass im betreffenden Bereich keine Begrenzungen festgelegt werden sollen.

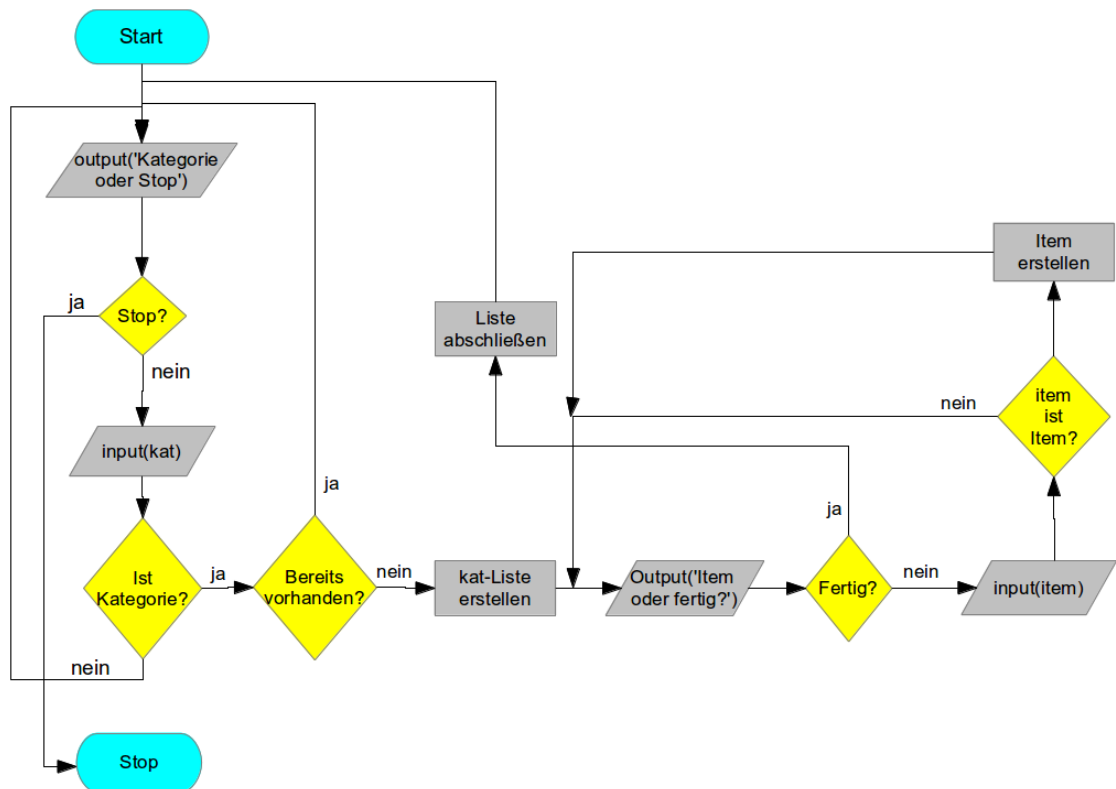


Abbildung 5.9: Flussdiagramm1 - Erstellung von Policies

Sammeln und Zusammenfassen von Policies

Wie bereits in Kapitel 5.1.2 erläutert wurde, ist für die Durchsetzung aller in einem Proxy enthaltenen Policies die gesamte Proxy-Kette nötig. Aus allen enthaltenen Zertifikaten, müssen die Policies geholt und in ein passendes Format gebracht werden. In 5.2.3 wurde außerdem gezeigt, dass mehrere Policies in der vorgeschlagenen Sprache relativ einfach zusammengefasst werden können. Der Ablauf von Sammlung und Zusammenfassung wird im Folgenden zuerst im Ganzen gezeigt und dann die in dieser Darstellung vorhandenen Unterprogramme noch genauer beleuchtet.

Wie bereits erwähnt, wird für die Authentifizierung des Clients im Gatekeeper die komplette Proxy-Kette benötigt. Dies hängt mit den Eigenschaften von Proxy Zertifikaten zusammen, welche nicht nur durch Certificate Authorities, sondern auch durch EECs und andere Proxies signiert werden können. Bei der Kommunikation zwischen Client und Gatekeeper wird also auch im bisherigen Ansatz schon die Kette komplett zum Server übertragen und dort als Stack von Proxies abgelegt. Dies wird in Kapitel 6 noch genauer beschrieben.

Der Stack mit allen Proxies der Kette ist also der vorhandene Input. Aus den enthaltenen Zertifikaten müssen dann jeweils die Policies geholt werden. Solange solche vorhanden sind, wird der Reihe nach jeweils eine ausgewählt, dann die enthaltenen Kategorie-Listen für die Beschränkungen extrahiert, und schließlich in jeder Kategorie die Liste mit der bereits vorhandenen Gesamtliste zusammengefasst. Auf die Kombination von Listen ist ja auch in 5.2.3 schon genauer eingegangen worden. Wurde das für alle vorhandenen Policies der Kette gemacht, so werden die Gesamtlisten für die jeweiligen Kategorien ausgegeben und der Ablauf ist beendet.

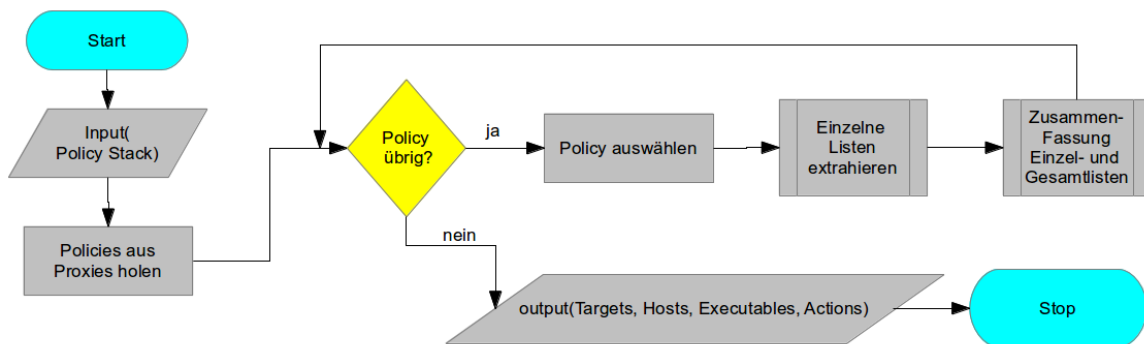


Abbildung 5.10: Flussdiagramm2 - Policies sammeln und zusammenfassen

In diesem Ablauf befinden sich zwei Unterprogrammaufrufe, die ebenfalls mit Hilfe von PAPs beschrieben werden. Das erste betrifft das Element *Einzelne Listen extrahieren*. Es wurde eine Policy ausgewählt, die als String vorliegt. Daraus müssen jetzt die Beschränkungen in den einzelnen Kategorien, also die jeweiligen Listen, geholt werden. Der Ablauf dafür ist in Abbildung 5.11 zu sehen. Als Eingabe dient hier also eine Policy, die als String vorliegt. Dieser muss nun geparkt werden, um die Listen zu bekommen. Solange weitere Kategorien vorhanden sind, werden diese der Reihe nach ausgewählt. Der Name wird eingelesen und überprüft, ob es sich um eine gültige Kategorie handelt. Ist dies der Fall, so wird eine Datenstruktur für die jeweilige Liste angelegt. Solange Items vorhanden sind, werden diese eingelesen und in die Datenstruktur eingetragen. Ist die Liste von Anfang an leer, oder sind alle Items abgearbeitet, so wird die Liste abgeschlossen und es geht mit dem Einlesen einer neuen Kategorie weiter. Ist keine mehr vorhanden, so werden die gewonnenen Listen ausgegeben und der Ablauf ist beendet.

Der Zweite Unterprogrammaufruf betrifft die *Zusammenfassung von Einzel- und Gesamtlisten*. Wie in 5.2.3 beschrieben, erlaubt die definierte Policy-Sprache das einfache Zusammenfassen von Regeln. Nachdem die Listen aus einer einzelnen Policy geholt wurden, können sie also mit der jeweiligen Gesamtliste vereint werden. Der Ablauf dazu ist in Abbildung 5.12 vorzufinden. Den Input stellt hier also jeweils eine Einzel- und die Gesamtlisten einer Kategorie dar.

