



# Diplomarbeit

## Entwicklung einer webbasierten Gruppenverwaltung

Michael Holtermann  
Matrikel-Nummer 4201726

11. Juli 2005

Fachhochschule Oldenburg/Ostfriesland/Wilhelmshaven  
Fachbereich Technik, Abteilung Elektrotechnik & Informatik  
und  
Stiftung Alfred-Wegener-Institut  
für Polar- und Meeresforschung Bremerhaven  
Rechenzentrum

Erstprüfer: Prof. Dr. Gert Veltink  
Zweitprüfer: Dipl.-Ing. Siegfried Makedanz

Copyright © 2005 Michael Holtermann. All rights reserved. Alle Rechte vorbehalten.  
Diese Arbeit unterliegt dem Urheberrecht. Sie darf innerhalb der Fachhochschule Oldenburg/Ostfriesland/Wilhelmshaven und der Stiftung Alfred-Wegener-Institut für Polar- und Meeresforschung für Studium, Forschung und Lehre frei verbreitet werden. Jede weitere Nutzung bedarf der Zustimmung des Rechteinhabers.  
Für die erstellte Software *eGroup*, inkl. der in dieser Arbeit abgedruckten Quelltexte:  
Copyright © 2005 Stiftung Alfred-Wegener-Institut für Polar- und Meeresforschung, Bremerhaven. All rights reserved. Alle Rechte vorbehalten. Für die beschriebenen Verfahren wird keine Haftung übernommen, ausgenommen bei Vorsatz und grober Fahrlässigkeit.  
Alle Warennamen werden ohne Gewährleistung der freien Verwendbarkeit benutzt und sind möglicherweise eingetragene Warenzeichen.

# Inhaltsverzeichnis

<b>1. Einführung</b>	<b>5</b>
1.1. Gliederung dieser Arbeit	5
1.2. Begriffsdefinitionen	6
1.3. Lastenheft	6
<b>2. Kontext</b>	<b>8</b>
2.1. Die Arbeit des Alfred-Wegener-Instituts	8
2.2. Forschungsnetzwerke im Vergleich	9
2.2.1. Deutsches Forschungsnetzwerk – DFN	11
2.2.2. The Swiss Education & Research Network – SWITCH	12
2.2.3. Trans-European Research and Education Networking Association – TERENA	14
2.2.4. Internet2	15
2.3. Einbindung des AWI in das Deutsche Forschungsnetzwerk	16
2.4. Verwaltung von Organisationen in Directories	17
2.5. Selbstverwaltung von Gruppen	20
<b>3. Implementierung</b>	<b>23</b>
3.1. Anforderungen an die Anwendung	23
3.2. Gewählte Technologien	26
3.2.1. Apple WebObjects	26
3.2.2. WebObjects Eclipse Plugin – WOLips	30
3.2.3. Persistenzschicht	30
3.2.4. JUnit	32
3.2.5. Logging mit Log4J	33
3.3. Modell	33
3.3.1. Abbildung der Organisationsstruktur	34
3.3.2. Anwendungsfälle in der Gruppenbildung	36
3.3.3. Datenmodell	42
3.4. Systembeschreibung	47
3.4.1. Packages	47
3.4.2. Ausgewählte Klassen	51
3.4.3. Einschränkungen	60
<b>4. Der Abschluss</b>	<b>61</b>

<b>A. Organisationsstruktur AWI</b>	<b>62</b>
<b>B. Organisationsstruktur MARCOPOLI</b>	<b>63</b>
<b>C. Quellcodes</b>	<b>64</b>
C.1. Application	64
C.1.1. Java-Klasse	64
C.1.2. UTF-8-Deklaration im HTML-Template	65
C.2. ApplicationDisabled	65
C.3. AwiLdapEntry	66
C.4. AwiPersonDisplayNameComponent	66
C.4.1. HTML-Template	66
C.4.2. Binding	66
C.4.3. Java-Klasse	66
C.5. Config	68
C.6. DirectAction	70
C.7. FunctionalAwiPersonsArrayComponent	71
C.7.1. Binding	71
C.7.2. Java-Klasse	72
C.8. InvitePeoplePage	73
C.8.1. HTML-Template	73
C.8.2. Binding	75
C.8.3. Java-Klasse	77
C.9. IsGroupOpenGroupCondComponent	79
C.9.1. HTML-Template	79
C.9.2. Binding	79
C.9.3. Java-Klasse	80
C.10. JNDIAccess	81
C.11. ModelAccess	87
C.12. SerialNumberHelper	92
C.13. Session	93
C.14. Utilities	100
<b>D. Glossar</b>	<b>111</b>
<b>Abbildungsverzeichnis</b>	<b>115</b>
<b>Listings</b>	<b>116</b>
<b>Literaturverzeichnis</b>	<b>117</b>

# 1. Einführung

Beisammen sind wir, fangt an!

*(Johann Wolfgang von Goethe)*

Die Stiftung Alfred-Wegener-Institut gilt als eines der renommiertesten Institute in der Polar- und Meeresforschung. Hier werden Daten gesammelt und analysiert, die das Ökosystem unseres Planeten beschreiben und damit auch direkt unser tägliches Leben betreffen. Neue Konzepte, Strategien und Technologien entstehen, um den selbstgestellten Ansprüchen gerecht zu werden.

In diesem Rahmen soll ein Softwaretool entwickelt werden, mit dem sich Arbeitsgruppen komfortabel verwalten lassen. Diese Arbeit bietet die Chance, direkten Einblick in einen ausgesprochen interessanten Forschungsbereich zu erhalten. Hier besteht die Gelegenheit, zumindest einen kleinen Beitrag zur einfacheren Zusammenarbeit unterschiedlichster Fachbereiche beizusteuern.

Die Erfolgsgeschichte des Internet wird zu einem guten Teil in den großen Forschungsnetzwerken überall auf dem Globus fortgeschrieben. Neue Techniken, neue Protokolle, neue Applikationen, neue Dienste und damit eine unüberschaubare Anzahl von Möglichkeiten werden zuerst in diesen Netzwerken erdacht, eingerichtet, getestet und schließlich in kommerzielle Netze übernommen. Diese Arbeit wird sich auch mit diesen Netzwerken auseinandersetzen und einen Einblick in deren Arbeit geben.

Mit dieser Arbeit werden einige interessante Technologien erschlossen, darunter die Datenhaltung in einem Verzeichnisdienst und die Entwicklung mit Apple WebObjects.

## 1.1. Gliederung dieser Arbeit

Diese Arbeit ist, neben dieser Einführung und einer abschließenden Beurteilung, in zwei große Abschnitte untergliedert.

Der erste Teil beschäftigt sich im Kapitel „**Kontext**“ mit den Rahmenbedingungen, vor denen der Wunsch nach einer Gruppenverwaltung besteht. Das Alfred-Wegener-Institut ist vielfältig in nationale und internationale Kooperationen eingebunden und tritt dabei sowohl als Verwalter als auch als Mitglied Virtueller Organisationen (VO) auf. Angesichts aktuell in der Entwicklung befindlicher Lösungen zur Unterstützung von Forschung und Entwicklung wird sich dieses Kapitel zu Forschungsnetzwerken und der Verwaltung von realen und Virtuellen Organisationen äußern.

Der zweite Teil „**Implementierung**“ geht auf die konkrete Umsetzung der Wünsche des AWI ein. Es beschreibt das für die Entwicklung verwendete Vorgehensmodell, die eingesetzten Technologien und Rahmenwerke sowie die Modelle, die hinter der Anwendung stehen. Es bietet darüber hinaus konkrete Hinweise, an welchen Punkten Erweiterungen möglich sind. Der wesentlichste Teil ist die Implementierung selbst.

Ein vollständiges Literaturverzeichnis und Quellcode-Auszüge finden sich im Anhang dieser Arbeit.

## 1.2. Begriffsdefinitionen

Das Thema und Umfeld der Arbeit bedingen einige Fachbegriffe, die zunächst kurz erläutert werden sollen.

Eine *Gruppe* ist ein Zusammenschluss mehrerer Personen. Sie kann aufgrund offizieller Vorgaben konstituiert sein, z.B. durch die vorgegebene Aufbauorganisation des Instituts. Sie kann aber auch ein loser Zusammenschluss interessierter Mitarbeiter sein.

Ein *Verzeichnisdienst* oder auch *Directory* ist ein Netzwerkdienst, der Informationen in einer Baumstruktur vorhält und über das Lightweight Directory Access Protocol<sup>1</sup> abgefragt werden kann.

Eine *Institution* oder auch *Organisation* ist die Gesamtheit aller Organisationseinheiten. Sie selbst wird durch einen Eintrag im Verzeichnisdienst repräsentiert, der an der Spitze der Aufbauorganisation steht.

Eine *Organisationseinheit* ist die Repräsentation einer Gruppe im Verzeichnisdienst. Sie ist durch mehrere Attribute gekennzeichnet und ist ein Teil der Institution.

Mit *Anwendung* oder *Applikation* werden Softwarelösungen bezeichnet, die im Intranet des Institutes zum Einsatz kommen.

Typografisch wird zwischen LDAP-Klassen, z.B. `eduPerson`, LDAP-Attributen, z.B. `awiGroupHead` und Java-Code, z.B. `public boolean getTrue()`; unterschieden.

## 1.3. Lastenheft

Das Alfred-Wegener-Institut hat folgendes Lastenheft ausgegeben, das Grundlage der vorgestellten Entwicklung ist.

„Es sind Prozesse zu definieren, nach denen eine Gruppenmitgliedschaft entsteht und wieder aufgehoben werden kann. Es sind mehrere Möglichkeiten denkbar, in eine Gruppe zu gelangen: Zwangseinschreibung, Selbsteinschreibung, Selbsteinschreibung mit Autorisierung.“

---

<sup>1</sup>LDAP [48], Protokoll zur Abfrage von Verzeichnisdiensten

Auch müssen verschiedene Gruppenarten abgebildet werden, wie Stabstellen, Fachbereiche, Sektionen, informelle Gruppen usw. Hier ist es notwendig, die Begrifflichkeiten klar zu definieren, auch deren englische Entsprechungen.

Neben der Selbstverwaltung der Gruppen soll auch die Verwaltung der Gruppen als Directory-Einträge durch einen oder mehrere Administratoren möglich sein.

Da im Directory bereits die Gruppenshierarchie erfasst wird, ist ein Konzept für die Ablage der Informationen nicht notwendig.

Im Hinblick auf zukünftige inter-institutionelle Partnerschaften muss dabei dafür gesorgt werden, dass die Mitgliedschaft in einer Gruppe autorisiert wird. Diese Autorisierung kann dabei vom Gruppenleiter an andere Personen delegiert werden.

Ein wichtiger Aspekt der zu erstellenden Anwendung ist, das Rechenzentrum von Routineaufgaben zu entlasten. Die Mitarbeiter des Alfred-Wegener-Institutes sollen in der Lage sein, komfortabel selbst Gruppen zu verwalten und ihre Mitgliedschaften in diesen Gruppen zu pflegen.

Die resultierende Anwendung ist mit Apple WebObjects zu programmieren.“

Der Umsetzung dieser Anforderungsbeschreibung widmet sich im Detail das Kapitel 3 – „Implementierung“.

## 2. Kontext

Ernst zu nehmende Forschung erkennt man daran, dass plötzlich zwei Probleme existieren, wo es vorher nur eines gegeben hat.

---

(Thorstein Bunde Veblen)

### 2.1. Die Arbeit des Alfred-Wegener-Instituts

„Wir leisten Beiträge zur Lösung großer und drängender Fragen von Gesellschaft, Wissenschaft und Wirtschaft durch strategisch-programmatisch ausgerichtete Spitzenforschung in den Bereichen Energie, Erde und Umwelt, Gesundheit, Schlüsseltechnologien, Struktur und Materie, Verkehr und Weltraum.

Wir erforschen Systeme hoher Komplexität unter Einsatz von Großgeräten und wissenschaftlichen Infrastrukturen gemeinsam mit nationalen und internationalen Partnern.

Wir tragen bei zur Gestaltung unserer Zukunft durch Verbindung von Forschung und Technologieentwicklung mit innovativen Anwendungs- und Vorsorgeperspektiven.“

Das ist die Mission der Helmholtz-Gemeinschaft Deutscher Forschungszentren, der auch das Alfred-Wegener-Institut angehört. Die gesamte Forschung, die in Bremerhaven betrieben wird, hat sich in den Dienst dieser Mission gestellt.

Die *Stiftung Alfred-Wegener-Institut für Polar- und Meeresforschung in der Helmholtz-Gemeinschaft*, so der vollständige Name des in Bremerhaven beheimateten Forschungsinstituts, beschäftigt sich mit allen Fragen rund um die Themen Meer und Polargebiete. Sie ist eingebunden in ein Netzwerk nationaler und internationaler Partner, die sich mit der Erforschung des Erd- und Umweltsystems beschäftigen.

Zu diesem Zweck unterhält das Alfred-Wegener-Institut, kurz *AWI*, mehrere Standorte innerhalb Deutschlands (Bremerhaven, Potsdam, List auf Sylt und Helgoland), sowie Forschungsstationen in den Polargebieten der Erde: die Neumayer-Station, die Kohlen-Station und das Dallmann-Labor in der Antarktis, sowie die Koldewey-Station auf Spitzbergen in der Arktis. Eine weitere Station in der Antarktis, Filchner, musste 1999 vom Eis geborgen werden, nachdem sich die Scholle, auf der die Station installiert war, vom Festlandeis gelöst hatte und durch das Weddelmeer driftete.

## 2.2. Forschungsnetzwerke im Vergleich

Neben dem Unterhalt dieser Stationen betreibt das AWI mehrere Flugzeuge und Forschungsschiffe, darunter den eigens entwickelten Eisbrecher *FS Polarstern*. Sie dienen zur Meeresforschung in den gemäßigten Breiten, aber auch zur Versorgung und Erforschung in den Polarregionen. Darüber hinaus koordiniert das Alfred-Wegener-Institut die Polarforschung in Deutschland und stellt Ausrüstung und Logistik für Polarexpeditionen zur Verfügung.

Der Helmholtz-Gemeinschaft gehören 15 naturwissenschaftliche, biologische, medizinische und technische Forschungszentren mit insgesamt 24.000 Beschäftigten an. Pro Jahr steht der Gemeinschaft ein Budget von ca. 2 Milliarden Euro zur Verfügung. 70 % dieser Mittel stellen Bund und Länder, der Rest wird als Drittmittel eingeworben.

Das Alfred-Wegener-Institut beschäftigt zur Zeit ca. 770 Mitarbeiter. Für seinen Schwerpunkt, der Polar- und Meeresforschung, stehen dem Institut jährlich 100 Millionen Euro zur Verfügung.

Im Rechenzentrum in Bremerhaven arbeiten ca. 30 Angestellte an der Einrichtung und Administration der Netzwerke, Datenbanken, Backupsysteme, Mail-, Web-, File- und Computeserver und Rechencluster. Darüber hinaus werden parallele Prozesse und Modelle definiert und auf den Hochleistungsclustern berechnet, darunter globale Ozeanmodelle und in Zukunft auch Berechnungen im Rahmen des deutschen Tsunami-Frühwarnsystems im Indischen Ozean. Für die aufwendigen Berechnungen stehen mehrere Clustersysteme, unter anderem ein Cray XD1<sup>1</sup> und eine IBM Regatta P655<sup>2</sup> zur Verfügung (Abbildungen 2.2 und 2.3).