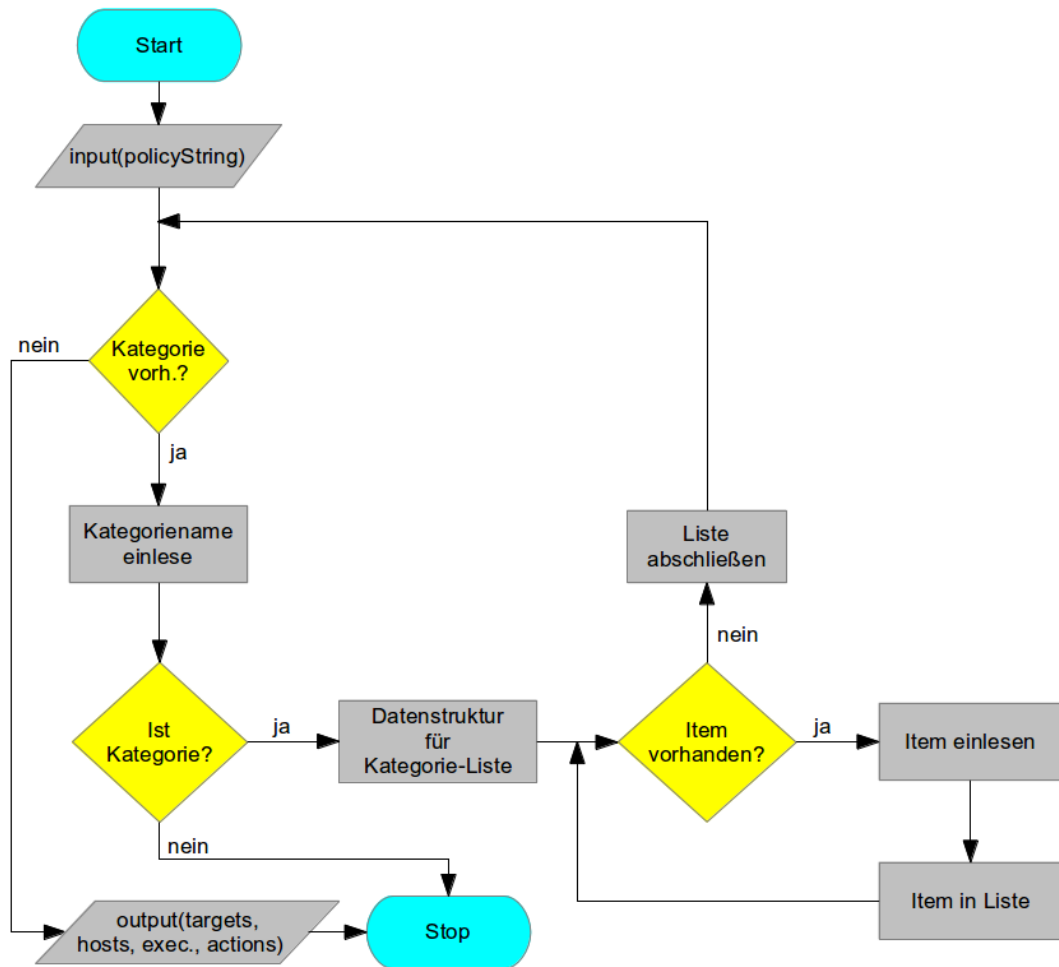


Abbildung 5.11: Flussdiagramm3 - Kategorie-Listen aus Policy extrahieren

Ist die Einzelliste nicht benutzt, das heißt es wurden vom Nutzer keine Einschränkungen in der betreffenden Kategorie festgelegt, so bleibt die Gesamtliste wie sie ist. Ist die Einzelliste hingegen leer, so wird auch eine leere Gesamtliste zurückgegeben, da die Einzelliste in der gegebenen Kategorie nichts zulässt. Ist das beides nicht der Fall, so befinden sich Items in der Einzelliste und die Gesamtliste muss weiter betrachtet werden. Ist sie unbenutzt, so wird die vorhandene Einzelliste zur neuen Gesamtliste. Ist die Gesamtliste jedoch leer, so bleibt sie das auch, da die Einzelliste hier keinen weiteren Einfluss hat. Es ist nichts erlaubt. Befinden sich Items sowohl in der Einzel-, als auch in der Gesamtliste, wird der Schnitt aus den Itemmengen der beiden Listen gebildet und dieser wird die neue Gesamtliste.

Die Schnittbildung wurde hier als Unterprogrammaufruf dargestellt und der dazugehörige Ablauf erhält ein eigenes Flussdiagramm, welches in Abbildung 5.13 zu sehen ist. Die Eingabe besteht aus einer Einzel- und einer Gesamtliste, die jeweils Items beinhalten. Zu Beginn muss eine geeignete Datenstruktur für die zu erstellende Schnittmenge erstellt werden. Solange dann Elemente in der Einzelliste sind, wird jeweils eines ausgewählt und mit allen Elementen der Gesamtliste verglichen. Bei Übereinstimmung wird das jeweilige Element in die Schnittmenge eingefügt.

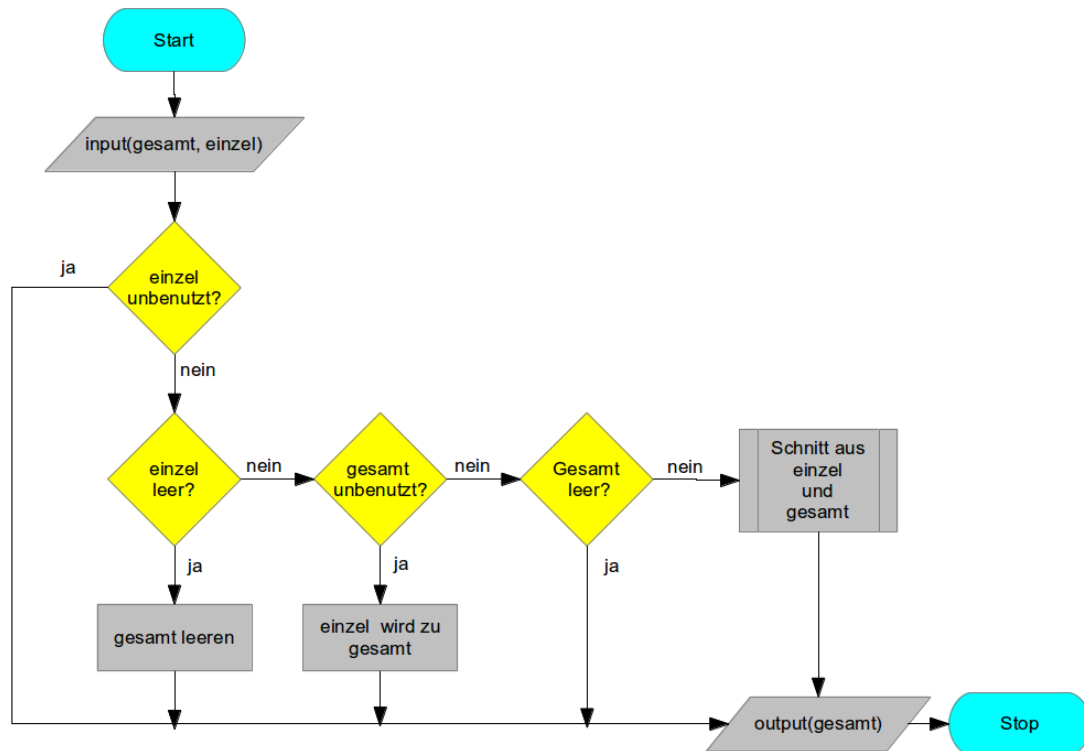


Abbildung 5.12: Flussdiagramm4 - Kombination von Einzel- mit Gesamtlisten

Wurden alle Elemente aus der Gesamtliste mit dem aktuellen Item aus der Einzelliste verglichen, so wird das nächste Element aus der Einzelliste gewählt und damit dasselbe gemacht. Sind auch in dieser Liste alle Items dran gewesen, so wird der Schnitt zur neuen Gesamtliste und diese wird ausgegeben. Damit endet der Ablauf dann auch. Aufbauend auf den erstellten Programmablaufdiagrammen, sollen die Komponenten nun im nächsten Kapitel umgesetzt werden.

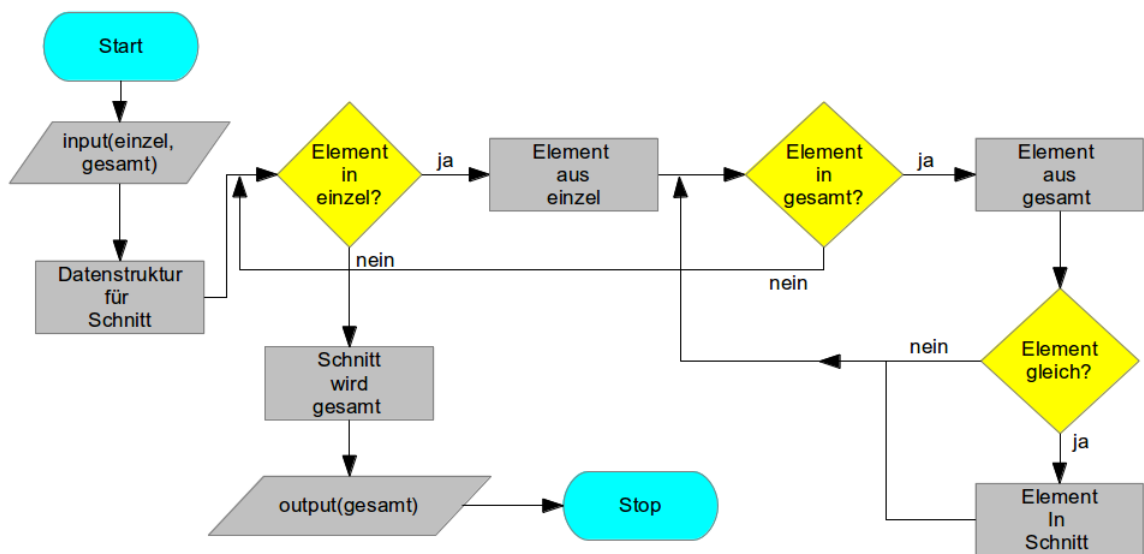


Abbildung 5.13: Flussdiagramm5 - Schnitt aus Einzel- und Gesamtliste bilden

6 Implementierung

In Kapitel 5.4 wurde bereits angesprochen, dass im Rahmen dieser Arbeit nicht alle Komponenten, die zur Realisierung der feingranularen Delegation von Nutzerrechten und der Spezialisierung auf bestimmte, eingegrenzte Aufgaben benötigt werden, implementiert werden können. In diesem Kapitel wird die Umgebung besprochen, in der die Implementierung stattfindet. Dies beinhaltet unter anderem Betriebssysteme und Programmiersprachen. Im Anschluss werden die tatsächlich umgesetzten und implementierten Komponenten beschrieben und Anmerkungen zu denjenigen Komponenten gemacht, die eben in dieser Arbeit nicht mehr umgesetzt wurden. Dies soll als Grundlage für Folgearbeiten dienen, die die komplette Umsetzung und Integration der Implementierung in den bereits bestehenden GT5-Code realisieren sollen.

6.1 Umgebung und verwendete Technologien

In diesem Abschnitt wird die Umgebung beschrieben, in der die Implementierung stattfindet, sowie dabei verwendete Technologien. In 6.1.1 geht es um die Version des Globus Toolkit, die zur Verwendung kommt. 6.1.2 beschäftigt sich mit dem Betriebssystem, auf dem die zu entwickelnden Komponenten laufen sollen und 6.1.3 bespricht die Wahl der Programmiersprache, die für die Implementierung verwendet wird.