Abbildung 2.1.: Das Alfred-Wegener-Institut (Neubau)

## 2.2. Forschungsnetzwerke im Vergleich

Universitäten, Behörden, Wissenschaftseinrichtungen und Unternehmen weltweit haben es sich zur Aufgabe gemacht, leistungsfähige und innovative Rechnerverbünde für Forschung und Entwicklung zur Verfügung zu stellen. Sie bieten Ihren Mitgliedern und Kunden Zugang zu Hochgeschwindigkeitsnetzen und entwickeln neue Lösungen, um die Zusammenarbeit in diesem Sektor zu verbessern.

<sup>1</sup>72 Prozessoren mit 2,2 GHz, 144 GB RAM

<sup>2</sup>5 Knoten à 8 Power4+-Prozessoren mit 1,7 GHz und 32 GB RAM

## 2.2. Forschungsnetzwerke im Vergleich



Abbildung 2.2.: SunFire 6800 und Cray XD1



Abbildung 2.3.: IBM Regatta P655

Oft sind diese Netzwerke Vorreiter, wenn es darum geht, neue Technologien und Konzepte zu entwickeln, die erst sehr viel später Eingang in die kommerziellen Netze finden.

Die folgenden Abschnitte betrachten einige dieser Forschungsnetze näher. Neben dem *Deutschen Forschungsnetz – DFN* (Kapitel 2.2.1) und dem schweizerischen *SWITCH* (Kapitel 2.2.2) wird auch auf die Arbeit von *TERENA*, dem Verbund der europäischen Forschungsnetze (Kapitel 2.2.3), sowie auf das amerikanische Netzwerk *Internet2* (Kapitel 2.2.4) eingegangen.

### 2.2.1. Deutsches Forschungsnetzwerk – DFN

Die Rechnernetz-Kompetenzen der deutschen Wissenschaft bündeln sich seit nunmehr über 20 Jahren im *Verein zur Förderung eines deutschen Forschungsnetzes – DFN-Verein* mit Sitz in Berlin. Er ist Betreiber des *Deutschen Forschungsnetzes – DFN*.

Das physische Netz wird unter dem Namen *Gigabit-Wissenschaftsnetz – G-WiN* [17] betrieben. Es verbindet auf nationaler Ebene ca. 550 Hochschulen, Labore und andere wissenschaftliche Einrichtungen mit Bandbreiten zwischen 128 kBit/s und 2,4 GBit/s mit dem Kernnetz. Dieses Backbone besteht seinerseits aus 27 Kernnetz-knoten, die untereinander mit 2,5 GBit/s bis zu 10 GBit/s verbunden sind. Während die Knoten selbst mit eigener Hardware betrieben werden, kauft der DFN-Verein Leitungen im Rahmen regelmäßiger Ausschreibungen auf dem freien Markt ein. Auf diese Weise ist der DFN-Verein in der Lage, seinen Mitgliedern hochverfügbare Netze mit kurzen Laufzeiten gegen Entgelt zur Verfügung zu stellen. Der Verein übernimmt auch die Weiterentwicklung des Netzes an künftige Nutzeranforderungen, wie z.B. der mobile Zugang zum Netz, Implementierung von Quality of Service – QoS und IPv6.

Dank der garantierten Bandbreiten und Latenzzeiten eignet sich das G-WiN für den Aufbau von transparenten Rechner-Grids, also der Bereitstellung von verteilten Rechenressourcen und Speicherkapazitäten. Zu diesem Zweck engagiert sich das DFN zusammen mit anderen Wissenschaftsorganisationen in der D-Grid-Initiative [20] zur Förderung eines e-Science-Frameworks in Deutschland. *e-Science* bezeichnet allgemein die kollaborative Nutzung von verteilten Ressourcen über Institutionsgrenzen hinweg. Im Rahmen dieser Initiative wird es auch um die Authentisierung und Autorisierung von Personen und Gruppen gehen. Das DFN ist mit seinen Dienstleistungen in der Lage, Virtuelle Organisationen zu unterstützen. So wird für den Höchstleistungsrechnerverbund Nord zur Kopplung seiner Supercomputer in Hannover und Berlin eine exklusive optische Verbindung bereitgestellt.

Das Bundesministerium für Bildung und Forschung hat im Sommer 2004 die Summe von 100 Mio. Euro für die Förderung von e-Science in Deutschland in Aussicht gestellt [15]. Ähnliche Summen sollen aus der Wissenschaftsgemeinschaft und der Wirtschaft einfließen. Diese Summen verdeutlichen den Stellenwert, den die Nutzung verteilter Rechenressourcen in Zukunft einnehmen wird. Künftig wird man in der Lage sein, über einfache Schnittstellen bzw. Terminals auf ganze Rech-

nerverbünde zugreifen zu können. Die Zusammenarbeit über Institutsgrenzen hinweg wird an Bedeutung gewinnen und neue Aspekte und Perspektiven für die Lösung wissenschaftlicher Fragestellungen ermöglichen.

Neben den Aufgaben der Weiterentwicklung des DFN ist der DFN-Verein auch Gesellschafter des *Computer Emergency Response Team – DFN-CERT* in Hamburg. Dieses Team überprüft das Netzwerk auf Sicherheitslücken und betreibt einen regen Informationsaustausch mit anderen Notfall-Teams. Es verbreitet regelmäßig bzw. im Notfall nach Bedarf Sicherheitsbulletins zu aktuellen, sicherheitskritischen Themen. Daneben betreibt das DFN-CERT zur Förderung einer umfassenden Public-Key-Infrastruktur eine *Policy Certification Authority – DFN-PCA*. Sie fungiert als Zertifizierungsinstanz des DFN und stellt auch für Privatpersonen Zertifizierungsdienste zur Verfügung. Von ihr ausgestellte Zertifikate sind jedoch im Sinne des Signaturgesetzes keine fortgeschrittenen, qualifizierten oder akkreditierten Zertifikate, vgl. [11, §§ 2, 4].

Über das pan-europäische Netzwerk *GÉANT* [19] ist das DFN mit den anderen nationalen Forschungsnetzen (*National Research and Education Networks – NRENs*) in Europa verbunden. Mit Ende dieses Jahres wird dieses Netzwerk durch *GÉANT2* ersetzt, das eine erneute Leistungssteigerung verspricht. Daneben ist das DFN in der *Trans-European Research and Education Network Association – TERENA* aktiv. Näheres zur Arbeit von TERENA siehe Abschnitt 2.2.3.

### 2.2.2. The Swiss Education & Research Network – SWITCH

Das *Swiss Education & Research Network – SWITCH* bietet in der Schweiz ähnliche Dienste an wie das DFN in Deutschland. SWITCH ist als Stiftung des schweizerischen Bundes und der Universitäten organisiert. Seine Mission ist in der Stiftungsurkunde verankert:

„Die Stiftung bezweckt, die nötigen Grundlagen für den wirksamen Gebrauch moderner Methoden für Teleinformatik im Dienste der Lehre und Forschung in der Schweiz zu schaffen, zu fördern, anzubieten, sich an solchen zu beteiligen und sie zu erhalten.“ (Zitiert nach [34, S. 10])

Im Unterschied zum DFN baut SWITCH jedoch eine eigene Netzinfrastruktur auf, die ihm Unabhängigkeit von kommerziellen Anbietern bescheren wird. Dieses Projekt *SWITCHlambda* [37] basiert auf einem Glasfasernetz mit bis zu 10 GBit/s je Faser. Darüber hinaus ist SWITCH der Registrar für die Topleveldomain *.ch*.

Ebenso wie das DFN engagiert sich SWITCH in vielen internationalen Organisationen, darunter in TERENA (siehe 2.2.3), DANTE [1], Internet2 (siehe 2.2.4) und der ICANN [4]. Über das *GÉANT*-Projekt ist das schweizerische Forschungsnetzwerk mit den europäischen Netzwerken verbunden.

Im Jahr 2001 konnte eine Studie vorgelegt werden, welche die zukünftigen Anforderungen von Lehrenden und Studierenden hinsichtlich ihres Mobilitätsbedürfnisses untersucht und einen daraus resultierenden Bedarf nach einer eindeutigen ortsunabhängigen Identifikation und Autorisierung beschreibt, um z.B. Gast-

studenten Zugriff auf Ressourcen der Gastschule zu gewähren. Diese Vision unter dem Namen *eAcademia* fußt auf einer älteren Idee nach der Schaffung einer Authentisierungs- und Autorisierungsinfrastruktur (AAI), wie sie vom Internet2 Konsortium vorgeschlagen wurde. Ende 2003 konnte die AAI-Pilotphase abgeschlossen werden.

Inzwischen sind 7 Hochschulen mit insgesamt über 110.000 Nutzern dieser AAI angeschlossen. Sie vereinfacht die Identifizierung und Autorisierung der Anwender ganz erheblich, erspart sie doch den Diensteanbietern (Ressource Providers) die Anmeldung und Verwaltung vieler einzelner Endnutzer. Den Endnutzern genügt zukünftig ein einziges Passwort, um Zugang zu einer Vielzahl von Ressourcen zu erhalten. Derzeit wird diese Technologie vor allem im eLearning-Umfeld eingesetzt. Sie basiert auf der von Internet2 entwickelten OpenSource-Lösung *Shibboleth* [7].

Das Prinzip einer AAI ist in den Abbildungen 2.4 und 2.5 dargestellt. Ohne eine AAI muss sich der Nutzer für jeden einzelnen Dienst separat registrieren, unter Umständen vergehen sogar mehrere Tage, bis er alle notwendigen Ressourcen benutzen kann. Gleichzeitig entsteht bei den Diensteanbietern ein hoher Verwaltungsaufwand, der mit dem Erfassen neuer Nutzer beginnt und über die Pflege der Nutzerdaten beim Austragen nicht mehr berechtigter Anwender endet.

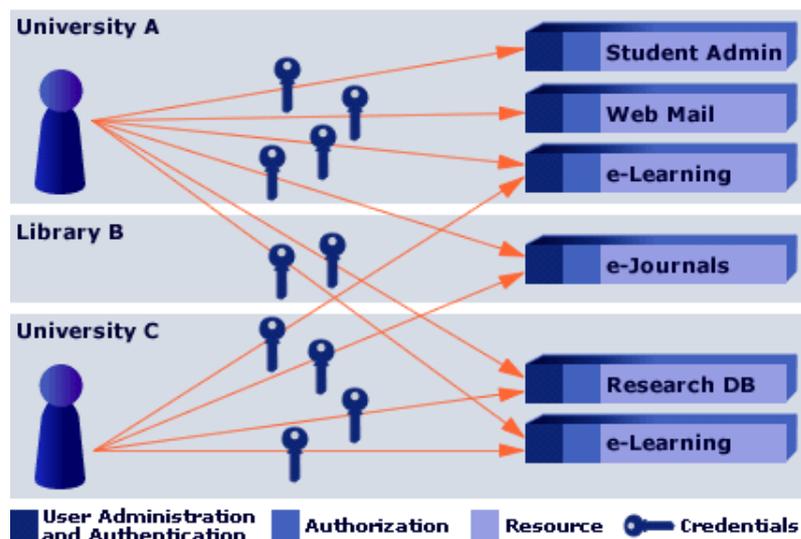


Abbildung 2.4.: Dienstzugriff ohne AAI

Mit einer AAI genügt eine einmalige Registrierung, z.B. im Studierendensekretariat. Sollen später weitere Dienste gestattet werden, genügt ein Eintrag in einer zentralen Datenbank, um den Zugriff zu erlauben. Die Autorisierung erfolgt dabei nicht mehr personengebunden, sondern z.B. aufgrund bestimmter Attribute, die in den Nutzereinträgen gesetzt sind. Hier bietet sich das Attribut `eduPersonEntitlement` an, vgl. Abschnitt 3.3.3.

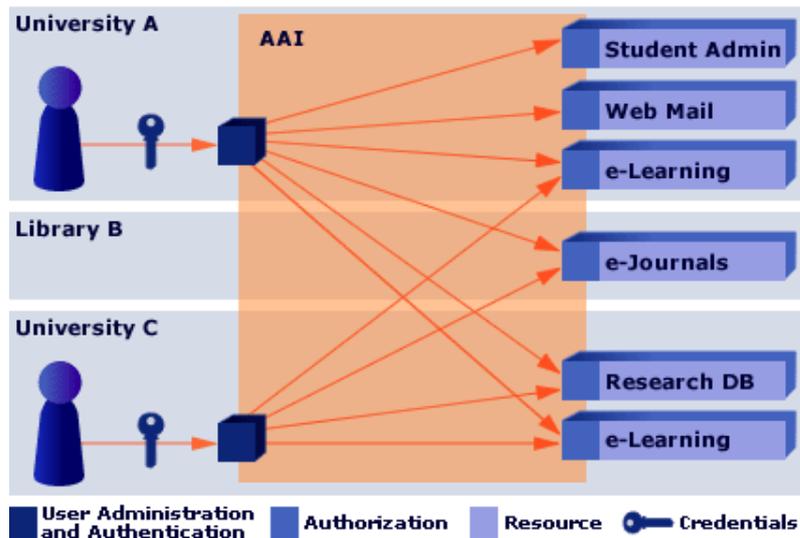


Abbildung 2.5.: Dienstzugriff mit AAI

Die Schweiz darf daher als führend in der Einführung einer landesweiten AAI-Struktur gelten. Ähnliche Ansätze sind beim DFN bisher nicht zu erkennen.

Mit dem Dienst *SWITCHmobile* [36] stellt die Schweiz Angehörigen einer Hochschule mobile Zugänge an fremden Standorten zur Verfügung. Der Mobilität der Nutzer wird somit weiter Rechnung getragen. Die Anwender erhalten Zugriff auf das Internet, die IT-Ressourcen ihrer eigenen Hochschule und ggf. auf Ressourcen der Gasthochschule. Zu diesem Zweck wird zwischen den mobilen Endgeräten der Nutzer, per WLAN im Netzwerk der Gasthochschule, und deren eigener Hochschule ein *Virtual Private Network – VPN* aufgebaut. Eine ähnliche Dienstleistung wird vom DFN ebenfalls nicht angeboten.

### 2.2.3. Trans-European Research and Education Networking Association – TERENA

Derzeit sind etwa 30 europäische Forschungsnetze Mitglied von *TERENA*, der 1994 gegründeten *Trans-European Research and Education Networking Association*. Diese Organisation mit Sitz in Amsterdam hat es sich zur Aufgabe gemacht, die

„Entwicklung von hochwertiger Informations- und Telekommunikationsinfrastruktur zum Wohle von Forschung und Ausbildung, und zwar auf Basis offener Standards und fortschrittlichster Technologie.“<sup>3</sup>  
[38]

<sup>3</sup>Original: „[...]to promote and participate in the development of a high quality information and telecommunications infrastructure for the benefit of research and education[...] based on upon open standards and uses the most advanced technology available.“

Um diesem Zweck gerecht zu werden, beobachtet *TERENA* neue Entwicklungen und Ideen, um diese bereits zu einem frühen Zeitpunkt aufzugreifen und in aktuelle Projekte einfließen zu lassen:

„[...] the *TERENA* Secretariat staff have an important role in picking up trends and (sometimes very rudimentary) ideas for potential new activities, and in investigating their feasibility and the support for them“ [39].