6.1.1 Globus Toolkit 5

Das zu entwickelnde System soll eine Erweiterung des Globus Toolkit 5 darstellen, die dem bisherigen Globus-Ansatz Möglichkeiten zur Begrenzung delegierter Rechte auf bestimmte Aufgaben, oder einzelne Jobs, hinzufügt. Als Grundlage wurde hier der Code der Version 5.0.4 des Toolkits verwendet, welche zum Beginn der Bearbeitungszeit die aktuelle Version war. Sie wurde am 18 Mai 2011 veröffentlicht und es handelt sich um eine stabile Version. Die darauf folgende stabile Version 5.2 ist am 15. Dezember des selben Jahres erschienen. GT 5.0 ist dabei bezüglich Client-API und Protokoll kompatibel mit Version 5.2. Das heißt, 5.2 Clients können mit GT 5.0 Diensten zusammenarbeiten und umgekehrt. Die implementierten Komponenten könnten also auch in GT 5.2 integriert werden, ohne dass dabei komplexe Änderungen anfallen. Die Kompatibilität mit dem Globus Toolkit 4 kann hingegen nicht garantiert werden. Die Codebasis der Versionen ist hier grundsätzlich verschieden. Version 4 ist Web Service-basiert, Version 5 basiert auf dem Code der pre-ws Version 2 des Toolkits.

6.1.2 Betriebssystem

Als Betriebssystem kommen im Rahmen dieser Arbeit *Debian* Version 6 und *Ubuntu* in Version 11.10 zum Einsatz. Diese beiden Systeme wurden auch in Kapitel 1.2 verwendet, um zu zeigen, welche Gefahren von Proxy Zertifikaten ohne feingranulare Beschränkungen ausgehen. Für Debian 6 wurde am Lehrstuhl außerdem eine ausführliche Installationsanleitung

für Globus Toolkit 4 erstellt ([Str11]). Die Schritte in dieser Anleitung können größtenteils auch für die Installation und Konfiguration von GT5 übernommen werden. Da Ubuntu auf Debian basiert, entstehen auch hier kaum Konflikte mit der Anleitung.

Globus Toolkit 5 kann auf Ubuntu, Apple OS X, RedHat, Fedora Core, Debian, SuSE, FreeBSD und Solaris installiert werden. Eine Installation unter Windows ist nicht vorgesehen. Obwohl GT5 unter OS X und anderen Unix-basierten Betriebssystemen installiert werden kann, läuft der Großteil der GT Clients und Services doch auf den verschiedenen Linux Distributionen. Dies ist ein weiterer Grund für die Wahl von Ubuntu und Debian.

6.1.3 Programmiersprache

Tool zur Policy-Definition

In Anforderung A1.1 aus Kapitel 3.2 geht es um die computergestützte Erstellung von Begrenzungen für Proxy Zertifikate. Erfüllt werden soll diese Anforderung mit einem Kommandozeilen Tool, da dies der üblichen Nutzung des Globus Toolkit entspricht. Es werden keine GUIs angeboten, sondern die Hauptkomponenten, wie beispielsweise der GRAM Client, GSISSH, GridFTP, werden über die Konsole gesteuert.

Zur Programmierung des Tools wurde die Bash¹ verwendet. Dabei handelt es sich um eine freie Unix-Shell, die Teil des GNU Projekts² ist. Sie ist in vielen Unix-basierten Systemen enthalten und häufig auch die Standardshell. Damit sind sowohl OS X, als auch die verschiedenen Linux Distributionen und andere unixoide Systeme, abgedeckt. Außerdem spielte die eigene Erfahrung im Bereich Bash-Programmierung eine wichtige Rolle bei der Wahl.

Erweiterungen des bisherigen Ansatzes

Das zu entwickelnde System basiert auf Globus Toolkit 5 und soll dieses erweitern. GT5 wurde in der Programmiersprache C implementiert und Entwicklern wird eine C-API³ bereitgestellt. In Version 3 und 4 des Toolkits wurden ebenfalls Java-APIs zur Verfügung gestellt. Davor war das nicht der Fall und in Version 5, die auf dem Code von GT2 basiert, ist diese Option auch nicht vorhanden.

Die Wahl der Programmiersprache ist nun vom Weg abhängig, der bei der Erweiterung gegangen wird. Entweder werden die zu entwickelnden Komponenten getrennt vom bisherigen Code implementiert und an den passenden Stellen, beispielsweise durch Systemaufrufe, eingebunden, oder die Komponenten werden direkt in den bestehenden Code integriert. Die erste Variante erlaubt es, eine andere Sprache für die Implementierung zu wählen. Werden die Komponenten direkt integriert, so ist man an die bisher verwendete Sprache gebunden. Einige der für die feingranularen Begrenzungsmöglichkeiten nötigen Änderungen, wie zum Beispiel das Sammeln der Policies aus einer Proxy-Kette, sind eng verzahnt mit dem bisherigen Code, was eine komplette Trennung sehr schwierig macht. Außerdem bietet die C-API viele nützliche Funktionen, die für die Erweiterungen verwendet werden können und so nicht eigens implementiert werden müssen. So ist beispielsweise eine Funktion vorhanden, mit der Policies aus Proxy Zertifikaten geholt werden können und es gibt einen großen Teil der API, der die Resource Specification Language behandelt. Dies ist wichtig für die Komponente des Ansatzes, die die nötigen Informationen aus der Anfrage gewinnt und dem PDP bereitstellt.

¹<http://www.gnu.org/software/bash/>

²<http://www.gnu.org/>

³<http://www.globus.org/api/c-globus-5.0.3>

Da GRAM-Anfragen in RSL beschrieben sind, kann die API hier sehr großen Nutzen haben. Dies führt zur Entscheidung, die nötigen Erweiterungen direkt zu integrieren, und als Programmiersprache C zu wählen.

6.2 Implementierte Komponenten

Für die Komponenten des erarbeiteten Ansatzes zur Realisierung der feingranularen Delegation in GT5 Grids, die im Rahmen der begrenzten Zeit umgesetzt werden konnten, wurden in Kapitel 5.4 Flussdiagramme erstellt. Darauf aufbauend wurden die Komponenten implementiert. Verwendete Datenstrukturen, Techniken und andere wichtige Bestandteile des Codes sollen in diesem Abschnitt besprochen und erklärt werden und so ein Überblick über das Vorgehen bei der Implementierung ermöglicht werden.

6.2.1 Tool zur Policy-Definition

Das Tool soll dem Nutzer die Arbeit erleichtern und syntaktische Fehler vermeiden helfen. Die Erstellung der Regeln selbst wird dem Nutzer komplett abgenommen. Er muss lediglich angeben, in welchen Kategorien er Begrenzungen vornehmen will und welche Items darin erlaubt werden sollen. Damit wird dann das Whitelisting umgesetzt. Erstellt wird die Policy dann durch das Tool, welches die angeforderten Kategorie-Listen anlegt und dabei die syntaktischen Vorgaben aus 5.2.3 beachtet.

```
#####
# By using this script, a policy to restrict proxy certificates can be #
# created. There are four possible categories. Empty lists restrict #
# everything, non specified lists allow everything. #
#####

The Policy will be created in current directory.
You can choose a name, or hit enter and use the default name policy.txt.
Filename [policy.txt]:
File already exists. Do you want to overwrite it? [y/n]: y

+-----| POSSIBLE CATEGORIES |-----+
(1) Targets
(2) Hosts
(3) Executables
(4) Actions
(q) Quit

*: already specified.

Choose category: █
```

Abbildung 6.1: Erstellung - Startbildschirm

Der Ablauf hält sich dabei sehr strikt an den entworfenen Programmablaufplan in Abbildung 5.9. Zwei zentrale Punkte, die während des Erstellungsvorgangs häufiger durchlaufen werden, sind die Auswahl der zu begrenzenden Kategorie und die Maske zur Eingabe weiterer erlaubter Items für eine Kategorie.

In Abbildung 6.1 ist der Startbildschirm zu sehen. Nach einer kurzen Erläuterung zur Semantik der Listen kann der Name der zu erstellenden Policy-Datei gewählt, oder der Standardname verwendet werden. Im Hintergrund wird nun die Policy-Datei angelegt und im Anschluss

folgt die Wahl der Kategorie. Dies geschieht durch die Eingabe einer Zahl zwischen eins und vier. Bereits bearbeitete Listen werden im Menü speziell markiert.

Wählt man eine Kategorie aus, so kommt man zum zweiten zentralen Punkt des Tools, der Eingabe der Items. Im Falle von Targets, Hosts, und Executables erhält man die Möglichkeit, die Items frei einzugeben. Es wird überprüft, ob es sich um ein für die entsprechende Kategorie gültiges Format handelt und die Kategorieliste wird um das eingegebene Element erweitert. Wird jedoch die Kategorie Actions gewählt, so ist der Ablauf etwas anders. In diesem Fall ist keine freie Eingabe möglich, sondern der Nutzer bekommt wieder seine Optionen angezeigt. Realisiert wird diese Auswahl, wie auch die Wahl der Kategorie, mit Hilfe eines *Case Statements*.

Abbildung 6.2 zeigt den Auswahlbildschirm, nachdem bereits ein Item für die Actions-Liste gewählt wurde. Das Tool verhält sich hier genau so, wie es in 5.2.3 festgelegt wurde.

```

You added job-submit to category Actions. This includes authenticate-only and
dryrun as well. Now you can choose another icon or finish with s

+-----| POSSIBLE ACTIONS |-----+
(1) stage-in
(2) stage-out
(3) job-submit*
(4) authenticate-only*
(5) dryrun*
(s) finish selecting actions

*: already included.

Choose action 1-5, or finish with s: █

```

Abbildung 6.2: Erstellung - Actions-Liste

Wird die Aktion job-submit zur Actions-Liste hinzugefügt, so werden im Hintergrund auch die Items dryrun und authenticate-only angehängt. Damit wird der definierten Semantik Rechnung getragen. Der Nutzer wird durch eine Ausgabe auf diesen Vorgang hingewiesen und hat dann die Möglichkeit, weitere Icons auszuwählen. Ist er fertig, so wird die aktuelle Liste abgeschlossen und er kommt wieder zurück zum Kategorie-Auswahl-Bildschirm.