Es werden Kontakte zwischen Netzbetreibern, Spezialisten und Managern in ganz Europa vermittelt, um die Landschaft der Forschungsnetze voranzubringen und in eine gemeinsame Richtung zu lenken. *TERENA* greift bei Fachfragen auf die Mitarbeiter der angeschlossenen Organisationen zurück und verfügt so über einen breiten Erfahrungsschatz. Dabei ist man nicht nur auf öffentliche Einrichtungen und Non-Profit-Organisationen angewiesen, sondern kann auch auf beteiligte Industrieunternehmen zugehen, wie z.B. 3Com, Cisco und IBM.

*TERENA* versteht sich als „Sprachrohr“ der europäischen Forschungsnetzwerke gegenüber Behörden, der Industrie und anderen Organisationen. Ein wesentlicher Aspekt ist der Wissenstransfer zwischen den Mitgliedern, dem durch Konferenzen, Workshops und Seminare Rechnung getragen wird. Daneben wird sich der Weiterentwicklung der Netzwerke und der Pflege guter Kontakte zu ähnlichen Einrichtungen weltweit gewidmet.

So wurde im April dieses Jahres das Projekt *6DISS - IPv6 Dissemination and Exploitation* gestartet. Es soll die Verbreitung neuer Internetprotokolle, vornehmlich IPv6, auf dem Balkan, in der Mittelmeerregion, in Afrika und Mittel- und Südamerika voranbringen [41]. Eine weitere so genannte Taskforce entwickelt Strategien, um Verzeichnisdienste für Authentisierungs- und Autorisierungsinfrastrukturen in Verbindung mit Single Sign-On-Systemen zu etablieren [40].

Dies sind nur zwei Beispiele für die vielfältigen Aktivitäten von *TERENA*. Weitere Projekte betreffen Grid-Technologien, Private-/Public-Key Infrastrukturen und Unterstützung für mobile Dienste. Zusammen mit der Schwesterorganisation *DANTE* [1] werden in den nächsten Jahren 200 Millionen Euro in die Entwicklung des europäischen Hochgeschwindigkeitsnetzes *GÉANT2* [2] investiert, um Datenraten bis zu 32 GBit/s zu ermöglichen.

### 2.2.4. Internet2

„Led by more than 200 U.S. universities, working with industry and government, Internet2 develops and deploys advanced network applications and technologies for research and higher education, accelerating the creation of tomorrow's Internet.“

Wie auch *SWITCH* und das *DFN* ist *Internet2* bestrebt, neue Technologien im Interesse seiner Mitglieder zu entwickeln, zu testen und schließlich einer breiten

### 2.3. Einbindung des AWI in das Deutsche Forschungsnetzwerk

Gemeinschaft zugänglich zu machen. Zu diesem Zweck baut das Netzwerk ständig neue Kontakte zu akademischen Einrichtungen, Forschungsinstituten, Behörden und der Industrie auf. Seinen Mitgliedern stellt das Netzwerk einen 10 GBit/s Backbone mit Namen *Abilene* [24] zur Verfügung, das IPv6 und Multicast bietet.

Seit Ende der 1990er Jahre arbeitet die *Middleware Initiative* [5] daran, verschiedenste Dienste an unterschiedlichen Standorten zusammenzufassen und über einheitliche Zugänge der wissenschaftlichen Gemeinschaft zugänglich zu machen. „Middleware“ bezeichnet eine Art „Kleber“ zwischen Anwendungen und Diensten. Sie bietet standardisierte Identifikations- und Autorisierungsdienste, Zugriff auf Verzeichnisdienste und berücksichtigt Belange der Sicherheit. Anwendungen können also über standardisierte Schnittstellen auf bestimmte Basisdienste zugreifen.

Diese Initiative hat auch das *eduPerson*-Schema [25] initiiert, das am AWI seine Anwendung findet. Mit *Shibboleth* [7] wird ein Framework zur Verfügung gestellt, um Zugriffsberechtigungen auf entfernte Ressourcen mit einheitlichen Nutzerkennungen zu ermöglichen. Diese Anwendung basiert auf der Auswertung von Attributwerten, die über Personen in Verzeichnisdiensten gespeichert werden. Dabei werden *Identity Provider* und *Resource Provider* unterschieden. Idealerweise muss der Dienstanbieter die Identität einer Person nicht kennen, um ihr Zugriff zu gewähren. Vielmehr liefert der Identity Provider einen Satz Attribute an den Resource Provider zurück, anhand derer die Zugangsberechtigung geprüft wird. Nähere Details können unter [31] nachgelesen werden.

Dieses Verfahren erfordert eine gemeinsame Vertrauensbasis, die außerhalb der technischen Implementierung gelegt werden muss. Diese Basis wird in so genannten *Federations* geschaffen. Innerhalb dieser Bündnisse besteht Einigkeit über die technischen Rahmendaten, vor allem aber auch über das Nutzer-Management und die Korrektheit der gespeicherten Daten. Mit SWITCH besteht auch in Europa eine solche Federation.

Mit *Groupier* befindet sich eine Gruppenverwaltung in der Entwicklung, derzeit in der Version 0.5.6 [26]. Aufgrund der Komplexität der Software und der unterschiedlichen Voraussetzungen wurde jedoch am AWI entschieden, eine eigene Lösung zu entwickeln.

Neben den hier genannten Diensten bietet *Internet2* ein breites Spektrum weiterer Anwendungen, darunter Anwendungen für die Echtzeitübertragung medizinischer Daten, Technologien für Virtuelle Realitäten und die verzögerungsfreie Übertragung von Videoströmen.

### 2.3. Einbindung des AWI in das Deutsche Forschungsnetzwerk

Das Alfred-Wegener-Institut ist mit einer 34 MBit/s Leitung an das G-WiN des DFN angeschlossen. Im Bremer Landesnetz und im *Berlin Research Area Information Network – BRAIN* ist das AWI mit 155 MBit/s erreichbar. Die Forschungsstellen Sylt, Helgoland und Potsdam sind mit jeweils 2 MBit/s mit dem Stammsitz in Bre-

merhaven verbunden, während Polarstern und die polaren Forschungsstationen über Satellitenverbindungen mit 64 bzw. 128 kBit/s auskommen müssen (Stand 2004).

Mit seinen Parallelcomputern stellt das AWI den Forschern leistungsfähige Rechnerhardware für deren Modelle bereit. Sie sind in den *Norddeutschen Verbund für Hoch- und Höchstleistungsrechnen – HLRN* [6] integriert und stehen prinzipiell auch AWI-fremden Personen offen.

Vor diesem Hintergrund gewinnen nicht nur organisationsweite Berechtigungsverwaltungen an Bedeutung, sondern immer mehr auch nationale und internationale Authentisierungs- und Autorisierungsinfrastrukturen. Während sich solche Lösungen in den Forschungsnetzen anderer Länder inzwischen etabliert haben, z.B. in der Schweiz, den Niederlanden, Finnland, den USA und anderen, waren im deutschen Forschungsnetz bisher keine derartigen Strukturen in Sicht. Mit der D-Grid-Initiative soll nun auch in Deutschland die verteilte Zusammenarbeit vorangetrieben werden.

## 2.4. Verwaltung von Organisationen in Directories

Das AWI nutzt seit 1999 einen LDAP-Server zur Verwaltung verschiedener Daten und Informationen. Neben den Nutzeraccounts für die Anmeldung an den Arbeitsplatzrechnern und Mailkonten werden auch Publikationen und Patente, Organisationseinheiten, Mailinglisten und weitere Daten in diesem Verzeichnisdienst erfasst.

Ein Directory unterscheidet sich grundlegend von einer relationalen Datenbank. Während eine Datenbank die Daten in Relationen ablegt, also in einer tabellenartigen Struktur, nutzt ein Directory eine Baumstruktur, den *Directory Information Tree – DIT*. Auf diesem Baum kann prinzipbedingt sehr schnell gesucht werden, vor allem kann die Suche auf Teilbäume eingeschränkt werden. Durch eine Indizierung der Datenbestände lässt sich der Geschwindigkeitsvorteil weiter ausbauen. Demgegenüber ist der Schreibzugriff in der Regel weniger performant, allerdings werden die Daten nur vergleichsweise selten geändert, so dass dieser Nachteil nicht weiter ins Gewicht fällt.

Darüber hinaus stellen Verzeichnisdienste nur wenige Funktionen bereit, die über die reine Datenhaltung hinausgehen. Aggregatfunktionen, wie das dynamische Erzeugen neuer Objekttypen (vgl. Views in relationalen Datenbanken), die Bestimmung der Trefferanzahl oder auch referentielle Integrität gehören nicht zum üblichen Leistungsumfang eines Directory-Servers. Vor allem die fehlende Möglichkeit, über eine Art „Fremdschlüssel“ Einträge technisch referenzieren zu können, macht eine sorgfältige Planung des DIT bereits im Vorfeld notwendig.

Nach Tom Barton [14] gibt es vier Möglichkeiten, in einem Directory eine Gruppe von Einträgen abzubilden. Bei *statischen* Gruppen ergibt sich die Mitgliedschaft aus Einträgen beim Gruppenobjekt, vgl. Listing 2.1. Der Einfachheit halber werden im

folgenden Personen zu Gruppen zusammengefasst; die Ausführungen betreffen jedoch alle Arten von Gruppen oder Zusammenschlüssen.

```
dn: ou=group1,ou=Groups,o=somewhere
uniqueMember: uid=Alice,ou=People,o=somewhere
uniqueMember: uid=Bob,ou=People,o=somewhere
```

```
dn: uid=Alice,ou=People,o=somewhere
uid: Alice
```

```
dn: uid=Bob,ou=People,o=somewhere
uid: Bob
```

### Listing 2.1: Beispiel für statische Gruppen

Der Vorteil dieser Methode liegt darin, dass direkt ersichtlich ist, welche Einträge zu einer Gruppe gehören. Im Gegensatz dazu ist die vielleicht viel interessantere Information, zu welchen Gruppen ein Eintrag gehört, nur über eine Suche über den gesamten Baum zu erhalten.

Die *dynamische Gruppe* entsteht durch die Auswertung eines LDAP-Filters oder einer LDAP-URL [21]. Die Gruppe liegt in den Werten eines oder mehrerer Attribute der Personeneinträge begründet. Ein entsprechendes Gruppenobjekt kann, muss aber nicht existieren. Am AWI werden Maillinglisten auf diese Art und Weise verwaltet, wie Listing 2.2 darlegt.

```
dn: ou=mailgroup,ou=Groups,o=somewhere
mail: brhv@example.com
mgrpDeliverTo: ldap://ldap:389/o=somewhere??sub?(&(objectClass=
    awiPerson)(l=Bremerhaven))

dn: uid=Alice,ou=People,o=somewhere
uid: Alice
l: Bremerhaven

dn: uid=Bob,ou=People,o=somewhere
uid: Bob
l: Bremerhaven

dn: uid=Eve,ou=People,o=somewhere
uid: Eve
l: Potsdam
```

### Listing 2.2: Beispiel für dynamische Mailgruppen

Hier würde eine eMail an die Adresse brhv@example.com an Alice und Bob versandt werden, nicht aber an Eve. Alice und Bob gehören zur Gruppe der Bremerhavener, während Eve in Potsdam tätig ist. Auf diese Weise ist praktisch eine unbegrenzte Anzahl von Gruppen möglich, denn der Suchfilter kann auch in einem Programm dynamisch erzeugt und zur Abfrage benutzt werden.

Die dritte Möglichkeit ist das so genannte *Forward Referencing*. Bei diesem Prinzip werden den Mitgliedern einer Gruppe Attributwerte zugeordnet, welche die

## 2.4. Verwaltung von Organisationen in Directories

Gruppe kennzeichnen. Am AWI wird dieses Verfahren verwendet, um Personen ihren Organisationseinheiten, wie Fachbereichen und Sektionen, zuzuordnen. Dazu wird das LDAP-Schema `eduPerson` [25] genutzt. Hier wird im Attribut `eduPersonOrgUnitDN` der Distinguished Name<sup>4</sup> der Organisationseinheit eingetragen, wie Listing 2.3 zeigt. Wie in Abschnitt 2.4 dargelegt, handelt es sich dabei nicht um eine „harte“ Referenzierung, sondern lediglich um einen Querverweis, der auch ins Leere zeigen kann.

```
dn: ou=group1,ou=Groups,o=somewhere
ou: group1

dn: ou=group2,ou=Groups,o=somewhere
ou: group2

dn: uid=Alice,ou=People,o=somewhere
uid: Alice
eduPersonOrgUnitDN: ou=group1,ou=Groups,o=somewhere

dn: uid=Bob,ou=People,o=somewhere
uid: Bob
eduPersonOrgUnitDN: ou=group1,ou=Groups,o=somewhere
eduPersonOrgUnitDN: ou=group2,ou=Groups,o=somewhere
```

Listing 2.3: Beispiel für Forward Referencing

In diesem Beispiel existieren zwei Gruppen und zwei Personen. Alice ist nur Mitglied der Gruppe `group1`, Bob hingegen zusätzlich auch von `group2`. Forward Referencing ist ein Spezialfall der dynamischen Gruppe, da hier ganz konkrete Attribute verwendet werden, um die Gruppe zu bilden.

Die letzte von Tom Barton beschriebene Möglichkeit sind *spatial groups*<sup>5</sup>. Diese entstehen durch die Position eines Eintrages im DIT. Sie bilden damit auch die unflexibelste Art der Gruppenbildung. So ist die Mitgliedschaft in mehreren Gruppen nur schwer zu realisieren. Ein weiterer Nachteil wird deutlich, wenn eine Gruppe gelöscht oder verschoben werden sollte. Die Untereinträge würden bei einem Löschvorgang ebenfalls entfernt werden. Listing 2.4 greift dieses Problem auf.

```
dn: ou=group1,ou=Groups,o=somewhere
ou: group1

dn: ou=subGroup1-1,ou=group1,ou=Groups,o=somewhere
ou: subGroup1-1
cn: Climate Sciences

dn: uid=Alice,ou=subGroup1-1,ou=group1,ou=Groups,o=somewhere
uid: Alice
```

---

<sup>4</sup>Im gesamten Verzeichnis eindeutiger Name eines Eintrages

<sup>5</sup>A group is considered *spatial* if its membership is inferred from the location in the DIT.

```
title: Meteorologist
```

Listing 2.4: Beispiel für *spatial groups*

Hier ist Alice unter der Gruppe „Climate Sciences“ angeordnet. Dadurch sind Änderungen an der Organisationsstruktur in diesem Modell nur mit hohem Aufwand umsetzbar.

All diese hier beschriebenen Modelle bergen in sich das Problem der referentiellen Integrität. Wie bereits erwähnt prüft ein LDAP-Server standardmäßig nicht, ob ein über einen DN referenzierter Eintrag auch tatsächlich existiert, von einigen proprietären Lösungen abgesehen. Lediglich die Syntax des DN wird getestet. Tom Barton beschreibt zwei Ansätze, um die Integrität der Datensätze sicherzustellen.

Zum einen bieten sich Skripte an, die nachts aus einem LDIF<sup>6</sup>-Dump des Verzeichnisses oder den Änderungsprotokollen evtl. Abweichungen extrahieren und dem Administrator am Morgen zur Prüfung vorlegen. Zum anderen schlägt Barton vor, mit abstrakten Bezeichnern zu arbeiten, um Gruppen zu benennen und darauf zu referenzieren. Ein Beispiel dafür ist in Listing 2.5 notiert.

```
dn: ou=group1,ou=Groups,o=somewhere
ou: group1
groupLabel: Administration
```

```
dn: uid=Alice,ou=People,o=somewhere
title: Meteorologist
groupMemberOf: Administration
```

Listing 2.5: Beispiel für abstrakte Bezeichner

Sollte nun die Gruppe verschoben werden, müssen keine weiteren Anstrengungen unternommen werden, um die Mitgliedschaften anzupassen. Das Integritätsproblem bleibt jedoch, spätestens beim Löschen der Gruppe, grundsätzlich bestehen.

In jedem Fall bleibt es Sache des Administrators, für die Konsistenz der gespeicherten Daten zu sorgen. Welcher der hier gezeigten Wege beschritten wird, muss sorgfältig erwogen werden und hängt letztlich immer auch am Verwendungszweck und der Menge der Daten.

## 2.5. Selbstverwaltung von Gruppen

Es ist allgemein üblich, dass Benutzer ihre Nutzereinträge in Verzeichnisdiensten weitgehend selbst verwalten, zumindest wenn es um die Telefonnummer, das Foto oder andere persönliche Informationen geht. Die Verwaltung eher „technischer“ Attribute, wie der UserID, verbleibt jedoch in den Händen des Administrators.

Die Zuordnung einer Person zu einer Gruppe ist auf verschiedene Weisen möglich. Diese sind nicht zuletzt durch die Art der Gruppe und das Verhältnis der Person zu dieser Gruppe bedingt. Eine disziplinarische Zuordnung darf nicht, bzw.

<sup>6</sup>LDAP Data Interchange Format, textuelles Format zum Austausch von Verzeichniseinträgen [18]

erst nach einer Genehmigung, verändert werden. Das betrifft am AWI z.B. die Zugehörigkeit zu einem Fachbereich. An personal groups, vgl. Abschnitt 2.4, werden jedoch keine besonderen Bedingungen geknüpft, daher können die Mitarbeiter selbst entscheiden, ob sie zu einer solchen Gruppe gehören möchten oder nicht.

Es sind verschiedene Arten von Gruppen denkbar. Mit *geschlossenen Gruppen* werden hier diejenigen Gruppen bezeichnet, die nicht jedem Mitarbeiter offen stehen. Eine neue Mitgliedschaft oder die Beendigung einer bestehenden Mitgliedschaft muss durch einen dazu Berechtigten genehmigt und dokumentiert werden. Bei neuen Nutzeranträgen geschieht das am AWI durch eine Zuordnung in *eAccount*, die von einem Sektions- oder Fachbereichsleiter unterschrieben werden muss. Diese Dokumentation kann auch durch eine Logdatei sichergestellt werden, in der alle relevanten Änderungen notiert werden.

Tom Barton definiert *persönliche Gruppen* [14], die nicht notwendigerweise den Aufgaben der Organisation dienen. Das kann z.B. eine interdisziplinäre Forschungsgruppe sein, die sich über ein bestimmtes Thema austauschen möchte, ebenso wie eine Liste der Mitarbeiter, die sich wöchentlich zum Segeln trifft. Hier liegt es an der Institution, welche Gruppen sie zulässt und wem sie die Pflege der Gruppen überlässt.

Darüber hinaus muss entschieden werden, ob und in welchem Umfang nicht-offizielle Gruppen in öffentlichen Übersichten erscheinen dürfen. Die erwähnte Forschungsgruppe mag noch für die anderen Mitarbeiter, vielleicht auch für die breite Öffentlichkeit interessant sein. Der Segelkurs ist es ganz sicher nicht. Zum einen muss also sichergestellt werden, dass private oder persönliche Gruppen nicht mit aufgelistet werden, und zum anderen dürfen diese Gruppen nicht in Entscheidungen über Zugriffsrechte und ähnliche Prozesse einbezogen werden. Hier empfiehlt sich die Verwendung eines eigenen Namensraums oder auch eines eigenen Teilbaums im Verzeichnis. Auch die Verwendung eines eigenen Attributes, welches die Sichtbarkeit einer Gruppe definiert, ist möglich, wie z.B. `awiGroupVisibility`.

In den meisten Organisationen wird es ohne weiteres möglich sein, die Gruppenpflege den offiziellen Vertretern der Gruppe zu übertragen, wie Abteilungsleitern und deren Sekretariaten. Diese Personen kennen ihre Mitarbeiter und können entscheiden, wer berechtigt ist, zu ihrer Gruppe zu gehören. Daher können sie diese Informationen selbst pflegen, ohne dass der Directory Administrator eingreifen muss.

Das ist allerdings nur praktikabel, wenn diesem Personenkreis geeignete Werkzeuge für die Verwaltung ihrer Gruppen zur Verfügung gestellt werden. Ein direktes Bearbeiten der Einträge im LDAP ist nicht zumutbar.

Barton weist auf einen weiteren Aspekt hin. Viele Gruppen, Mailinglisten und dergleichen mehr werden neu kreiert, geraten aber bald darauf in Vergessenheit oder werden überhaupt nicht genutzt. Hier sollte demzufolge ein Alterungsprozess definiert werden. Es werden zwei Modelle vorgeschlagen: *keep-alive* und *inactivity timer*. Im ersten Fall wird ein Ablaufdatum eingetragen und rechtzeitig vor diesem Stichtag der Gruppeneigentümer informiert. Im zweiten Fall wird die Zeit seit der letzten Aktion notiert und nach einer gewissen Zeitspanne die Gruppe deaktiviert.

In beiden Fällen ist ein automatischer Prozess zu definieren, der die Informationen auswertet und entsprechend agiert.

Ein interessanter Aspekt ist die Zuordnung von bestimmten Rechten zu einer Gruppe. Das können z.B. Zugriffsrechte auf Institutsressourcen, wie Rechenzeiten auf den Hochleistungsclustern sein, aber auch Zutrittsberechtigungen zu gesicherten Räumen und Bereichen. Auch hier liegt es in der Verantwortung der Organisation klar zu regeln, welche Rechte einer Gruppe zugeordnet werden können. So wird zum Beispiel der Zugriff auf eLearning-Materialien problemlos der Gruppe der eingeschriebenen Studierenden erlaubt werden können.

Um aber Zugang zum Serverraum zu erlangen, ist es häufig zweckmäßiger dieses Recht personenbezogen zu definieren, wenn nicht automatisch alle Administratoren dazu berechtigt sein sollen.

Diese Restriktionen und Regeln werden üblicherweise in so genannten „Policies“ definiert. Sie dienen zum einen dem geordneten Ablauf innerhalb der Organisation, können aber auch Bestandteil von übergreifenden Vereinbarungen mit externen Partnern werden. Im Rahmen einer AAI wird das Vertrauen zwischen Resource Provider einerseits und Identity Provider andererseits auf Basis derartiger Policies hergestellt. Die Policies legen fest, welche Attribute ein Resource Provider geliefert bekommt und welche Rechte sich aus den Attributwerten ableiten lassen. Daher müssen die Policies der Provider den Anforderungen des gesamten Verbundes bzw. der „Federation“ genügen. Bei missbräuchlicher Anwendung der Policies könnten so auch Vertragsstrafen festgelegt werden.

Die Entscheidung darüber, ob eine Person anhand einer Gruppenmitgliedschaft implizit eine Autorisierung erhält, hängt natürlich wieder vom konkreten Anwendungsfall ab.

## 3. Implementierung

Nichts ist gefährlicher als eine Idee, wenn es die einzige ist, die man hat.

---

(Emile-August Chartier)

### 3.1. Anforderungen an die Anwendung

Über das in Abschnitt 1.3 definierte Lastenheft hinaus lassen sich weitere Anforderungen an die zu entwickelnde Anwendung formulieren.

Das Alfred-Wegener-Institut betreibt bereits mehrere Applikationen zur Verwaltung der anfallenden Daten, darunter Anwendungen zur Nutzerregistrierung (*eAccount*), zur Erfassung und Abfrage von Publikationen, Patenten usw. (*ePic*), zur Pflege persönlicher Daten wie Raum-, Telefonnummer und Homepage (*ePersonal*) sowie zur Darstellung der Expeditionen der Schiffe und Flugzeuge des AWI mit Berichten und Positionsinformationen (*eXpedition*). Teilweise überschneiden sich die Funktionsbereiche der Anwendungen. So kann derzeit sowohl mit *eAccount* als auch mit *ePersonal* eine Zuordnung zu einer Sektion aus Nutzersicht gemacht werden. *eAccount* ist jedoch vornehmlich dafür gedacht, Neu- und Änderungsanträge für Benutzer zu erstellen und nicht die Pflege persönlicher Daten, wie *eDirectory* es vorsieht.

Diese Anwendungen sind im Laufe der Jahre in wechselnden Programmiersprachen und mit unterschiedlichen Ansätzen, oft auch unter Zeitdruck, erstellt worden. Sie wurden immer für einen eng begrenzten Funktionsumfang entwickelt und sind in ihrer Funktion voneinander unabhängig, werden aber über Links in der Darstellungsschicht miteinander verzahnt. So wurde die Bildung eines monolithischen Systems vermieden. Allerdings existieren damit auch keine Klassenbibliotheken, welche die Anbindung an den Verzeichnisdienst vereinheitlichen würden. Der Zugriff auf die Datenbasis musste für jede Anwendung neu erstellt werden.

Da keine gemeinsame Code-Basis vorhanden ist, sind die Anwendungen nur schwer zu pflegen und an neue Anforderungen anzupassen. So führte die Umstellung der Organisationsstruktur des AWI im Herbst 2004 und der Start des Forschungsprogramms MARCOPOLI dazu, dass eine Programmiererin mehrere Tage LDAP-Suchfilter in *ePic* suchen und anpassen musste, damit alle Informationen abrufbar blieben. Hier würde eine Trennung zwischen Model, View und Controller (*MVC-Pattern* [51]) und der Ansatz „Konfigurieren Sie, statt zu integrieren“ [22, S. 136] wesentlich dazu beitragen, den Pflegeaufwand auch in Zukunft zu reduzieren.

Ein weiteres Problem stellt die bisherige Praxis dar, neue Konzepte zur Erstellung der Benutzerschnittstelle zu erarbeiten und umzusetzen, ohne auf gängige und erprobte Frameworks zurückzugreifen. Das hatte zur Folge, dass Zeit für die Implementierung von Funktionen verbraucht wurde, die bei Verwendung geeigneter Bibliotheken für die eigentliche Geschäftslogik der Anwendung zur Verfügung gestanden hätte. Beispiele solcher Frameworks sind *Jakarta Struts* [43], *Jakarta Turbine* [45], *Jakarta Tapestry* [44], *Java Server Faces* [32] und *Apple WebObjects* [13], um nur einige zu nennen. Eine nähere Betrachtung der Vor- und Nachteile dieser Lösungen würde jedoch den Rahmen dieser Arbeit sprengen.

Diese Problematik war dem Rechenzentrum von Anfang an bewusst. Dennoch wurde im Interesse einer schnellen Umsetzung der Anforderungen jeweils auf eine längere Evaluierungsphase verzichtet und individuellen Lösungen der Vorzug gegeben.

Ein ebenfalls nicht zu vernachlässigendes Thema ist der Zugriff auf die Datenbasis. Für die Anbindung relationaler Datenbanken existieren mittlerweile eine Reihe kommerzieller und freier objekt-relationaler Mapping-Tools (*ORM-Tools*), welche die Datenschicht vollständig von der Anwendungslogik abstrahieren und der Anwendung eine Reihe von Java-Klassen anbieten. Man arbeitet also nicht mehr mit Datenbankabfragen sondern mit so genannten *POJOs* – *Plain Old Java Objects*, normalerweise einfachen JavaBeans. Das Auffinden von bestimmten Objekten läuft meist über eine eigene Abfragesprache, die an SQL angelehnt ist, aber von der verwendeten Datenbank absolut unabhängig ist. Als Beispiele sind hier *Hibernate* [3], *ObjectStyle Cayenne* [28], *Oracle Toplink* [30] und *Enterprise Objects* [13] zu nennen.

Leider unterstützen die meisten kostengünstig oder frei erhältlichen ORM-Tools keine Abfragen von LDAP-Servern über die JNDI-Schnittstelle<sup>1</sup>. *Enterprise Objects* bildet hier eine Ausnahme und ist in *Apple WebObjects* integriert. Mit diesem Framework beschäftigen sich daher die Kapitel 3.2.3 und 3.3.3 eingehender.

Unter Berücksichtigung der hier erläuterten Aspekte lassen sich zusätzlich zum Lastenheft die folgenden Anforderungen formulieren:

**Geringer Pflegeaufwand** Die Anwendung muss auch mit einer Änderung der Organisationsstruktur zurechtkommen. Um weiteren Entwicklern die Einarbeitung zu erleichtern, müssen gängige Standards umgesetzt werden.

Soweit irgend möglich wird die Anwendung über Textdateien oder XML-Strukturen konfigurierbar sein, so dass Änderungen schnell und ohne erneutes Kompilieren möglich sind. Die Entwicklung wird den Sun Java Code Conventions [33] folgen.

**Weniger Lines of Code** Soweit möglich soll Code automatisch erzeugt und auf passende Frameworks zurückgegriffen werden, die beispielsweise eine einfache und komfortable Ansteuerung der Präsentationsschicht erlauben.

---

<sup>1</sup>Java Naming and Directory Interface, Standard-API um auf Verzeichnisdienste zuzugreifen

### 3.1. Anforderungen an die Anwendung

Falls sinnvoll möglich wird auf die Datenbasis über ein Mapping-Framework zugegriffen. Die Darstellungsschicht wird ebenfalls durch entsprechende Bibliotheken erzeugt, so dass die Anwendungsentwicklung sich ganz auf die Geschäftslogik konzentrieren kann. Auf diese Weise wird auch eine Abstraktion zwischen Darstellung und Anwendungslogik erreicht, so dass Änderungen an der Darstellung keinen Einfluss auf die Implementierung der Logik haben.

**Leicht erweiterbar** Es ist davon auszugehen, dass die Anwendung in Zukunft erweitert wird und z.B. die Pflege der Nutzeraccounts integriert.