Als Endprodukt erhält der Nutzer im aktuellen Verzeichnis eine Policy-Datei mit dem von ihm gewählten Namen und den gewünschten Begrenzungen. Diese Datei kann nun verwendet werden, um sie mit dem *grid-proxy-init* Befehl bei der Erstellung eines Proxys zu übergeben. Abbildung 6.3 zeigt eine so erzeugte Policy. Darin wurden Begrenzungen für die Kategorien Hosts und Actions festgelegt.

```

#####
# This is a policy to restrict proxy certificates #
#####

{hosts;test1.nm.ifi.lmu.de;},{actions;job-submit;dryrun;authenticate-only;
stage-in;}

```

Abbildung 6.3: Erstellung - fertige Policy

6.2.2 Sammeln und Zusammenfassen von Policies

Dieser Abschnitt beschäftigt sich mit der Implementierung der Policy Retrieval Point Komponente des hier vorgestellten Ansatzes. Auf Ressourcen-Seite müssen die von Nutzern aufgestellten Beschränkungen gesammelt und aufbereitet werden, damit sie der PDP Komponente als Grundlage für Autorisierungsentscheidungen zur Verfügung gestellt werden können.

Der Client authentifiziert sich mit seinem Sitzungsproxy und schickt dabei dem Globus-Gatekeeper auf der Seite der Ressource die komplette Proxy-Kette, um diesem die Validierung des Zertifikats zu ermöglichen. Die übertragenen Proxy Zertifikate werden dabei in Form eines Stacks im Hauptspeicher abgelegt. Die Tatsache, dass die komplette Kette auch schon für die Authentifizierung des Clients benötigt wird, liefert dem PRP genau die Eingabe, die er benötigt. Er muss also nur noch die Kette durchgehen und jeweils die enthaltenen Policies extrahieren.

Im Globus-Gatekeeper wird die Funktion *doit()* aufgerufen. Innerhalb dieser Funktion findet die Authentifizierung statt. Sie wird mit dem Aufruf von *globus_gss_assist_accept_sec_context()* angestoßen. Dabei wird der Zeiger auf ein *context_handle* übergeben. Nach Beendigung der Funktion ist in diesem Handle die Zertifikats-Kette des Client-Zertifikats als Stack abgelegt. Hier ist also der Einstieg für die Komponente zum Sammeln aller Policies einer Kette.

Datenstruktur für Kategorie-Listen

Bevor aber die Policies gesammelt werden können, muss eine Datenstruktur vorhanden sein, die die Listenelemente aufnehmen kann und somit die Listen, die die Grundlage der spezifizierten Policy-Sprache sind, repräsentiert. Dafür wurden einfach verkettete Listen gewählt. Damit ist es sehr gut möglich, auf die variable Anzahl von Items zu reagieren. Für jedes zusätzliche Item wird einfach ein neues Listenelement angelegt und als Nachfolger des bisher letzten eingetragen. In Listing 6.1 ist die Struktur zu sehen, die verwendet wurde, um Listenelemente zu beschreiben.

```

1 // Struktur für einen Listenknoten
2 typedef struct policy_list_node_s
3 {
4     // Item das abgelget wird
5     char *          wert;
6     // Flags für Listenstatus
7     int            new;
8     int            not_used;
9     //Nachfolger in der Liste
10    struct policy_list_node_s *  next;
11
12 }policy_list_node_t;

```

Listing 6.1: Listen-Element für verkettete Listen

wert ist dabei ein Zeiger auf den String, der den tatsächlichen Inhalt des Elements enthält. Dort wird also jeweils ein Item einer Policy-Kategorie abgelegt. *new* und *not_used* sind Flags, die benötigt werden, um den aktuellen Status einer Liste zu setzen und abzufragen. Der Zeiger *next* verweist wieder auf einen *policy_list_node_s* und damit wird die Verkettung der Liste umgesetzt. Wird ein neuer Eintrag zur Liste hinzugefügt, so muss ein neuer Listenknoten

erstellt, mit Inhalt gefüllt und ein Zeiger darauf in der *next* Variable des bisher letzten Knoten der Liste abgelegt werden. So ist es einfach möglich, mit lediglich dem Zeiger auf den Startknoten als Voraussetzung, eine gesamte Liste, entlang all ihrer Einträge, zu durchlaufen.

Einlesen der Listen

Ein wichtiger Schritt beim Sammeln und Aufbereiten der Policies ist es, zuallererst die Policies aus den Zertifikaten zu gewinnen. Auch hier hält die C-API bereits eine Funktion bereit. In Listing 6.2 ist der nötige Aufruf zu sehen. Bei *handle* handelt es sich um ein Credential Handle, das verschiedenste Informationen zu einem Zertifikat und dem dazugehörigen privaten Schlüssel enthält. *policies* ist ein Zeiger auf einen Stack von Strings, auf dem nach dem Durchlaufen der Funktion die in der Kette enthaltenen Regeln liegen. Der Aufruf in Zeile fünf holt dann jeweils die oberste Policy, die als String vorliegt, vom Stack und der Zeiger *policy* verweist im Anschluss darauf.

```

1  /* Sammeln der Policies aus der im Handle enthaltenen Proxy-
   * Kette */
2  result = globus_gsi_cred_get_policies(handle, &policies);
3
4  /* Holen der obersten Policy vom Stack */
5  policy = sk\_pop(policies)

```

Listing 6.2: Policies aus der Proxy-Kette gewinnen

Liegen die Policies nun einzeln in Form von Strings vor, so kann mit der Analyse begonnen werden. Die angegebenen Listen, sowie auch die jeweiligen Bestandteile müssen dafür getrennt und in der dafür vorgesehenen Datenstruktur abgelegt werden. Dies erledigt die Funktion *globus_gsi_parse_policy* aus Listing 6.3. Als Eingabe bekommt sie einen Zeiger pro Kategorie, in der Beschränkungen möglich sind, sowie den Zeiger auf die zu untersuchende Policy und eine Variable, die nach der Ausführung Aufschluss über den momentanen Status gibt. Sind Fehler aufgetreten, so wird dies durch den *minor_status* angezeigt. Aufgrund der Definition der Struktur für Listenknoten, müssen hier ebenso Zeiger übergeben werden. Dies gewährleistet, dass sowohl die Startknoten, als auch die Nachfolger in der Liste jeweils als Zeiger vorliegen.

Die Funktion parst nun den angegeben String, gemäß der in Kapitel 5.2.3 definierten Policy-Sprache. Die dort vorgegebene Policy-Syntax wird Schritt für Schritt abgearbeitet. Zuerst wird die geprüft, um welche Kategorie es sich aktuell handelt und dem entsprechend dann Items in die jeweilige verkettete Liste gehängt. Enthält eine Liste keine Einträge, so kann das später an der *new* Variable des Listenkopfes abgelesen werden, wurde eine Liste in einer Policy überhaupt nicht verwendet, so ist dies an der *not_used* Variable zu erkennen.

Nach dem Durchlauf der Funktion sind die in der Policy vorhandenen Listen in den dafür vorgesehenen Datenstrukturen abgelegt. Ein Zugriff ist über den jeweiligen Listenkopf möglich. Dieser gibt auch Aufschluss darüber, ob eine Kategorie überhaupt angegeben wurde und ob Items darin enthalten sind, oder nicht.

```

1 /* Analyse einer Policy und Aufteilung in Kategorie-Listen */
2 int globus_gsi_parse_policy(
3     char *                policy,
4     policy_list_node_t *  first_host_list,
5     policy_list_node_t *  first_target_list,
6     policy_list_node_t *  first_executable_list,
7     policy_list_node_t *  first_action_list,
8     int *                 minor_status)

```

Listing 6.3: Policies aus der Proxy-Kette gewinnen

Zusammenfassen der Listen

Liegen die Kategorie-Listen in der gewünschten Form vor, so ist es möglich, die Listen aus mehreren Policies zusammenzufassen. Die Erklärung dafür wurde im Abschnitt **Listenkombination** des Kapitels 5.2.3 geliefert und der grundsätzlich Ablauf mit den verschiedenen möglichen Wegen ist in den dazu erstellten Flussdiagrammen zu erkennen (Abbildung 5.12, 5.13).

Um dies zu realisieren, wurde die Funktion *globus_gsi_intersect_policy_lists* implementiert (Listing 6.4). Als Eingabe erhält sie zwei Listen und den *minor_status*, der schon aus der davor erklärten Funktion bekannt ist und hier dieselbe Aufgabe erfüllt.

Diese Funktion wird verwendet, um die aktuelle Policy mit den bisher gesammelten Policies zu kombinieren. Dabei wird in jeder Kategorie die Liste aus der aktuellen Regel mit der Gesamtliste der Kategorie zusammengefasst. Dabei wird zuerst überprüft, ob eine der Listen leer oder unbenutzt ist. In diesen Spezialfällen ist die neue Gesamtliste sehr leicht zu bestimmen. Sind aber in beiden Listen Elemente enthalten, so wird der Schnitt aus den Item-Mengen gebildet und dieser stellt die neue Gesamtliste dar. Das Vorgehen muss sich auch hier an der Datenstruktur *Verkettete Liste* orientieren.

```

1 /* Kombination zweier Listen */
2 int globus_gsi_intersect_policy_lists(
3     policy_list_node_t *  complete_list,
4     policy_list_node_t *  first_list,
5     int *                 minor_status);

```

Listing 6.4: Policies aus der Proxy-Kette gewinnen

Die beschriebenen Funktionen zum Sammeln und aufbereiten der Policies, sowie zur Zusammenfassung von Regeln im Rahmen einer Kategorie, müssen nun noch kombiniert werden. Es soll eine Komponente entstehen, die eine Proxy-Kette als Eingabe erhält und daraus vier Listen mit den gesamten Policies in den verschiedenen Kategorien erzeugt. Diese Listen können dann vom Policy Decision Point verwendet werden, um Entscheidungen bezüglich eingegangener Anfragen zu fällen.

In einer Schleife werden dabei die Policies einzeln vom Stack geholt und mit der Funktion *globus_gsi_cred_get_policies* eingelesen sowie in einzelne Kategorielisten abgelegt. Es ist dann pro Kategorie, für die die Regel Begrenzungen enthält, eine Liste vorhanden. Diese Listen werden mit Hilfe der Funktion *globus_gsi_intersect_policy_lists* mit den Gesamtlisten der Kategorien kombiniert. Wurden alle Policies der Kette derart bearbeitet, so muss dies

auch noch mit der Policy des aktuellen Proxy Zertifikats passieren. Danach liegt für jede Kategorie, in der überhaupt Begrenzungen festgelegt wurden, genau eine Liste vor, die alle getätigten Begrenzungen widerspiegelt.