Durch die Verwendung eines Frameworks können weitere Funktionen hinzugefügt werden. Durch die exakt definierte Anbindung der Datenbasis an die Geschäftslogik wird es problemlos möglich sein, weitere Funktionen auf dieser Datenbasis hinzuzufügen oder auch die Datenbasis zu ändern und zu erweitern, ohne die Anwendung zu beeinträchtigen. Die Pflege der Anwendung soll mit den von Apple zur Verfügung gestellten Tools (Xcode, WOBuilder) möglich sein.

**Angepasst im Layout** Das Design der Intranetseiten des AWI folgt einem einheitlichen Schema, im wesentlichen gekennzeichnet durch einen blauen Titelbanner und darunterliegender Nutzerschnittstelle. Dieses Erscheinungsbild soll auch hier zum Einsatz kommen.

Durch die Anwendung der W3C-Standards *HTML 4.01* [53] und *CSS 2* [52] wird in der Präsentationsschicht bzw. Nutzerschnittstelle ebenfalls eine Trennung von Struktur und Layout erreicht. Definitionen von Positionen, Abständen, Farben und Schriftarten sind so an einer Stelle im Stylesheet gebündelt und leicht anpassbar.

**Gut dokumentiert** Es ist damit zu rechnen, dass die Anwendung zukünftig auch von anderen Entwicklern erweitert wird. Daher ist auf eine gute Dokumentation zu achten.

Der Quelltext wird durchgängig kommentiert und weitgehend auf seine Funktion hin getestet. Darüber hinaus wird über entsprechende UML-Diagramme der Zusammenhang der Komponenten verdeutlicht.

**Getestet** Die Qualität der Funktionen ist geeignet sicherzustellen.

Die Anwendung wird nicht nur Integrations- und Nutzertests unterzogen. Es werden auch, soweit sinnvoll und möglich, Unittests der zu erstellenden Funktionen und Klassen geschrieben, um die Funktionsweise einer Methode zu garantieren. Die Implementierung folgt hier der Methodologie *Test Driven Development* [16, 23]. Darüber hinaus können Unittests auch als eine Form der Dokumentation betrachtet werden, da sie beschreiben, welche Funktionalität eine Methode zur Verfügung stellen muss.

**Name** Die Anwendung wird den Namen *eGroup* tragen.

Bei der Implementierung wird auf einen vollständigen softwaretechnischen Entwurf verzichtet. Dieses Vorgehen steht jedoch keinesfalls im Widerspruch dazu, eine qualitativ hochwertige Anwendung zu erstellen. Vielmehr erlauben es die skizzierten Verfahren, in Anlehnung an Methoden des agilen Projektmanagements [49], schnell zu einer lauffähigen Anwendung zu kommen, die allen Anforderungen gerecht wird, insbesondere der Erwartung nach einem geringen Pflegeaufwand.

Der eingangs erwähnte Zeitdruck, der hier durch den Rahmen dieser Diplomarbeit gesetzt ist, erfährt so eine deutliche Abmilderung. Die im Vergleich zu einem starren Entwurf gewonnene Flexibilität ermöglicht es, auch zu einem späteren Zeitpunkt leicht Änderungswünsche zu berücksichtigen.

## 3.2. Gewählte Technologien

Wie bereits erwähnt (3.1) wurden im Laufe der Zeit immer wieder Anwendungen entwickelt, bei denen kaum auf bestehende Komponenten zurückgegriffen wurde. Mit *eAccount* und *eGroup* setzen nun zwei Anwendungen auf einem gemeinsamen Framework auf. Die verwendeten Technologien und Frameworks sollen im folgenden Abschnitt näher betrachtet werden. Dabei wird auch auf entdeckte Schwachstellen hingewiesen.

### 3.2.1. Apple WebObjects

Bereits vor dem Beginn der Arbeiten an *eGroup* wurde im AWI-Rechenzentrum die Entscheidung getroffen, die Anwendung mit *Apple WebObjects* [13] zu implementieren. Als eine der wenigen Umgebungen bringt WebObjects das integriertes Objekt-relacionales Mappingframework Enterprise Objects mit, das auch auf einen LDAP-Server als Datenbasis vorbereitet ist. Daneben lassen sich mit WebObjects sehr schnell komplette Anwendungen entwickeln. *eAccount* konnte innerhalb weniger Wochen komplett neu implementiert und in den Produktivbetrieb übernommen werden.

WebObjects wurde 1996 von NeXT offiziell vorgestellt. Damals wurde es zu Preisen ab 25.000 USD ausgeliefert. Seit Version 5 basiert es vollständig auf Java und wurde zuletzt für ca. 700 USD gehandelt. Seit der WWDC 2005 (Worldwide Developers Conference) ist WebObjects in der Version 5.3 vollständig in Apples Entwicklungsumgebung Xcode integriert und damit kostenfrei erhältlich.

Bis zu diesem Zeitpunkt wurden neben Mac OS X als Entwicklerplattform auch Windows und Solaris (und damit auch Linux) unterstützt. Ab Version 5.3 wird nur noch Support für die Mac-Umgebung angeboten. Updates wird es für Windows und Solaris nicht mehr geben. Das Deployment, also die Installation auf einem Application Server, wird nur noch auf Mac OS X Server unterstützt. Es ist zwar nach wie vor möglich, eine WebObjects-Anwendung in einem standardkonformen

Servletcontainer, wie etwa Tomcat, auszuliefern, allerdings ohne jede Unterstützung seitens Apple.

WebObjects besteht aus einer Reihe von Softwaretools und einzelnen Frameworks, die zusammen eine Suite bilden. Prinzipiell erlaubt es diese Suite, lokale Anwendungen mit einer Swing-Oberfläche oder Webanwendungen mit HTML-Seiten sowie Webservices zu erstellen. Zu diesem Zweck greift der Entwickler bisher auf die mitgelieferten Softwaretools zurück, um das relationale Mapping zu erstellen (*EOModeler*), die HTML-Templates zu erstellen (*WOBuilder*) und das Projekt selbst zu verwalten (*Project Builder*). Alternativ kann WOLips als Entwicklungsumgebung verwendet werden, siehe Abschnitt 3.2.2. Unter Windows werden die genannten Anwendungen integriert.

Die Entwicklung basiert auf der Erstellung einzelner Komponenten, welche die Präsentationsschicht und die Anwendungslogik bereitstellen. Diese Komponenten können über eine Schnittstelle konfiguriert werden und so in verschiedenen Kontexten wiederverwendet werden.

Die mit WebObjects gelieferten Anwendungen erstellen neben den HTML-Templates weitere Textdateien. Im einzelnen sind das `eomodel`-Dateien für das Mapping des Models, sowie API- und WOD-Dateien für die Einrichtung der Komponenten. Sie lassen sich grundsätzlich auch von Hand erstellen und bearbeiten, allerdings sind sie kaum dokumentiert und ohne Vorlage praktisch nicht zu erstellen. Hier muss bei der Entwicklung also auf die nativen Tools zurückgegriffen werden. Seit Version 5.3 wird ihre Funktionalität für Mac-Entwickler durch Xcode-Plugins bereitgestellt. Unter Windows besteht nur die Möglichkeit, die ursprünglichen Anwendungen zu verwenden. Diese wurden jedoch seit mehreren Jahren nicht mehr gepflegt und auf neue Standards angepasst. So erzeugt WOBuilder lediglich HTML 3.2, XML-Deklarationen für XHTML-konforme Seiten werden als Fehler markiert. Praktische Hilfen, wie Codecompletion, automatisches Einrücken und Validierung fehlen oder sind allenfalls rudimentär vorhanden. Unter Linux und Solaris ist der Entwickler völlig auf sich gestellt, da diese Anwendungen nur für Mac OS und Windows zur Verfügung gestellt wurden.

Eine Komponente besteht also aus einer Java-Klasse, einem HTML-Template, einer API-Datei welche die möglichen Bindungen beschreibt, sowie einer WOD-Datei. Letztere definiert, wie die im HTML-Template eingebauten `<webobject>`-Tags dargestellt werden. Dabei kann es sich um einfache Zeichenketten handeln, aber auch um Formularfelder, Schaltflächen, Iteratoren und client-seitige Funktionalität in JavaScript-Funktionen. Sie können auch verwendet werden, um Informationen für andere Komponenten bereitzustellen. Ein Beispiel dafür ist in Abschnitt 3.4.2.7 beschrieben.

Durch die Komponentenarchitektur wird eine sehr hohe Flexibilität der Anwendung erreicht, da Komponenten nahezu beliebig erstellt und arrangiert werden können. Leider wird auf diese Technologie in den offiziellen Tutorials nicht weiter eingegangen. Lediglich in der Sekundärliteratur [27] findet sich eine nähere Erläuterung. Daher konnte diese Möglichkeit für *eGroup* erst in einem fortgeschrittenen Stadium der Implementierung genutzt werden.

Die WebObjects-Architektur erlaubt es also, sehr flexibel Anwendungen zu erstellen und zu erweitern. Allerdings ist der vorgezeichnete Weg nicht immer konfliktfrei zu beschreiten. Der Zugriff auf die Datenbasis ist nur durch eine `Session`-Instanz möglich. Die Komponentenklassen erben direkt von der `WOComponent`-Klasse, sind also fest in die Vererbungshierarchie eingebunden. Spring [8] geht hier den Weg über Interfaces, so dass die Komponenten ihre konkrete Implementierung nicht kennen müssen. Dadurch sind sie wesentlich einfacher zu testen also fest vererbte Komponenten, siehe auch Abschnitt 3.2.4.

Im Gegensatz zu WebObjects bedient sich etwa Tapestry [44] in den HTML-Templates üblicher Standard-Tags, um dynamischen Inhalt einzubinden. Während in WebObjects eine Seite etwa wie folgt angelegt wird:

```

1 <html>
  <head>
    <title>Hallo</title>
  </head>
5 <body>
  Willkommen , <webobject name="userName"></webobject>!
  </body>
</html>

```

wird die gleiche Funktionalität in Tapestry wie folgt erreicht:

```

1 <html>
  <head>
    <title>Hallo</title>
  </head>
5 <body>
  Willkommen , <span jwcid="userName">User Name</span>!
  </body>
</html>

```

Der Vorteil liegt auf der Hand – während die WebObjects-Tags in einem Browser nicht dargestellt werden und ein HTML-Editor die ihm unbekannt Elemente als Fehler anmerken wird, ist mit dem Tapestry-Template bereits eine Vorschau möglich, außerdem ist die Seite mit üblichen HTML-Editoren problemlos zu erstellen.

Die bereits erwähnten offiziellen Tutorials vermitteln eine Arbeitsweise, bei der die einzelnen Seiten sehr stark aneinander gebunden werden. So nimmt z.B. eine Seite Nutzereingaben entgegen, verarbeitet diese, erzeugt eine Instanz der nächsten Seite und ruft direkt Methoden der neuen Seite auf, um ihr die Ergebnisse der Verarbeitung zu übergeben. Ein Austausch einer Seite ist so praktisch nicht möglich, da sie sowohl von ihrer Vorgängerin als auch Nachfolgerin direkt abhängt. Ein Ausweg aus dieser Situation ist, die Werte in der `Session`-Instanz zwischenspeichern. Der Nachteil dabei ist jedoch, dass diese Klasse umso unübersichtlicher wird, je mehr Seiten implementiert werden. Auch lässt sich nur schwer nachvollziehen, in welcher Reihenfolge die Seiten aufgerufen werden, da diese Informationen über das gesamte Projekt verstreut liegen.

Bei Struts [43] gibt es kein explizites Session-Objekt. Vielmehr wird es vom Framework bereitgestellt, so dass es nicht selbst implementiert werden muss. Der „Workflow“ der Anwendung wird hier zentral in einer XML-Struktur definiert, so dass jederzeit die volle Kontrolle über die Anwendung besteht.

Für *eGroup* wurde versucht, dieses Konzept in Teilen nachzubilden. Die Namen der Seiten sind in der Konfigurationsdatei definiert und können über Methoden der *Session*-Klasse abgerufen werden.

Dagegen ist die Mehrsprachenunterstützung in *WebObjects* vorbildlich. Es genügt, einen separaten Ordner je Sprache anzulegen (*German.lproj* und *English.lproj*) und in *Session* die HTTP-Requestparameter auszuwerten. In den beiden Ordnern müssen dann jeweils sprachspezifische HTML-Templates abgelegt werden. Sprachunabhängige Komponenten können im Hauptverzeichnis verbleiben.

*Enterprise Objects* erlaubt ein Mapping von LDAP-Einträgen auf Java-Klassen und stellt eine Persistenzschicht zur Verfügung. Diese Schicht übernimmt, für die Applikation völlig transparent, das Caching der Einträge und stellt eine Abfragesprache zur Verfügung. Während als mit einfach gehaltenen Java-Objekten gearbeitet wird, spielt die konkrete Datenbasis kaum eine Rolle.

Allerdings lässt sich bei der Arbeit mit dem *Enterprise Objects Framework* feststellen, dass es näher an relationalen Datenbanken als an baumartigen Verzeichnisdiensten orientiert ist. So wird versucht, das Konzept der Fremdschlüssel aus der Welt der Datenbanken auf den Verzeichnisdienst zu übertragen. Dabei werden Einträge allerdings über ihre Relative Distinguished Name (RDN) identifiziert, während in den referenzierenden Attributen vollständige Distinguished Names (DN) eingetragen sind. Es war nicht möglich, hier eine Umsetzung von RDN in DN zu implementieren, da diese Zugriffsmechanismen im Hintergrund abgearbeitet werden.

Natürlich unterstützt *Enterprise Objects* auch Schreibzugriffe auf dem Directory. Hier kam es bei der Implementierung von *eAccount* zu Datenfehlern, so dass für Schreibaktionen von Anfang an auf die JNDI-Schnittstelle gesetzt wurde. Somit existieren in *eGroup* zwei Zugriffsmethoden: über *Enterprise Objects* und über JNDI. Weitere Details zur Persistenzschicht siehe in den Abschnitten 3.2.3 und 3.4.2.

Abschließend sei darauf hingewiesen, dass der Quelltext von *WebObjects* nicht offengelegt ist. Ein Debugging interner Vorgänge ist so nicht möglich.

*Apple WebObjects* wird von vielen Unternehmen, darunter auch einigen großen Konzernen wie der Deutsche Bank oder Toyota, eingesetzt. Man findet Unterstützung in mehreren Mailinglisten, der Sekundärliteratur und natürlich bei Apple direkt. Die Suite erlaubt die schnelle Implementierung funktionierender Anwendungen. Dem gegenüber stehen die hier beschriebenen Probleme und seit Kurzem der „vendor lock-in“ bis auf Hardwareebene, also die Festlegung auf Produkte eines Herstellers ohne echte Alternative. Daher erscheint es sinnvoll, auch einige der anderen Frameworks und Umgebungen zu evaluieren. Beispiele wurden bereits in Abschnitt 3.1 genannt.

### 3.2.2. WebObjects Eclipse Plugin – WOLips

Abseits der nativen Apple-Tools hat sich mit *ObjectStyle WOLips* ein Plugin für Eclipse etabliert, das unter einer OpenSource-Lizenz steht [29]. Es lässt sich, der Anleitung auf der Homepage folgend, unkompliziert installieren und erlaubt so auch die Entwicklung unter Linux. Unter Windows werden die nativen Apple-Anwendungen (WOBuilder, EOModeler usw.) eingebunden.