Genau dieses Verhalten zeigt die Funktion `globus_gsi_get_complete_proxy_policy`, welche in Listing 6.5 zu sehen ist. Als Eingabe werden Zeiger auf vier Listenköpfe, sowie ein Credential Handle und der Zeiger auf den `minor_status` übergeben. Die Listen repräsentieren die Gesamtlisten, die erzeugt werden sollen, das Handle enthält die Proxy-Kette, aus der die Policies geholt werden sollen, sowie das Zertifikat selbst. Der Status gibt Auskunft über Erfolg und Misserfolg der Ausführung.

```

1  /* Erstellung der Gesamtlisten */
2  globus_result_t globus_gsi_get_complete_proxy_policy(
3      policy_list_node_t *      first_complete_host_list,
4      policy_list_node_t *      first_complete_target_list,
5      policy_list_node_t *      first_complete_executable_list,
6      policy_list_node_t *      first_complete_action_list,
7      globus_gsi_cred_handle_t  handle,
8      int *                      minor_status)

```

Listing 6.5: Generierung einer Gesamtliste pro Kategorie

Tests

Das Tool zur computergestützten Erstellung von Policies in der in Kapitel 5.2.3 spezifizierten Sprache kann relativ unabhängig von der Umgebung getestet werden. Nötig ist dafür nur ein System mit einer Bash. Dies sind wie bereits erwähnt, die meisten unixoiden Systeme. Die Tests wurden auf einer *Ubuntu 11.10* Maschine durchgeführt.

Besonderes Augenmerk wurde dabei auf die wichtigsten Punkte der Ausführung gelegt. Dabei handelt es sich um die Wahl der Kategorie, sowie um das Eintragen von Items in bestimmte Kategorie-Listen. Hier ist speziell die Kategorie *Actions* zu nennen, da hier nur spezielle Items zugelassen sind und mit dem Eintrag `job-submit` automatisch weitere Einträge getätigt werden sollen. Der dritte und gleichzeitig wichtigste Punkt ist natürlich die Umsetzung der Nutzereingaben in eine syntaktisch korrekte Policy gemäß der vorgegebenen Policy-Sprache. Da diese mit dem Listenansatz sehr einfach gehalten wurde, kann leicht überprüft werden, ob das Tool korrekt reagiert. Zusätzlich wurde noch der Mechanismus kontrolliert, der bereits bearbeitete Kategorien und bereits eingetragene Items in die Actions-Liste markiert.

Alle durchgeführten Tests lieferten dabei die gewünschten Ergebnisse. Kategorien konnten gewählt und Items wie gewünscht eingetragen werden. Die Kennzeichnung bereits bearbeiteter Bereiche funktioniert und auch das Anlegen von leeren Listen läuft richtig ab. Wird eine Kategorie ausgewählt und dann kein Item eingetragen, so entsteht eine leere Liste.

Um die erstellten Policies weiter zu prüfen, wurden sie im Test der PRP Komponente als Ausgangspunkt genommen, um daraus die enthaltenen Einzellisten zu generieren und zusammenzufassen. Diese Tests werden im Folgenden genauer beschrieben.

Da das erarbeitete System zur feingranularen Delegation von Nutzerrechten in Globus Toolkit 5-basierten Grids im Rahmen dieser Arbeit nicht komplett, also mit allen dafür nötigen Komponenten, implementiert werden konnte, ist auch noch keine Integration vorgenommen worden, die ein Testen in einer solchen Umgebung sinnvoll machen würde.

Um die entwickelten Funktionen trotzdem zu testen, wurde ein Testprogramm entwickelt, das unabhängig vom Globus Toolkit ausgeführt werden kann.

Dabei werden aufeinander aufbauend zuerst die Funktion *globus_gsi_parse_policy* und dann die Funktion *globus_gsi_intersect_policy_lists* getestet. Der zweite Test nutzt dabei als Eingabe Listen, die beim Test des Parse-Vorgangs erstellt wurden. Getestet können damit natürlich auch Policies werden, die mit Hilfe des entwickelten Tools erstellt wurden. Dabei kann getestet werden, ob die vom Tool erzeugte Syntax von der Policy Retrieval Point auch korrekt verarbeitet werden kann. Außerdem werden auch Policies zum Test angeboten, die der Syntax nicht vollständig entsprechen. Man kann damit prüfen, ob leichte Abweichungen, wie das Vergessen des Kommas zwischen den einzelnen Listen, oder das Einfügen von Leerzeichen, von der Anwendung abgefangen werden und die Listen und Items trotzdem korrekt gebildet werden. Dies spielt dann natürlich nur eine Rolle, wenn die Policies nicht mit der angebotenen Unterstützung durch die Anwendung, sondern manuell erzeugt werden. Dies soll aber natürlich auch möglich sein. Außerdem können mit Hilfe von kleinen, im Programm dokumentierten Änderungen noch verschiedene Listenzustände für das Testen der Policy-Zusammenfassung simuliert werden.

Zu Beginn werden benötigte Variablen deklariert. Darunter die für die Tests nötigen *first_*_lists*, sowie eine *complete_host_list* und die erwähnten Test-Strings. Im Anschluss werden noch benötigte Variablen, wie beispielsweise die Flags für die verschiedenen Listen, initialisiert. Dann wird mit *globus_gsi_parse_policy* die aktuell aktivierte Policy geparkt und nach dem Durchlauf die Elemente der einzelnen Listen ausgegeben. Dabei wird zum Durchlauf der verketteten Listen der in Listing 6.6 gezeigte Algorithmus verwendet. Der Listenknoten *list* wird dabei als Zeiger auf die aktuelle Position verwendet und die Liste so durchwandert, ohne den Zeiger *first_target_list* zu verändern.

```

1 /* Ausgabe aller targets der target_list */
2 list = first_target_list;
3 while(list->next != NULL)
4 {
5     if(list->next->wert == NULL)
6         break;
7     puts(list->next->wert);
8     list = list->next;
9 }

```

Listing 6.6: Listenausgabe

Die *first_host_list*, eine der Listen, die beim Aufruf von *globus_gsi_parse_policy* erstellt wurden, wird nun als Eingabe für den zweiten Test benutzt. Dazu wird die *complete_host_list* noch mit Testwerten gefüllt und der Funktion ebenfalls übergeben. Die einzelnen Schritte beim Durchlauf werden, genau wie die neue Gesamtliste, am Bildschirm ausgegeben. Damit ist der Test dann beendet. Eine komplette Ausgabe einer Ausführung des Testprogramms ist in Listing 6.7 zu sehen. Der Mechanismus der Schnittbildung ist dabei genau zu erkennen. Es werden jeweils die beiden getesteten Items, sowie das Ergebnis der Auswertung angegeben. Dies kann mit der Policy und der gegebenen Gesamtliste, welche beide zu Beginn ausgegeben werden, verglichen werden.

```
1
2 PRP - Test
3
4 Given complete_host_list:
5
6 test1.nm.ifi.lmu.de
7 test3.nm.ifi.lmu.de
8
9 Given Policy-String:
10
11 #####
12 # This is a policy to restrict proxy certificates #
13 #####
14
15 {hosts;test1.nm.ifi.lmu.de;test2.nm.ifi.lmu.de;},{executables;/
16     usr/bin/test;},{actions;job-submit;dryrun;authenticate-only;
17     stage-in;},
18 -----
19 Start of Parsing:
20
21 host_list:
22 test1.nm.ifi.lmu.de
23 test2.nm.ifi.lmu.de
24
25 target_list:
26
27 action_list:
28 job-submit
29 dryrun
30 authenticate-only
31 stage-in
32
33 executable_list:
34 /usr/bin/test
35
36 End of Parsing
37
38 Start of intersection:
39 test1.nm.ifi.lmu.de
40 test1.nm.ifi.lmu.de
41 equal
42
43 test1.nm.ifi.lmu.de
44 test2.nm.ifi.lmu.de
```

```

45 unequal
46
47 test3.nm.ifi.lmu.de
48 test1.nm.ifi.lmu.de
49 unequal
50
51 test3.nm.ifi.lmu.de
52 test2.nm.ifi.lmu.de
53 unequal
54
55 End of intersection
56
57 New complete_host_list:
58 test1.nm.ifi.lmu.de
59
60 The End

```

Listing 6.7: Listenausgabe

Die möglichen Listenzustände, sowie die verschiedenen Fälle, die bei der Zusammenfassung der Listen auftreten, wurden damit getestet und die Ergebnisse waren durchwegs wie gewünscht und erwartet. Das Testprogramm ist dabei sehr flexibel änderbar, um die verschiedenen Fälle simulieren zu können. An den dabei interessanten Stellen im Programm sind Hinweise vorhanden, wie welche Konstellation erreicht werden kann.

6.3 Anmerkungen zu weiteren Komponenten

In diesem Abschnitt sollen Anmerkungen und Hinweise zu den Komponenten des Ansatzes gegeben werden, die aus den genannten Gründen nicht vollständig umgesetzt und implementiert wurden. Anhand dieser Informationen kann der Einstieg für die Verfasser von eventuellen Folgearbeiten erleichtert werden.

6.3.1 GRAM-Client und Kommunikation zum Gatekeeper

In diesem Zusammenhang sind nötige Änderungen am GRAM-Client zu nennen. In Kapitel 5.1.1 wurde die grundsätzliche Vorgehensweise erläutert und in Abbildung 5.2 zusammengefasst, wie die Mechanismen beim Abschicken von Jobs und der dabei fälligen Proxy-Delegation aussehen müssen. Um zu sehen, wie das Einbringen von Policies in PCs ermöglicht werden kann, sollten die vom *grid-proxy-init* Tool verwendeten Funktionen genauer betrachtet werden. Mit dem Befehl ist es bereits möglich, die Integration durchzuführen. Es ist auch wichtig zu verstehen, wie der GRAM-Client mit der entfernten GRAM-Schnittstelle kommuniziert, um Authentifizierung und Autorisierung zu realisieren. Dabei spielen zwei Funktionen eine entscheidende Rolle. Der Globus-Gatekeeper wartet mit dem Aufruf der Funktion *globus_gss_assist_accept_sec_context* auf Anfragen durch Clients. Im GRAM-Client wird beim Abschicken einer Anfrage die Funktion *globus_gss_assist_init_sec_context* aufgerufen. Über diese beiden Funktionen wird dann die Kommunikation abgewickelt, die Proxy-Kette, sowie die Zertifikate zur Authentifizierung geschickt, diese durchgeführt und die Delegation

eines Job-Proxies vollzogen. Die Funktionen sind in der C-API des Globus Toolkit im Bereich *Globus GSI GSS Assist*⁴ zu finden. Es handelt sich um Funktionen zur Nutzung der Globus GSS-API.

Die Akzeptanz von *restricted RFC 3820 compliant Proxies* beim Submit von Jobs an eine entfernte GRAM-Schnittstelle muss sichergestellt werden. Dafür sind Änderungen im Code notwendig.

6.3.2 Gatekeeper - Authentifizierung und Autorisierung

Die Komponenten PRP und PIP wurden in dieser Arbeit implementiert und könnten somit in GT5 integriert werden. Für die anderen Komponenten der in Abbildung 5.3 gezeigten Architektur wurden Theorie und Vorgehensweise festgelegt, die Implementierung war jedoch in der Kürze der Zeit nicht machbar.

Der Policy Enforcement Point des Ansatzes könnte in die *doit* Funktion integriert werden, welche vom Gatekeeper aufgerufen wird. Hier wird die Autorisierung durchgeführt, welche wie in Abbildung 5.5 zu sehen, um die Durchsetzung von durch Nutzer aufgestellten, feingranularen Regeln erweitert werden soll. Nach der Durchsetzung der Site-Policies, welche zum Beispiel durch Grid-Mapfiles repräsentiert werden können, soll also die Durchsetzung der Nutzerregeln stattfinden.

In dieser Arbeit wurde festgelegt, welches Format eine Anfrage haben muss, wenn sie dem PDP übergeben wird, und welche Informationen in welcher Form dazu nötig sind. Das Sammeln dieser Infos und die Bereitstellung müssen hier noch implementiert werden.

Der Policy Decision Point trifft die benötigten Zugriffskontrollentscheidungen und gibt diese an den PEP zurück. Der erarbeitete Ansatz sieht vor, dass der PDP als Eingabe die Kategorie-Listen aus den Policies, sowie die aus der Anfrage erhält. Dank der einfach gehaltenen Policy-Sprache ist der für die Entscheidung nötige Algorithmus wenig komplex und wurde im Kapitel 5.3 beschrieben. Er müsste also lediglich in Code umgesetzt werden.

⁴http://www.globus.org/api/c-globus-5.0.3/globus_gss_assist/html/index.html

7 Evaluation

In diesem Kapitel sollen nun der entwickelte Ansatz und die Implementierung anhand der in Kapitel 3.2 erstellten Anforderungen evaluiert werden. Als Grundlage dafür dient die in Abbildung 4.3 gezeigte Schablone, welche auch für die Bewertung der verwandten Arbeiten in Kapitel 4.1 verwendet wurde. Ansatz und Implementierung werden dabei getrennt voneinander betrachtet. Die Schablone dient als Überblick, die informellen Anforderungen, aus welchen der jeweilige Use Case abgeleitet wurde, sollen zur genaueren Einordnung ebenfalls beachtet werden. Zur Beurteilung der Use Cases wird für die Schablonen noch ein neues Icon eingeführt. Es handelt sich dabei um einen Haken mit einem Fragezeichen oben rechts. Wird es verwendet, so wird die Anforderung, beziehungsweise der Use Case, zwar grundsätzlich erfüllt, es sind aber noch Fragen zu klären. Dies kann auch bedeuten, dass eine komplette Erfüllung erst durch die Implementierung nachgewiesen werden kann.

7.1 Ansatz

Das in Kapitel 1.2 gezeigte Risiko bei der Verwendung von unbegrenzten Proxy Zertifikaten, sowie die Erkenntnisse aus dem Fortgeschrittenenpraktikum zum Thema *Sicherheit bei der Verwendung von Proxy Zertifikaten im Globus Toolkit*, führten zu den Anforderungen an einen Ansatz zur feingranularen Delegation von Nutzerrechten, welcher dann in Kapitel 5 erarbeitet und dargelegt wurde. Nun soll bewertet werden, inwieweit dieser theoretische Ansatz die gestellten Anforderungen erfüllt.

Die Tabelle in Abbildung 7.1 zeigt die Ergebnisse der Evaluation für die theoretische Ausarbeitung des Ansatzes. Wie auch schon bei der Evaluation der verwandten Arbeiten in Kapitel 4, wird auch hier tabellarisch gezeigt, ob die mit den jeweiligen Use Cases verbundenen Anforderungen vom System erfüllt werden, oder eben nicht.






	UC1	UC2	UC3	UC4	NFA1
Erweiterter Ansatz (Theorie)					

Abbildung 7.1: Erweiterter Ansatz (Theorie) - Evaluation

In Kapitel 5.1.1 wurde das Thema computergestützte Erstellung von Policies aufgegriffen, und erläutert, warum eine derartige Option wichtig für die sinnvolle Nutzung des feingranularen Delegationsansatzes ist. Grundlage für eine unterstützte Erstellung, wie auch für viele der folgenden Use Cases und Anforderungen, ist natürlich die verwendete Policy-Sprache. Syntax und Semantik wurden in Kapitel 5.2.3 eingeführt und definiert (A2.1/✓). Die Rechte können damit auf bestimmte Ziele (A2.2/✓), ausführbare Dateien (A2.3/✓) und bestimmte

GRAM-Aktionen (A2.4/✓) beschränkt werden. Um die Sicherheit noch weiter zu erhöhen, wird zusätzlich die Begrenzung der Quellen von Anfragen ermöglicht.

Da die GT-Nutzer außerdem mit der Kommandozeile vertraut sind, kann das Werkzeug zur Erstellung der Regeln in einem einfachen Skript realisiert werden. Der User gibt die Items an und das Tool kann sie einfach mitsamt der sehr einfachen Listenstruktur an eine Datei anhängen, welche später direkt verwendet werden kann, um die Regeln in Zertifikate einzubringen (UC1/✓).

Kapitel 5.1.1 erläutert ausführlich die Möglichkeit zur Integration von Beschränkungen in Sitzungsproxies unter Verwendung des *grid-proxy-init* Befehls und zeigt in Abbildung 5.1 ein auf diesem Wege erstelltes Zertifikat, mitsamt *Object Identifier* und gewünschten Beschränkungen (UC2/✓).

Das Abschicken eines GRAM-Jobs mit feingranularen Begrenzungen und die Umsetzung selbiger, wurde in den Abschnitten 5.1.1 und 5.1.2, sowie mit Hilfe von Abbildung 5.2 ausführlich besprochen. Dabei wird eine Erweiterung vorgeschlagen und beschrieben, die zusätzlichen Komfort bei der Nutzung des Ansatzes bringt. Es handelt sich dabei um das direkte Einbringen von Begrenzungen in delegierte Job-Proxies. Über die Begrenzungen im Sitzungsproxy ist dies auch so schon möglich, die Erweiterung erlaubt allerdings eine Trennung von Sitzungs- und Job-Proxy. Dafür nötige Änderungen wurden aufgezeigt und sind Teil der genannten Abbildung. Im Tool *globusrun* müssen Optionen für die Policy-Sprache (-pl), sowie die tatsächlichen Regeln (-policy) nach dem Vorbild des *grid-proxy-init* Befehls geschaffen werden (A3.2/✓).

Die erforderlichen Änderungen für das Abschicken von GRAM-Jobs mit begrenzten Sitzungsproxies wurden ebenfalls aufgezeigt und das dafür nötige Standardverhalten erläutert. Es muss anstatt eines *limited* Proxys standardmäßig ein *full Proxy* delegiert werden, um die Policies aus dem Sitzungsproxy entsprechend umzusetzen. Das Vorgehen ändert sich dabei abhängig davon, ob die Begrenzungen im aktuellen Proxy oder dessen Proxy-Kette enthalten sind. Die verwendeten *globusrun* Optionen wurden ebenfalls in die Unterscheidung mit einbezogen. Damit die Verwendung von restricted Proxies grundsätzlich möglich wird, müssen auch noch Änderungen auf Ressourcen-Seite im Gatekeeper getätigt werden. Die Kommunikation zwischen Client und Gatekeeper ist ein relativ komplexer Prozess und die Stellen, an denen die Änderungen genau stattfinden müssen, wurden nicht angegeben. Zusätzlich muss noch eine geeignete OID gewählt oder neu definiert werden. Bei der kompletten Umsetzung des Ansatzes durch eine Folgearbeit müssen hier also noch offene Fragen geklärt werden (A4.1, A4.2/✓?).

Die Voraussetzungen dafür, dass die Regeln auf Seiten der Ressource verstanden werden können, wurden im Kapitel pol geschaffen. Nach der Definition der Sprache selbst, wurde die einfache Kombination von Regeln erklärt. Das Whitelisting in Kombination mit der Listensyntax ermöglicht die Zusammenfassung zweier Policies über den Schnitt der einzelnen Kategorie-Listen (A5.1/✓).

Von der PEP Komponente wird erwartet, die Anfrage in ein passendes Format zu verwandeln und an den PDP weiterzugeben. Die benötigten Informationen und die Form, in der die Anfrage letztendlich vorliegen muss, wurden angegeben (refpdp). Für die Gewinnung der Informationen aus der RSL-Beschreibung wird auf die Globus RSL-API verwiesen, welche unter anderem Funktionen zum Parsen von Beschreibungen und zum Transformieren zwischen verschiedenen Formaten beinhaltet. Ein genaues Vorgehen zur Gewinnung der Informationen wurde allerdings nicht angegeben (A5.2/✓?).