Das Plugin unterstützt den WebObjects eigenen Build-Prozess und erstellt bei Bedarf auch eine `web.xml`-Datei für ein J2EE-Deployment und eine `build.xml` für die Verwendung mit *Ant* [42]. Allerdings unterstützt es nicht Erstellung der WebObjects-typischen Dateien, wie den WOD-Files oder den Model-Beschreibungen. Falls dem Entwickler die Syntax dieser Dateien bekannt ist, können diese manuell erstellt werden, anderenfalls muss auf die Apple-Tools zurückgegriffen werden, mit der genannten Linux-Einschränkung.

Allerdings gibt es auch in der aktuell vorliegenden Version noch einige Stolpersteine. So ist ein erzeugtes WAR-Archiv im Tomcat nicht ohne weiteres lauffähig, und auch das Einbinden von Ressourcen wie CSS-Dateien und Bildern ist problematisch. Allerdings handelt es sich im Moment auch um einen Release Candidate, so dass mit derartigen Fehlern zu rechnen ist.

### 3.2.3. Persistenzschicht

Die Datenbasis im Hintergrund stellt am AWI ein *Java Enterprise System Directory Server Version 5.2* zur Verfügung. Im Vergleich zu einer relationalen Datenbank bietet ein LDAP-Server wesentlich kürzere Antwortzeiten bei Lesezugriffen. Am AWI werden so nicht nur Benutzeraccounts und Gruppen verwaltet, sondern auch ca. 13.000 Publikationen, Daten zu Expeditionen usw.

Auf einen LDAP-Server wird mit so genannten Filtern zugegriffen, deren Syntax der polnischen Notation [50] angelehnt ist. Das bedeutet im Wesentlichen, dass bei einer booleschen Verknüpfung von Suchattributen die Operation vor den Operanden steht, z.B.

```
(&(l=Bremerhaven)(|(cn=F*)(cn=G*)))
```

Dieser Filter würde im Attribut `cn` alle Einträge suchen, die mit „F“ oder „G“ (OR-Verknüpfung mit senkrechtem Strich, Pipe-Symbol) beginnen und gleichzeitig in Bremerhaven arbeiten (AND-Verknüpfung durch Kaufmanns-Und.). Darüber hinaus kann eine Suchbasis angegeben werden, unterhalb derer nach Einträgen gesucht wird.

Wie bereits erläutert, gestattet Enterprise Objects einen transparenten Zugriff auf eine Datenquelle. Zu diesem Zweck werden eine Reihe von Adaptern mitgeliefert, die auf verschiedene Datenbanken zugeschnitten sind. Darunter findet sich auch ein JNDI-Adapter, mit dem auf LDAP-Server zugegriffen werden kann.

Das EO-Framework erlaubt über eine eigene Abfragesprache den Zugriff auf die Daten unabhängig von ihrer konkreten Repräsentation. Obiger LDAP-Filter kann auch durch einen folgenden so genannten EOQualifier beschrieben werden:

```
(l='Bremerhaven') AND ((cn like 'F*') OR (cn like 'G*'))
```

Leider besteht hier ein Fehler im JNDI-Adapter, der bei verschachtelten booleschen Ausdrücken zusätzliche Klammern einfügt und so dafür sorgt, dass alle Argumente nur OR- oder nur AND-verknüpft werden. Entsprechende Nachfragen zu diesem Verhalten auf einer Entwicklermailingliste blieben leider unbeantwortet.

Um die Suche auf einen bestimmten Zweig des Directories zu beschränken ist es theoretisch möglich, einen EOQualifier nach folgendem Muster zu konstruieren:

```
(relativeDistinguishedName like 'ou=Groups') AND (cn like 'awi-2005')
```

Leider bleibt aufgrund eines weiteren Fehlers diese Variante eine theoretische Möglichkeit. Der EOQualifier wird vom Framework korrekt in einen LDAP-Filter mit der richtigen Suchbasis übersetzt und liefert vom Verzeichnisdienst auch die gewünschten Einträge zurück, wie ein Blick in die TCP-Pakete mit dem Netzwerkniffer *Ethereal* zeigt. Dann jedoch werden intern alle Einträge, bis auf den ersten, aus dem Ergebnisarray entfernt und statt dessen der verbleibende erste Eintrag entsprechend häufig wiederholt. Dieser Fehler wurde unter der Problem-ID 4131433 an Apple mitgeteilt, eine Reaktion von dort steht noch aus.

Um also die Suche auf Teilbäume des Directories zu beschränken sollten entweder verschiedene EOModel-Definitionen erstellt werden, bei denen explizit eine eigene Suchbasis angegeben wird, oder im Java-Code muss ein EOModel-Objekt mit passenden Einstellungen erzeugt werden, wie in [12, S. 104] beschrieben.

Weiterhin bietet das Framework die Möglichkeit, einen Suchfilter dynamisch zu erzeugen, indem der Methode `EOQualifier.qualifierWithQualifierFormat(String, NSArray)` ein Suchstring mit Platzhaltern und ein Array mit Werten für eben diese Platzhalter übergeben wird. Ein Beispiel für die Verwendung dieser Methode ist in Abschnitt 3.4.2.14 beschrieben.

Enterprise Objects bietet eine gute Möglichkeit, transparent auf verschiedenste Datenquellen zuzugreifen. Für relationale Datenquellen gibt es eine Reihe sowohl kommerzieller als auch freier Alternativen, siehe Abschnitt 3.1. Für den Zugriff auf Verzeichnisdienste ist die Auswahl deutlich eingeschränkter, allerdings existiert hier bereits in der Java Standard-API mit JNDI eine standardisierte Schnittstelle.

Im Rahmen der Implementierung von *eGroup* wurde versucht, eine eigene Persistenzschicht auf der Basis von Java Standardkomponenten zu entwickeln. Die entstandenen Klassen ermöglichten bereits den direkten Zugriff auf referenzierte Einträge (z.B. über das `awiGroupHead`-Attribut direkt auf den Leiter einer Gruppe) und lazy fetching, also den Abruf referenzierter Einträge erst dann, wenn sie tatsächlich benötigt werden. Leider skalierte diese Lösung linear zur Anzahl der vorhandenen Datensätze, so dass bei einigen Hundert Personeneinträgen die Wartezeiten nicht mehr akzeptabel waren. Vermutlich hätte hier ein Caching-Mechanismus wesentlich zur Verbesserung beigetragen, allerdings wurde im Sinne einer schnelleren Implementierung letztlich wieder auf Enterprise Objects zurückgegriffen.

### 3.2.4. JUnit

Wie im Abschnitt 3 eingeführt, sollte die Implementierung der Methodologie *Test Driven Development* – TDD folgen. Bei TDD wird vor der Implementierung der Anwendungslogik eine Testklasse geschrieben, welche die Funktionalität der zu testenden Klasse prüft. Im nächsten Schritt wird die eigentliche Logik-Klasse so implementiert, dass der Testfall gerade erfüllt wird. Dann wird der nächste Testfall erstellt und die Klasse ggf. angepasst. Diese Testfälle werden auch *Unittests* genannt.

Auf diese Weise wird zuerst das Ergebnis einer Funktion beschrieben und erst dann die Funktion selbst implementiert, so dass gerade das erwartete Ergebnis erreicht wird. Wird zu einem späteren Zeitpunkt der Quelltext angepasst oder eine Funktion hinzugefügt, kann über die Testfälle geprüft werden, ob die Anwendungslogik noch immer korrekte Ergebnisse liefert.

Mit *JUnit* [9] steht ein OpenSource-Framework zur Verfügung, das genau diese Möglichkeit zum Funktionstest bereitstellt. Es wurde bei *eGroup* in der Version 3.8.1 verwendet.

Die implementierten Testfälle sind eng mit der Datenbasis gekoppelt und setzen zum Teil bestimmte Datenkonstellationen voraus. Zu diesem Zweck wurden die Tests über die Datei `messages.properties` soweit wie möglich von konkreten Datensätzen getrennt. Sollten Testfälle fehlschlagen kann so ein anderer Datensatz ausgewählt werden, der die im konkreten Test abzutestenden Verhältnisse und Funktionen mitbringt.

Der einzige Weg, auf die Datenbasis zuzugreifen, geht über eine `Session`-Instanz, die zunächst recht aufwendig erzeugt werden muss. Aus diesem Grund wurde die Klasse `W0ComponentTestConstructor` geschaffen, welche die `Application`-Klasse aufruft, eine `Session` kreiert und diese einen Kontext der Datenbasis herstellen lässt.

Leider waren nicht alle Komponenten so leicht zu testen, wie es wünschenswert gewesen wäre. Zum Abschluss der Implementierung entzogen sich die in deutsch und englisch vorliegenden Komponenten völlig den Tests. Das `WebObjects`-Framework schien nicht in der Lage zu sein, die für die Initialisierung der Komponenten notwendigen Elemente aufzufinden. Welche Elemente das genau sind, ließ sich nicht ermitteln. Hier die ersten Zeilen des Stacktraces, der das Problem zumindest einkreist:

```

1 junit.framework.AssertionFailedError: Exception in constructor:
   testGetAwiGroup (java.lang.NullPointerException
   at com.webobjects.appserver._private.W0ComponentDefinition.<init>
     >(W0ComponentDefinition.java:83)
   at com.webobjects.appserver.W0Application.
     _loadComponentDefinition(W0Application.java:2098)
   at com.webobjects.appserver.W0Application._componentDefinition(
     W0Application.java:2213)
5  at com.webobjects.appserver.W0Application._componentDefinition(
     W0Application.java:2343)

```

```

    at com.webobjects.appserver.WOApplication.pageWithName(
        WOApplication.java:1878)
7   at de.awibremerhaven.egroup.Application.pageWithName(Application
        .java:221)

```

Die betreffende Klasse `WOComponentDefinition` ist nicht in der API-Dokumentation von Apple aufgeführt. Daher bleibt es im Dunkeln, wo exakt der Fehler auftritt.

### 3.2.5. Logging mit Log4J

Um die Aktionen in der Anwendung nachvollziehen und ein Logbuch der Benutzeraktionen schreiben zu können, wird Log4J der Apache Foundation [46] verwendet. Diese Bibliothek ermöglicht flexibles Logging auf verschiedenen Stufen, wie `debug`, `warn` oder `info`. Log4J wird über eine einfache Konfigurationsdatei (`log4j.properties`) eingerichtet. Die Form der Logdateien ist beliebig wählbar.

In *eGroup* wird die Bibliothek vorwiegend für Debug-Meldungen genutzt. Eine besondere Bedeutung kommt jedoch dem Logger `UserLog` zu, der die Nutzeraktionen protokolliert, also das Einladen, Einschreiben und Austragen neuer Gruppenmitglieder und das Erzeugen neuer Gruppen. Auf diese Weise können im Nachhinein Unstimmigkeiten nachvollzogen und geklärt werden.

Dieser Logger steht in jeder Klasse über die Anweisung

```
1 private static Logger userLog = Logger.getLogger("UserLog");
```

zur Verfügung. Dieser wurde über die Konfigurationsdatei `log4j.properties` so konfiguriert, dass seine Ausgaben in eine Textdatei geschrieben werden.

Diese Logmeldungen folgen einem einfachen Muster:

```

[2005-06-08 10:48:51,695] [alice] accepted
  subscription of <bob> to <cn=climate,ou=Groups> as requested.
[2005-06-13 09:49:59,333] <eve> refused the
  invitation to <cn=geosciences,ou=Groups>

```

Jede Zeile trägt einen Zeitstempel in Lokalzeit, dann in eckigen Klammern die `uid` des aktuellen Benutzers falls es sich um eine administrative Änderung handelt, gefolgt von der `uid` einer Person und dem Relative Distinguished Name einer Gruppe, die von der Änderung betroffen sind.

## 3.3. Modell

Der folgende Abschnitt befasst sich mit der Modellbildung für *eGroup*. Er beschreibt die Abbildung der Wirklichkeit, in diesem Fall der Organisationsstruktur, auf die Daten- und Anwendungsebene. Es finden sich hier daher eine Darstellung der Aufbauorganisation des AWI, die Modellierung der von *eGroup* abzudeckenden Anwendungsfälle und eine Erläuterung der notwendigen Objektattribute.

### 3.3.1. Abbildung der Organisationsstruktur

Die Forschungsaktivitäten innerhalb der Helmholtz-Gemeinschaft sind inzwischen größtenteils in Programmen zusammengefasst. Das Alfred-Wegener-Institut hat in diesem Sinne unter anderem mit der GKSS-Forschungszentrum Geesthacht GmbH das Forschungsprogramm *MARCOPOLI* ins Leben gerufen. Dieses Forschungsprogramm ist auf 5 Jahre Laufzeit angelegt, doch schon während dieser Laufzeit können neue Programme mit anderer Struktur und neuen Partnern entstehen.

Um dieser neuen Anforderung gerecht werden und dennoch auch außerhalb der programmorientierten Forschung wirken zu können, hat das AWI Ende 2004 eine duale Institutsgliederung gewählt [10]. Für die interne Forschungstätigkeit wurde dabei die generelle Gliederung in Fachbereiche und Sektionen beibehalten. Das institutsübergreifende Forschungsprogramm *MARCOPOLI* hingegen ist in Topics und Workpackages unterteilt. Zur Darstellung finden sich die beiden Organigramme des AWI und von *MARCOPOLI* im Anhang A und B. Sie sollen hier kurz erläutert werden.

Intern gliedert sich das AWI in drei Fachbereiche (Geowissenschaften, Biowissenschaften, Klimawissenschaften) und zwei Bereiche (Neue Technologien, Allgemeine Dienste). Die Fachbereiche sind in jeweils mehrere Sektionen unterteilt, die innerhalb der Fachbereiche ihre spezifischen Forschungsaufgaben wahrnehmen. Zu den Neuen Technologien gehören Abteilungen, die sich mit Flugzeugtechnik, Eisbohrungen und Erdbeobachtungssystemen beschäftigen. Den Allgemeinen Diensten sind u.a. die Verwaltung, das Rechenzentrum und die Bibliothek untergeordnet. Disziplinarisch ist jeder Mitarbeiter einem Fachbereich bzw. einer Abteilung zugeordnet. Dem Direktorium als übergeordneter Einrichtung sind verschiedene Beiräte und Stabsstellen zugeordnet (Wissenschaftlicher Rat, Personalrat und Frauenbeauftragte, Ombudsmann, Nutzerbeiräte, Wissenschaftliches Referat, Justiziar, Presse- und Öffentlichkeitsarbeit und Innenrevision).

Bei *MARCOPOLI* handelt es sich um ein inter-institutionelles Programm, bei dem die Workpackages gemeinsam durch Mitarbeiter der GKSS und des AWI bearbeitet werden. Dieses Programm bildet eine Virtuelle Organisation, da die Themen auch räumlich getrennt bearbeitet werden. Das Forschungsprogramm *MARCOPOLI* unterteilt sich in die vier Topics Marine (MAR), Coast (CO), Polar (POL) und Infrastructure (I). Daneben gibt es die beiden Projektbereiche COM und New Keys, die über Drittmittel finanziert werden. Die vier Topics sind ihrerseits in Workpackages unterteilt.