Der PDP des Ansatzes muss Entscheidungen bezüglich Anfragen und vorhandener Policies

treffen. Der Algorithmus ist dabei stark abhängig von der gewählten Policy-Sprache und dem Format, in dem die Anfragen vorliegen. Der vorgestellte Ansatz beinhaltet einen sehr einfachen Entscheidungsalgorithmus (5.3). Items in einer Kategorie der Anfrage müssen auch in der entsprechenden Regel-Kategorie vorhanden sein. Es werden also vier Entscheidungen getroffen und diese per *permit overrides* kombiniert (A5.3/✓). Es wurden alle Anforderungen dieses Use Cases grundsätzlich erfüllt, an einigen Stellen sind allerdings vor der Umsetzung noch Fragen zu klären (UC3/✓?).

Der Ansatz sieht sowohl die Nutzung von full-, als auch die von limited Proxies weiterhin vor. Die beiden Optionen wurden auch bei der Behandlung der Delegation mittels *globus-run* beachtet. Liegen in einem verwendeten Zertifikat und der dazugehörigen Kette keine Begrenzungen vor, so soll beispielsweise auch weiterhin ein limited Proxy delegiert werden (UC4/✓).

Die in NFA1 geforderte Benutzbarkeit des Systems wird durch die einfach gehaltene Policy-Sprache garantiert. Sowohl das Einarbeiten in den vorgelegten Mechanismus, als auch die tatsächliche Definition von Regeln, zumal diese computergestützt ablaufen kann, stellen hier keine großen Herausforderungen dar (NFA1/✓).

7.2 Implementierung

Bei der Implementierung des erarbeiteten Ansatzes ist die Situation etwas anders. Die Ergebnistabelle für diese Auswertung ist in Abbildung 7.2 zu sehen.

	UC1	UC2	UC3	UC4	NFA1
Erweiterter Ansatz (Implement.)	✓	✓	✗	?	✓

Abbildung 7.2: Erweiterter Ansatz (Implementierung) - Evaluation

Ein Werkzeug zur computergestützten Erstellung von Begrenzungen entsprechend der Policy-Sprache des Ansatzes wurde implementiert und getestet (6.2.1). Der Nutzer gibt dabei lediglich die von ihm gewünschten Items in den gewünschten Kategorien an und die Policy wird automatisch erstellt und in einer Datei abgelegt (UC1/✓).

Die Erstellung von Sitzungsproxies mit integrierten Begrenzungen ist möglich. Das vorgestellte System wird in Globus Toolkit 5 integriert und bietet die dort bereits vorhandene Möglichkeit weiter an. Das zur Erstellung benutzte *grid-proxy-init* Tool muss zur Realisierung der feingranularen Delegation nicht verändert werden und behält seine Funktionalität vollständig (UC2/✓).

Die Anforderungen, die mit dem Use Case UC3 einhergehen, werden durch die Implementierung nicht realisiert. Lediglich das Einlesen und anschließende Zusammenfassen der Policies aus dem Zertifikat und der dazugehörigen Proxy-Kette wurden umgesetzt (6.2.2). Die Regeln werden aus der Policy und der Kette geholt, in die einzelnen Kategorien unterteilt und im Anschluss die Regeln in den Kategorien zu jeweils einer Liste zusammengefasst. Dies entspricht dem Format, in dem der PDP die Gesamtpolicy erhalten soll (A5.1/✓).

Weder die nötigen Änderungen am Client zur Verwendung von restricted PCs beim Submit von GRAM-Jobs, noch die Gewinnung der Informationen aus der Anfrage und die Autorisie-

rungsentscheidung selbst wurden implementiert (UC3/✗). Durch die theoretische Vorarbeit wurde jedoch eine solide Grundlage geschaffen, um diese Komponenten in einer Folgearbeit zu implementieren. Darauf wird dann im Ausblick der Arbeit (Kapitel 8) noch genauer eingegangen.

Da kein Prototyp implementiert wurde und es dadurch auch noch nicht zur Integration der neuen Funktionalität in den Bestehenden Ansatz von GT5 gekommen ist, kann über die Anforderungen aus UC4 keine Aussage gemacht werden. Die Verwendung von limited- und full Proxies ist in GT5 möglich, ob diese Optionen bei der endgültigen Implementierung erhalten bleiben, muss sich dann zeigen (UC4/?).

Die gestellten Anforderungen im Bezug auf die Benutzbarkeit des Systems, die in NFA1 festgelegt wurden, können hingegen als erfüllt betrachtet werden. Wie erwähnt sollte aufgrund der Einfachheit des erstellten Ansatzes die Einarbeitungszeit recht kurz ausfallen. Durch das Tool zur Erstellung der Regeln wird auch dieser Vorgang noch weiter verkürzt. Hat der Nutzer die Begrenzungen gewählt, so ist die gewünschte Policy in wenigen Minuten erstellt (NFA1/✓).

7.3 Fazit der Evaluation

Der in Kapitel 5 vorgestellte Ansatz zur feingranularen Beschränkung delegierter Rechte erfüllt die Anforderungen, welche in Kapitel 3.2 analysiert wurden, nahezu vollständig. Nur in einigen Punkten, wie der Gewinnung der Informationen aus RSL-Job-Beschreibungen, sind noch abschließende Fragen offen, welche für eine vollständige Umsetzung des Ansatzes beantwortet werden müssen. Sowohl die Definition der Policy-Sprache mitsamt Anfrageformat, als auch das Vorgehen bei der Umsetzung der beteiligten Komponenten wurden ausführlich beschrieben.

Die Implementierung eines Prototypen konnte aufgrund der begrenzten Zeit nicht umgesetzt werden. Einige Komponenten wurden allerdings bereits implementiert und die daran gestellten Anforderungen erfüllt. Durch die Einfachheit der angegebenen Policy-Sprache wurde eine gute Grundlage geschaffen, um Komponenten wie den Policy Decision Point, samt Entscheidungsalgorithmus, einfach umsetzen zu können.

Ein wichtiges Ziel dieser Arbeit war es, eine unkomplizierte Möglichkeit zur feingranularen Delegation in GT5 zu zeigen. Dies ist mit dem vorgestellten Ansatz gelungen, die endgültige Umsetzung kann in nachfolgenden Arbeiten erledigt werden.

8 Zusammenfassung und Ausblick

8.1 Zusammenfassung

In der vorliegenden Arbeit wurden die Ergebnisse eines Fortgeschrittenenpraktikums zum Thema *Sicherheit bei der Verwendung von Proxy Zertifikaten im Globus Toolkit* aufgegriffen ([Sch10]). Ein darin genanntes Sicherheitsproblem wurde aufgrund seiner Wichtigkeit ausgewählt, um eine Lösung dafür zu erarbeiten. Im Globus Toolkit fehlt die Möglichkeit, feingranulare Beschränkungen delegierter Rechte festzulegen. Die in GT zur dynamischen Delegation von Nutzerrechten verwendeten Proxy Zertifikate können zwar eingeschränkt werden, allerdings nur ziemlich grobkörnig. Die Rechte lassen sich nicht auf spezielle Aufgaben begrenzen. Welche Folgen das haben kann, wurde in Kapitel 1.2 gezeigt. Es sollte nun ein Ansatz für die aktuelle Version des Toolkits erarbeitet werden, der eben diese feingranularen Begrenzungen erlaubt und speziell auf bestimmte Jobs zugeschnittene Proxies ermöglicht. Die Anforderungen an einen Ansatz zur feingranularen Beschränkung delegierter Rechte im Globus Toolkit wurden in diesem Zusammenhang ausführlich analysiert (Kapitel 3) und darauf basierend eine Evaluation verwandter Ansätze durchgeführt. Da die Aufgabenstellung sehr speziell und auf die sehr aktuelle Version 5 des Toolkits begrenzt ist, gibt es dafür natürlich wenige vergleichbare Arbeiten und Ansätze. Es wurden also bekannte Grid Middlewares und speziell ihr Sicherheitskonzept, sowie die Umsetzung der dynamische Delegation evaluiert (Kapitel 4). Unter anderem wurden auch GT4 und GT5 untersucht, da es von besonderer Bedeutung war, wo hier die zu lösenden Probleme liegen. Es hat sich dabei gezeigt, dass die gestellten Anforderungen in all diesen Ansätzen nicht oder nur teilweise erfüllt sind. Im Advanced Resource Connector ist eine feingranulare Delegation umgesetzt, allerdings basiert diese Middleware auf grundlegend anderen Techniken als das Globus Toolkit und der Ansatz ist somit nicht übertragbar.

Aufgrund dieser gefundenen Fehlstellen in den untersuchten Ansätzen wurde in dieser Arbeit ein neuer entwickelt, der die analysierten Anforderungen erfüllen soll. Dabei wurde die in Kapitel 2.2.3 erläuterte Eigenschaft von Proxy Zertifikaten genutzt, die den Transport von Autorisierungsregeln in einer kritischen Erweiterung, der sogenannten *Proxy Certificate Information Extension*, ermöglicht. Der neue Ansatz, der eine Erweiterung des Globus Toolkits darstellt, sollte in erster Linie zeigen, dass feingranulare Beschränkungen bei der Verwendung von Proxy Zertifikaten in GT5 möglich sind und diese mit einfachen Mitteln umgesetzt werden können. Die für die Begrenzungen nötige Regelsprache, samt Anfrageformat und Entscheidungsalgorithmus, wurde entwickelt (Kapitel 5.2.3) und nötige Änderungen des bisherigen Ansatzes beschrieben (Kapitel 5.1). Mechanismen des bestehenden Ansatzes, welche für den neuen genutzt werden können, wurden mit einbezogen. So ist die für die Validierung der von den Nutzern verwendeten Zertifikate nötige Proxy Kette auch für den neuen Ansatz wichtig. Um die enthaltenen Regeln vollständig durchsetzen zu können, sind nicht nur die Regeln aus dem aktuellen Proxy, sondern eben auch die aus der gesamten Kette zu beachten. Diese wird auch bisher schon vom Client zur Site übertragen und kann dort also auch vom neuen Ansatz verwendet werden.

Der Ansatz verwendet also eine eigene Policy-Sprache, mit der Begrenzungen in PCs integriert werden können. Diese werden auf Ressourcen-Seite aus der gesamten Proxy-Kette extrahiert und passend zusammengefasst. Außerdem wird die Anfrage in das für den Entscheidungsalgorithmus passende Format gebracht und zusammen mit den Regeln dem PDP übergeben, welcher eine Autorisierungsentscheidung trifft. Dieser Mechanismus greift nach der Autorisierung anhand der Ressourcen-Begrenzungen. Es entsteht also ein zweischichtiger Ansatz. Zuerst werden die Autorisierungsregeln der Site, dann die des Nutzerzertifikats durchgesetzt. Fallen beide Entscheidungen positiv aus, so wird die Anfrage angenommen. Der vorgelegte Ansatz ermöglicht die Begrenzung delegierter Rechte, zugeschnitten auf spezielle Aufgaben. Es können Anfrage-Ziele, Anfrage-Quellen, ausführbare Dateien, sowie die möglichen GRAM-Aktionen beschränkt werden.

Ein Teil der für den Ansatz nötigen Komponenten wurde dann auf Basis des erarbeiteten Entwurfs auch noch implementiert (Kapitel 6). Die Entwicklung der weiteren Komponenten, sowie die endgültige Integration in GT5 müssen von Folgearbeiten erledigt werden.

8.2 Ausblick

Der in dieser Arbeit entwickelte Ansatz zur feingranularen Delegation von Nutzerrechten erfüllt bereits nahezu alle gestellten Anforderungen (Kapitel 7.1). Die dafür nötige tiefgreifende Analyse des bisherigen Ansatzes und die Erarbeitung des darauf aufbauenden neuen Ansatzes waren so aufwändig, dass eine vollständige Implementierung der nötigen Komponenten in der gegebenen Zeit nicht mehr bewerkstelligt werden konnte.

8.2.1 Vollständige Implementierung des Ansatzes

Die Fertigstellung der Implementierung sollte also in einer Folgearbeit durchgeführt werden. Die Grundlage dafür ist mit dem Entwurf aus Kapitel 5 gegeben. Die Policy-Sprache, samt Anfrageformat, Entscheidungsalgorithmus und Kombinationsalgorithmus, ist vorgegeben, Vorgehensweisen für die Umsetzung der noch fehlenden Komponenten wurden beschrieben. Das Werkzeug zur computergestützten Definition von Policies wurde ebenso bereits implementiert, wie der Mechanismus zum Sammeln und Zusammenfassen aller Regeln, welche für eine Autorisierungsentscheidung relevant sind.

Zu erledigen wären also noch die in Kapitel 5.1.1 beschriebenen Änderungen am *globusrun* Werkzeug, sowie an der Kommunikation zwischen Client und entferntem Gatekeeper bei der Verwendung von begrenzten Zertifikaten. Die Akzeptanz von restricted Proxies muss also implementiert werden. Die Gewinnung der nötigen Informationen aus der Anfrage, sowie die Darstellung im benötigten Format müssen ebenso noch in Code umgesetzt werden (PEP Komponente). Beide Bereiche wurden in Kapitel 5.1.2 beschrieben und in Kapitel 6.3.2 erneut aufgegriffen. Die letzte noch zu implementierende Komponente ist der Policy Decision Point. Dies sollte aber basierend auf dem in Kapitel 5.2.3 vorgelegten Kombinationsalgorithmus keine große Hürde darstellen.

Sind diese Ziele erreicht, so muss die vollständige Integration des Ansatzes in GT5 vollendet werden und eine ausgiebige Testphase folgen. Dann ist es auch möglich, eine vollständige Evaluation anhand aller Anforderungen durchzuführen. Die Umsetzung der feingranularen Delegation kann so in allen Bereichen des Globus Toolkit erreicht werden, welche eine Authentifizierung mit Proxy Zertifikaten voraussetzten.

8.2.2 Mögliche Erweiterungen

Ist diese Arbeit abgeschlossen, so kann über Erweiterungen des Ansatzes nachgedacht werden. Um zu zeigen, dass die geforderten Delegationsmechanismen in den Globus Toolkit 5-Ansatz integriert werden können, reichte es aus, den vorgestellten Ansatz auf die Ausführung von GRAM-Jobs zu begrenzen. Eine Erweiterung auf andere Komponenten, wie GridFTP und GSIssh ist aber durchaus denkbar und relativ einfach realisierbar. Die Verwendung von begrenzten Zertifikaten müsste auch hier erst ermöglicht werden. Die Komponenten des vorgestellten Ansatzes sind dafür ebenso geeignet, wie im Falle des Submits von GRAM-Jobs. Es werden auch hier Proxy Zertifikate zur Authentifizierung verwendet, welche die Regeln transportieren können. Der vorgestellte Entscheidungsalgorithmus könnte problemlos angepasst werden. Erweiterungen wären allerdings bei den möglichen Items für die Actions-List nötig. Hier könnten allerdings einfach Aktionen wie *gridftp_transfer* und *ssh_login* hinzugefügt werden. Für eine derartige Anfrage wäre wenig Information nötig. Der Client kontaktiert die Ressource ohnehin über den jeweiligen Port und somit ist klar, welche Aktion gewünscht ist. Der PDP müsste in diesem Fall nicht alle Kategorien prüfen, da beispielsweise ausführbare Dateien hier nicht relevant sind.

In allen Bereichen des Toolkits, die eine Authentifizierung mittels PCs voraussetzen, könnte also eine feingranulare Delegation von Nutzerrechten umgesetzt werden.

Abbildungsverzeichnis

1.1	Problembeschreibung	2
1.2	grid-proxy-info: Gestohlenes Zertifikat	4
1.3	Job-Submit mit gestohlenem Zertifikat	5
2.1	Globus Toolkit 5 - Komponenten	7
2.2	Grid Security Infrastructure	8
2.3	DFN-PKI	10
2.4	Wechselseitige Authentifizierung	11
2.5	XACML Framework (in vereinfachter Form)	14
3.1	Sicherheitskonzept GT (stark vereinfachte Form)	22
3.2	Darstellung eines Use Case	29
3.3	Darstellung des Systems	30
3.4	Darstellung eines Akteurs	30
3.5	System mit Akteuren und Use-Cases	31
4.1	Globus Toolkit 4 - Überblick	36
4.2	Unicore - Überblick	41
4.3	Evaluations-Schablone	42
4.4	Globus Toolkit 4/5 - Evaluation	43
4.5	gLite - Evaluation	43
4.6	ARC - Evaluation	44
4.7	Unicore - Evaluation	44
5.1	Policy mit OID und Policy	46
5.2	Delegationsmöglichkeiten für Job-Proxy	48
5.3	Komponenten des umzusetzenden Autorisierungsframeworks	49
5.4	Bob's Proxy-Kette	49
5.5	Sicherheitskonzept GT - Soll-Zustand (stark vereinfachte Form)	52
5.6	Kategorien und ihre Items	56
5.7	Beispiel einer Zugriffskontrollentscheidung	60
5.8	Flussdiagramm: Verwendete Elemente	61
5.9	Flussdiagramm1 - Erstellung von Policies	62
5.10	Flussdiagramm2 - Policies sammeln und zusammenfassen	63
5.11	Flussdiagramm3 - Kategorie-Listen aus Policy extrahieren	64
5.12	Flussdiagramm4 - Kombination von Einzel- mit Gesamtlisten	65
5.13	Flussdiagramm5 - Schnitt aus Einzel- und Gesamtliste bilden	66
6.1	Erstellung - Startbildschirm	69
6.2	Erstellung - Actions-Liste	70

Abbildungsverzeichnis

6.3	Erstellung - fertige Policy	70
7.1	Erweiterter Ansatz (Theorie) - Evaluation	79
7.2	Erweiterter Ansatz (Implementierung) - Evaluation	81

Literaturverzeichnis

- [All] ALLIANCE, GLOBUS: *GT 5.0.0 Component Guide to Public Interfaces: GRAM5, Chapter 2*. <http://globus.org/toolkit/docs/5.0/5.0.0/execution/gram5/pi/gram5PublicInterfacesGuide.pdf>, Zugriff am: 15.12.2011.
- [Coc01] COCKBURN, ALISTAIR: *Writing Effective Use Cases*. Addison-Wesley, Amsterdam, 2001.
- [HLP⁺] HOUSLEY, R., RSA LABORATORIES, W. POLK, NIST, W. FORD, VERISIGN, D. SOLO und CITIGROUP: *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. <http://www.ietf.org/rfc/rfc3280.txt>, Zugriff am: 25.11.2011.
- [Led10] LEDERER, HERMANN: *DEISA Extreme Computing*. http://www.deisa.eu/news_press/symposium/barcelona2010/presentations/8_DPS-2010-DECI-HermannLederer.pdf, 2010.
- [QK] QIANG, WEIZHONG und ALEKSANDR KONSTANTINOV: *SECURITY FRAMEWORK OF ARC - Documentation and developer's guide*. <http://www.nordugrid.org/documents/arc-security-documentation.pdf>, Zugriff am: 15.12.2011.
- [Rup07] RUPP, CHRIS: *Requirements-Engineering und -Management: Professionelle, iterative Anforderungsanalyse für die Praxis. 4. Aufl.* Hanser, München, 2007.
- [Sch10] SCHULZ, CHRISTIAN: *Beurteilung der Sicherheit von Proxy Zertifikaten im Globus Toolkit*. <http://www.nm.ifi.lmu.de/pub/Fopras/schu10/PDF-Version/schu10.pdf>, 2010.
- [Str11] STRAUBE, CHRISTIAN: *Installationsanleitung Globus Toolkit 4.0.8. Lehr- und Forschungseinheit für Kommunikationssysteme und Systemprogrammierung*. 2011.
- [TWE⁺] TUECKE, S., V. WELCH, D. ENGERT, L. PEARLMAN und M. THOMPSON: *Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile*. <http://www.ietf.org/rfc/rfc3820.txt>, Zugriff am: 28.07.2011.
- [Waa] WAANANEN, ANDERS: *ARC Policy XML Schema*. <http://svn.nordugrid.org/trac/nordugrid/browser/arc1/trunk/src/hed/shc/arcdpd/Policy.xsd>, Zugriff am: 15.12.2011.