*eGroup* soll auch die Verwaltung solcher institutsübergreifender Gruppen ermöglichen. Bis zur Etablierung einer nationalen Authentisierungs- und Autorisierungsinfrastruktur ist es jedoch notwendig, zumindest einen Basisdatensatz für externe Personen anzulegen. Hier lässt sich bereits ein Anwendungsfall für ein Schema wie *eduPerson* [25] und die Entwicklungen der Internet2 Middleware Initiative [5] erkennen. In Zukunft wird es möglich sein, aus Verzeichnissen fremder Institutionen Informationen und Berechtigungen nach klar definierten Regeln auszulesen und davon ausgehend Entscheidungen über Zugriffsrechte zu treffen, vgl.

Abschnitt 2.5. *eGroup* ist ein Schritt des AWI in diese Richtung, auch in Hinblick auf die Planungen des D-Grid, das noch in diesem Jahr seinen Betrieb aufnehmen soll [20].

Diese Darstellung macht deutlich, dass ein Mitarbeiter nicht nur einer Organisationseinheit zugeordnet sein muss. Er kann durchaus zu mehreren Gruppen gehören. Es ist auch denkbar, dass er zwar zu einer Sektion gehört, nicht aber zum übergeordneten Fachbereich dieser Sektion. Er kann auch von außerhalb des AWI kommen, so dass keine weiteren Informationen über ihn direkt abrufbar sind. Und diese Struktur kann jederzeit wieder geändert werden. Abbildung 3.1 illustriert die verschiedenen Möglichkeiten.

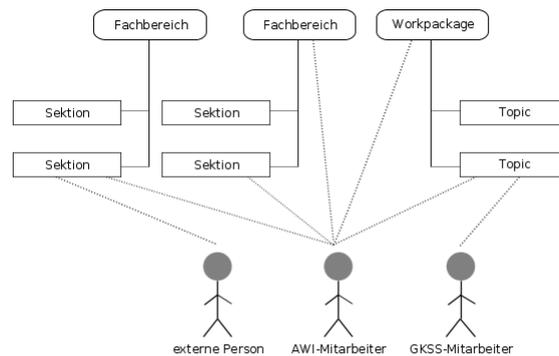


Abbildung 3.1.: Möglichkeiten der Mitgliedschaft

Zukünftig soll es den Gruppen möglich sein, sich selbst zu verwalten und zu pflegen. Untergruppen sollen angelegt und editiert werden können, Mitarbeiter hinzugefügt und auch wieder entfernt werden können. Mitarbeiter sollen sich auch selbst Gruppen zuordnen können. Es sollen Personengruppen möglich sein, die vollständig außerhalb der Organisationsstruktur stehen und z.B. die Zugriffsberechtigungen auf Fremdsysteme regeln.

Die Organisationsstruktur wird innerhalb des Directories abgebildet. Die technischen Details hierfür sind in Abschnitt 3.3.3 näher beleuchtet. *eGroup* muss diese Informationen auslesen und entsprechend darauf reagieren. So darf eine Übersichtsseite der Sektionen, zu denen ein Mitarbeiter gehört, nicht auch automatisch die passenden Fachbereiche enthalten, da zwischen Sektions- und Fachbereichszugehörigkeit kein zwingender Zusammenhang besteht. Die Nutzerschnittstelle muss also klare Aussagen machen und darf keine impliziten Zugehörigkeiten aufzeigen.

Bei der Definition neuer Gruppen muss darauf geachtet werden, dass offizielle Gruppen korrekt in die Organisationsstruktur eingebunden werden, *personal groups* aber außerhalb der üblichen Struktur stehen, siehe Abschnitt 2.4. Das betrifft sowohl die Festlegung eindeutiger technischer Kennzeichen, als auch die Referenzierung zu anderen, übergeordneten Gruppen. Da das Directory prinzipbedingt

keine referentielle Integritätsprüfung hat, kommt hier jeder schreibenden Anwendung eine große Verantwortung zu.

Für die Abbildung der verschiedenen Hierarchieebenen werden einige Schlüsselwörter definiert, deren Entsprechung zur offiziellen Bezeichnung in Tabelle 3.1 widergegeben wird. Diese internen Schlüsselwörter sind in der Konfigurationsdatei unter dem Wert `typesOf0us` abgelegt.

awiGroupDisplay-Affiliaton	awiGroupDisplay-AffiliationEN	eGroup-intern	awiGroup-Affiliation
Fachbereich	Research Division	department	Fachbereich
Sektion	Section	section	Sektion
Forschungsprogramm	Programme	programme	Fachbereich
Programmthema	Topic	topic	Fachbereich
Arbeitspaket	Workpackage	workpackage	Sektion
Thema	Theme	newThemes	Sektion
Bereich	Division	division	Fachbereich
Abteilung/Sektion	Department/Section	subDivision	Sektion
Team	Team	open	

Tabelle 3.1.: Abbildung interne Bezeichnungen – offizielle Bezeichnungen

### 3.3.2. Anwendungsfälle in der Gruppenbildung

Aus dem Lastenheft in Abschnitt 1.3 und Gesprächen mit den verantwortlichen Administratoren des AWI-Rechenzentrums werden in diesem Abschnitt die Anwendungsfälle herausgearbeitet, die *eGroup* abdecken soll. Sie reichen von der einfachen Information über Gruppen bis hin zur Verwaltung Virtueller Organisationen mit externen Mitgliedern. Diese zentralen Anwendungsfälle sind in Abbildung 3.2 in einem Use-Case-Diagramm zusammengetragen.

Grundsätzlich wird zwischen zwei Arten von Gruppen unterschieden: zum einen Gruppen, die jedem offen stehen, zum anderen nur eingeschränkt zugängliche Gruppen. Wir bezeichnen die ersteren als *offene Gruppen* (personal groups nach Barton, siehe Abschnitt 2.5) und die letzteren als *geschlossene Gruppen*. Um Mitglied einer geschlossenen Gruppe zu werden ist eine Genehmigung durch einen der Funktionsträger der Gruppe (`awiGroupHead`, `awiGroupDeputy`, `awiGroupSecretary` oder `awiGroupAuthorizedSigner`) erforderlich. Ebenso ist eine Austragung erst nach einer Genehmigung möglich. Für offene Gruppen besteht diese Einschränkung nicht.

Offene Gruppen eignen sich für abteilungsübergreifende Forschungsthemen, die für einen breiten Personenkreis interessant sind. Geschlossene Gruppen finden ihre Anwendung vor allem in der Abbildung der Organisationszugehörigkeit, für disziplinarische Zuordnungen und Virtuelle Organisationen.

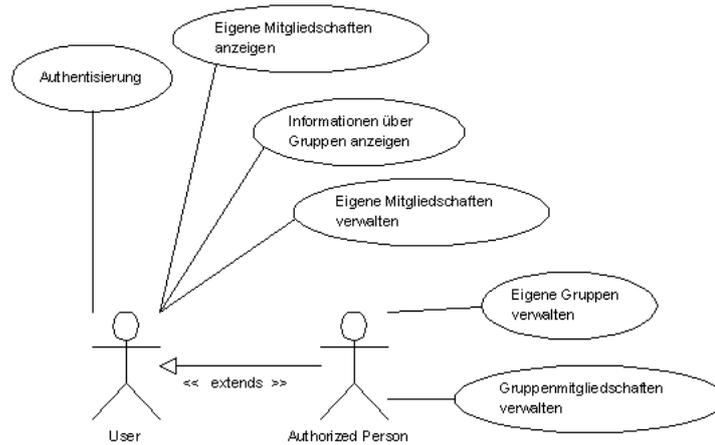


Abbildung 3.2.: Generelle Anwendungsfälle

Eine Person muss sich gegenüber der Anwendung authentisieren. Sie hat dann Zugriff auf die Information, zu welcher Gruppe sie explizit gehört (die Gruppe ist im Attribut `eduPersonOrgUnitDN` der Person eingetragen) und hiervon ausgehend auf weitere Information über die einzelnen Gruppen. Dazu gehören vor allem Angaben über den Gruppenleiter, das Sekretariat, die weiteren Mitglieder usw. Da sich diese Anwendungsfälle nicht weiter untergliedern, werden hierfür keine eigenen Use-Case-Diagramme präsentiert.

Die Verwaltung der eigenen Mitgliedschaften ist in einem Diagramm in [Abbildung 3.3](#) dargestellt. Grundsätzlich kann die Person sich selbst in offene Gruppen eintragen oder auch wieder austragen. Für geschlossene Gruppen können diese Anwendungsfälle nur über Anträge abgewickelt werden. Stellt eine Person einen solchen Antrag, erhält der Gruppenverwalter eine eMail und kann so den Antrag weiter bearbeiten. Abschließend erhält die Person eine Nachricht zu ihrem Antrag.

Ein Funktionsträger oder auch Gruppenverwalter, in den Diagrammen als *Authorized Person* bezeichnet, hat über die bisher beschriebenen Anwendungsfälle hinaus weitere Möglichkeiten. Ein Gruppenverwalter kann über die oben genannten Felder identifiziert werden. Eine differenzierte Rechtevergabe wird nicht benötigt und ist daher in *eGroup* auch nicht weiter implementiert. Lediglich Directory-Administratoren werden zusätzlich unterschieden. Sie sind durch einen bestimmten Eintrag im Attribut `eduPersonEntitlement` gekennzeichnet, vgl. [Abschnitt 3.3.3](#). Sollten in Zukunft detailliertere Unterscheidungen zwischen den Rechten etabliert werden, so sollte dieses Attribut für die Verwaltung genutzt werden.

Evtl. wird zu einem späteren Zeitpunkt ein weiterer Funktionsträger innerhalb einer Gruppe definiert, der für die Pflege der Gruppe zuständig ist. In diesem Fall ist der hierfür eingerichtete Eintrag `groupAdmins` in der Konfigurationsdatei entsprechend anzupassen.

Der Gruppenverwalter ist, wie in [Abbildung 3.4](#) gezeigt, in der Lage für bestehende Gruppen die Funktionsträger zu definieren und Attribute wie Adresse, URL

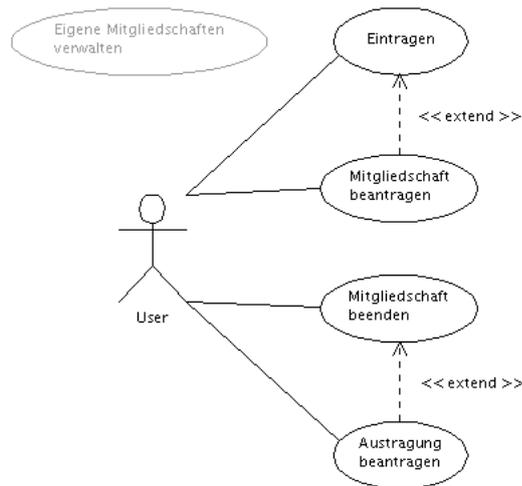


Abbildung 3.3.: Anwendungsfall: Verwaltung der eigenen Mitgliedschaften

usw. zu pflegen. Weiterhin kann jeder Mitarbeiter eigene offene Gruppen anlegen, die außerhalb der Organisationsstruktur stehen. Für diese Gruppen wird er automatisch als „Sprecher“ eingetragen und ist damit Verwalter dieser Gruppe.

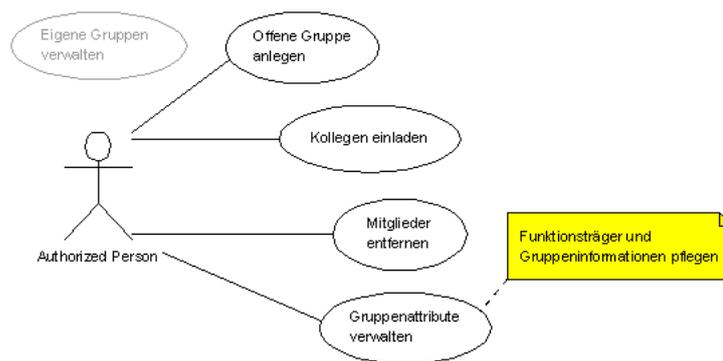


Abbildung 3.4.: Anwendungsfall: Verwaltung von Gruppeninformationen

Zu diesen offenen Gruppen kann der Verwalter Kollegen einladen (siehe Abbildung 3.5) und bestehende Mitgliedschaften aufkündigen. Bei einer Einladung erhält der betreffende Mitarbeiter eine eMail, während auf seiner eGroup-Startseite ein entsprechender Hinweis erscheint. Er kann der Einladung folgen oder sie natürlich auch ausschlagen (siehe Abbildung 3.6).

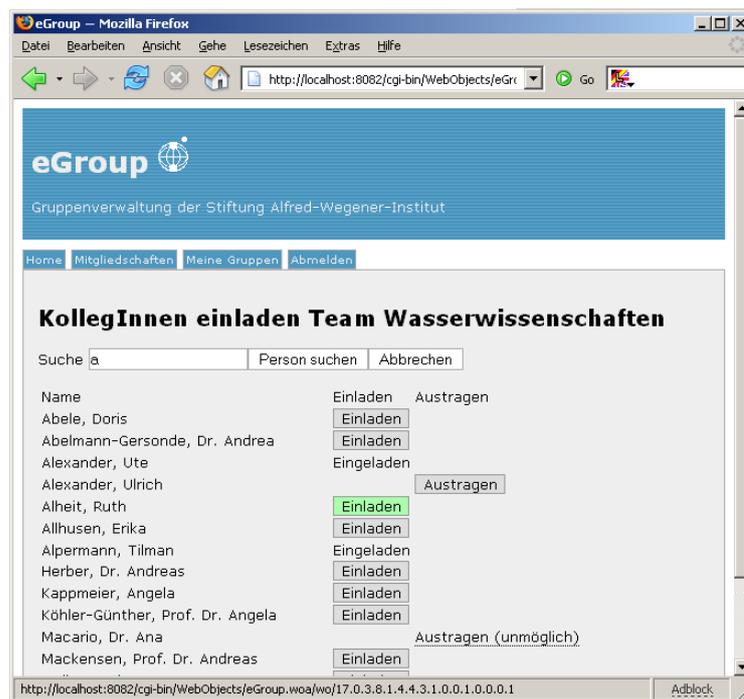


Abbildung 3.5.: Frontend KollegInnen einladen

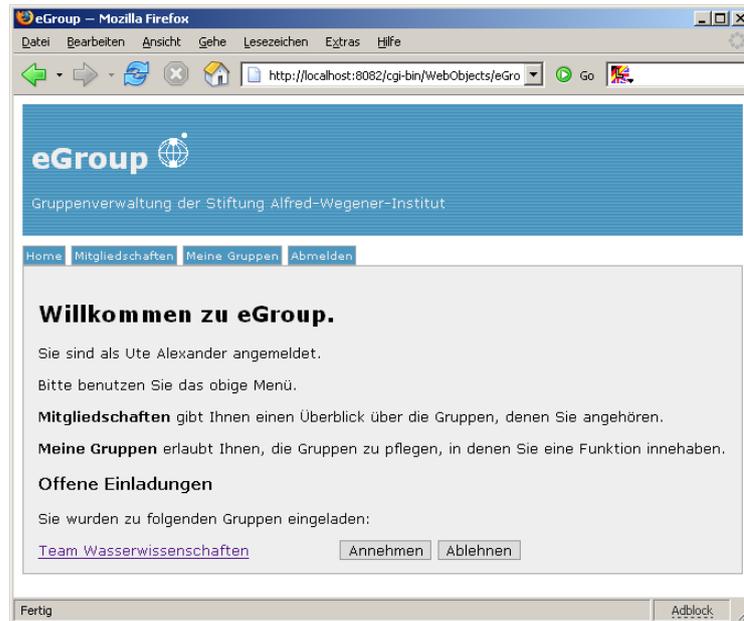


Abbildung 3.6.: Frontend Homepage

Neben der Bearbeitung der Anträge von bestehenden oder zukünftigen Mitgliedern (siehe oben) können für geschlossene Gruppen Zwangsmitgliedschaften eingerichtet werden, z.B. für die disziplinarische Zuordnung von Mitarbeitern zu einem Fachbereich. Außerdem können Mitglieder aus einer Gruppe entfernt werden. Diese Anwendungsfälle sind in Abbildung 3.7 zusammengefasst.

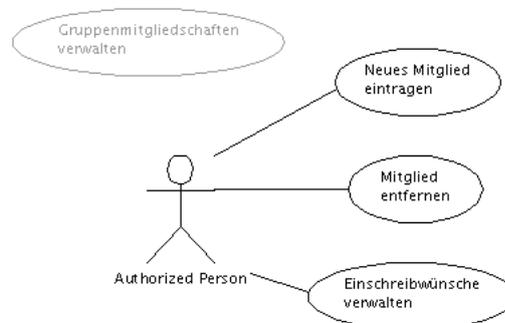


Abbildung 3.7.: Anwendungsfall: Mitgliederverwaltung

Die Zusammenhänge zwischen den einzelnen Vorgängen *Einladen*, *Einschreiben* und *Austragen* sind mit den beteiligten Komponenten und Schlüsselwörtern in Tabelle 3.2 niedergelegt.

Abbildung 3.8 verdeutlicht den Ablauf einer Einschreibung. Je nach Gruppenart (offen oder geschlossen) wird ein Einschreibwunsch sofort genehmigt oder zunächst an einen Gruppenverwalter weitergeleitet. Dieser befindet über die Ein-

Gruppenart	Aktion Person	Aktion Gruppe	typeOfRequest	Komponente
offen	Einschreibung			SubscribeLinkComponent
offen		Einladung		InvitePeopleLinkComponent
offen	Einladung folgen		invitation	FollowInvitationLinkComponent
offen	Einladung ausschlagen			DeclineInvitationLinkComponent
offen	Austragung			UnsubscribeLinkComponent
offen		Zwangsaustragung		UnsubscribeLinkComponent
geschlossen	Einschreibwunsch		subscribe	InvitePeopleLinkComponent
geschlossen		Einschreibung erlauben		FollowInvitationLinkComponent
geschlossen		Einschreibung ablehnen		DeclineInvitationLinkComponent
geschlossen	Austragungswunsch		unsubscribe	InvitePeopleLinkComponent
geschlossen		Austragung erlauben		FollowInvitationLinkComponent
geschlossen		Austragung ablehnen		DeclineInvitationLinkComponent
geschlossen		Zwangseinschreibung		SubscribeLinkComponent
geschlossen		Austragung		UnsubscribeLinkComponent

Tabelle 3.2.: Matrix Gruppenart / Aktion / Komponenten

schreibung und veranlasst entweder eine Ablehnung oder eine Bestätigung der Einschreibung. Eine Austragung wird in gleicher Art gehandhabt.

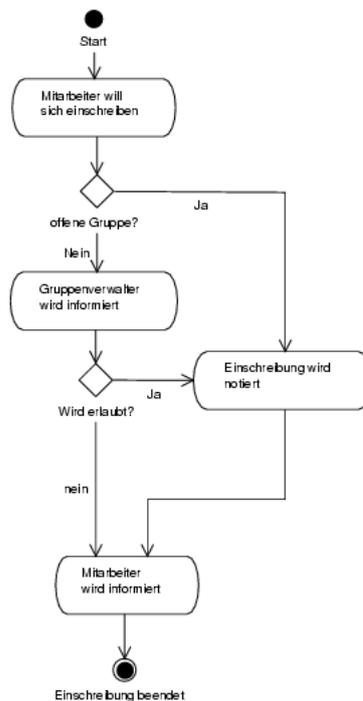


Abbildung 3.8.: Ablauf Einschreibung

### 3.3.3. Datenmodell

Wie bereits in Abschnitt 2.4 erläutert, liegen die von dieser Anwendung zu bearbeitenden Informationen in einer Baumstruktur vor. Drei LDAP-Objektklassen sind hier für *eGroup* von Bedeutung: *awiPerson*, *awiGroup* und *awiPendingMember*.

#### **awiPerson**

*awiPerson* ist direkt von der Klasse *eduPerson* [25] abgeleitet, einer Spezifikation der Internet2 Middleware Initiative. Dieses Schema wurde geschaffen um „die Kommunikation zwischen akademischen Institutionen zu erleichtern“<sup>2</sup>. Es definiert Attribute, welche unter anderem die Zugehörigkeit einer Person zu einer Organisationseinheit (*eduPersonOrgUnitDN*) und ihr Verhältnis zu einer Organisation (*eduPersonAffiliation*) definieren. Das AWI hat diese Objektklasse mit *awiPerson* um einige Attribute erweitert um z.B. die Forschungsgebiete einer Person erfassen zu können.

<sup>2</sup>Original: „EduPerson is an auxiliary object class for campus directories designed to facilitate communication among higher education institutions.“ [25]

Ein `awiPerson`-Eintrag wird im Wesentlichen durch die folgenden Attribute gekennzeichnet (`eduPerson`-Attribute aus [25] entnommen, nähere Details siehe dort):

**eduPersonAffiliation** Spezifiziert das Verhältnis einer Person zu ihrer Organisation in definierten Begriffen, z.B. `staff`, `student` oder `alumn`.

**eduPersonEntitlement** Erlaubt einem Resource Provider die Entscheidung darüber, ob eine Person Zugriff auf eine Ressource erhalten soll oder nicht.

Eine erste Anwendung hierfür ist `eGroup` selbst. Die Administratoren tragen hier den Eintrag

```
urn:awi:awi-bremerhaven.de:application:eGroup:admin:all
```

Weitere Berechtigungen sind natürlich denkbar, wie „darf nur interne Gruppen anlegen“ oder „darf überhaupt keine Änderungen durchführen“. Ein Grundstein für eine feiner granulいたe Rechteverwaltung ist mit dem Interface `AccessRightsHelper` gelegt.

**eduPersonOrgDN** Bezeichnet die Organisation, zu der eine Person gehört.

**eduPersonOrgUnitDN** Nennt die Organisationseinheiten, denen eine Person angehört. Hier trägt `eGroup` die DN's der Fachbereiche und Sektionen ein, aber auch der anderen Gruppen, zu der eine Person gehört.

**eduPersonPrimaryAffiliation** Benennt das primäre Verhältnis einer Person zu ihrer Organisation. Eine Person kann grundsätzlich in mehreren Verhältnissen, z.B. als Student und als Mitarbeiter, zu einer Institution stehen. Hier wird die vorherrschende Art definiert.

**eduPersonPrimaryOrgUnitDN** Ebenso kann eine Person mehreren Gruppen angehören. Hier wird die primäre Zugehörigkeit benannt. Dieses Attribut kann z.B. nach disziplinarischen Gesichtspunkten gesetzt sein.

**eduPersonPrincipalName** Auch *EPPN* oder *NetID*. Dieses Attribut ist für die inter-institutionelle Kommunikation bedeutend. Sein Wert benennt eine Person innerhalb einer Organisation eindeutig. Der Aufbau ist wie bei einer eMail-Adresse `user@organisation.tld`, und kann, muss jedoch nicht, mit der öffentlichen eMail-Adresse des Nutzers identisch sein.

Der *EPPN* kann in anderen Organisationen genutzt werden, um einen externen Mitarbeiter eindeutig zuordnen zu können.

**eduPersonScopedAffiliation** Mit diesem Attribut lässt sich das Verhältnis einer Person zu einer bestimmten Organisation beschreiben, wieder in Begriffen wie `student`, `staff`, `alumn` usw., zusätzlich jedoch mit angehängter Domain, z.B. `also staff@awi-bremerhaven.de`.

**awiPersonExpirationDate** Diese Zeichenkette nach dem Muster JJJJMMTT benennt den Zeitpunkt, zu dem ein Nutzeraccount ausläuft. Nach diesem Zeitpunkt kann eine Person keine Änderungen mehr vornehmen.

Über diese Attribute hinaus wird eine Person im Directory über ihre `userID` – `uid` als Teil des Distinguished Name eindeutig identifiziert. Weitere Merkmale sind der Common Name – `cn` und der `displayName`. Sie ergeben sich aus dem Vererbungsschema für `awiPerson`. Alle Attribute aus `awiPerson` und `eduPerson` sind optionale Angaben. Die beiden Pflichtattribute `cn` und `sn` werden durch die übergeordnete Objektklasse `person` definiert.

### **awiGroup**

`awiGroup` kennt bisher keine mit `eduPerson` vergleichbare offizielle Entsprechung in den internationalen Forschungsnetzen. Vielmehr greifen die Organisationen auf eigene Implementierungen zurück. `awiGroup` stellt Attribute bereit, um die Funktionsträger einer Gruppe und eine Hierarchie innerhalb der Organisation über die Möglichkeiten eines Verzeichnisdienstes hinaus abzubilden. Die Objektklasse wurde Ende 2001 am AWI entwickelt und basiert auf der älteren Klasse `structOrgUnit`, die hier 1999 entstand.

Einige der Attribute enthalten einen *Distinguished Name – DN*, also eine eindeutige Referenz auf einen anderen Eintrag im Directory. Andere können auch einen *eduPersonPrincipalName – EPPN* enthalten. Die meisten Attribute enthalten jedoch einfache Zeichenketten, z.B. zur Ablage des Gruppennamens.

Die wichtigsten Attribute im Rahmen von *eGroup* sind derzeit:

**awiGroupAffiliation** Über dieses Attribut wird die Aufbauorganisation abgebildet. Es enthält *Fachbereich* oder *Sektion*. Diese beiden Begriffe spiegeln jedoch nicht die tatsächliche Bezeichnung wieder, diese wird in `awiGroupDisplayAffiliation` abgelegt.

**awiGroupAncestorDN** Enthält einen DN. Hin und wieder wird die Organisationsstruktur umgestellt, zuletzt mit dem Jahreswechsel 2004/2005. Dieses Attribut nimmt dann den Vorläufer der Gruppe aus der alten Struktur auf.

**awiGroupAuthorizedSigner** Enthält DN oder EPPN. Hier kann ein Unterschriftsberechtigter hinterlegt werden.

**awiGroupDeputy** Enthält DN oder EPPN, derzeit nicht benutzt. Ein Stellvertreter für den Leiter einer Organisationseinheit.

**awiGroupDisplayAffiliation** Dieses Attribut enthält die offizielle Bezeichnung für die Art der Organisationseinheit und wird zweisprachig gepflegt, siehe [47]. Es werden Begriffe wie „Bereich“, „Sektion“, „Fachbereich“, „Topic“ etc. verwendet, die direkt zur Anzeige genutzt werden können.

**awiGroupDisplayName** Diese Zeichenkette wird bei der Anzeige einer Gruppe als Gruppenname verwendet. Sie ist i.d.R. kürzer als `awiGroupName`. Wird vor allem für Listenansichten verwendet und kann Abkürzungen enthalten.

**awiGroupDocmaster** Enthält DN oder EPPN. Dabei handelt es sich um den Publikationsbeauftragten. Dieser muss in *ePic* neue Publikationen freigeben, bevor diese in der Veröffentlichungsübersicht erscheinen.

**awiGroupExternalURL** Enthält eine URL unter der sich die Gruppe nach außen präsentiert.

**awiGroupHead** Enthält DN oder EPPN. Hier wird der Leiter einer Organisationseinheit, z.B. eines Fachbereiches, eingetragen.

**awiGroupInternalURL** Unter dieser URL präsentiert sich die Gruppe im Intranet.

**awiGroupName** Enthält zweisprachig den offiziellen Namen der Gruppe.

**awiGroupPCBeauftragterDN** Enthält einen DN. Als erste Ansprechpartner der Computeranwender fungieren im AWI die PC-Beauftragten. Sie helfen bei kleineren Problemen und stellen den Kontakt zum Rechenzentrum her.

**awiGroupSecretary** Enthält DN oder EPPN. Benennt die Sekretärin einer Organisationseinheit.

**awiGroupSuperiorDN** Enthält einen DN. Über dieses Attribut wird die Hierarchie der Organisation abgebildet, es enthält die hierarchisch übergeordnete Organisationseinheit.

**awiGroupWebmasterDN** Enthält einen DN. Der Webmaster ist für die Webseiten seiner Gruppe verantwortlich.

#### **awiPendingMember**

`awiPendingMember` schließlich dient zur Erfassung von Einladungen, Einschreib- und Austragungswünschen. Sie wurde eigens für *eGroup* definiert.

**awiPendingMemberTimeOfRequest** Enthält einen Zeitstempel vom Zeitpunkt der Erzeugung des Eintrags nach dem Muster YYYYMMDDHHmmSSmsms (Y=Jahr, M=Monat, D=Tag, H=Stunde, m=Minute, S=Sekunde, ms=Millisekunde 4-stellig).

**awiPendingMemberTypeOfRequest** Enthält den Typ der Anforderung. Kann bzw. sollte einer der Werte „invitation“ für Einladungen, „subscribe“ für Eintragungswünsche und „unsubscribe“ für Austragungswünsche sein.

**awiPendingMemberEduPersonOrgUnitDN** Eine Referenz auf die Gruppe, in der eine Mitgliedschaft erzeugt werden soll.

**awiPendingMemberUserDN** Eine Referenz auf die Person, die Mitglied werden soll.

Im AWI-Directory wird die Hierarchie der Organisationseinheiten nicht durch Teilbäume im Verzeichnisdienst abgebildet, sondern über die Attribute `awiGroupSuperiorDN`, `cn` und `awiGroupSerialNumber`. Ersteres enthält, wie beschrieben, die übergeordnete Gruppe oder Einheit, `cn` einen hierarchisch aufgebauten Common Name der Gruppe und das letztgenannte Attribut eine Ziffernfolge, das die Reihenfolge der Organisationseinheiten widerspiegelt. So gehört die Gruppe `awi-2005-0505` „unter“ die Gruppe `awi-2005-05`, die wieder `awi-2005` (dem AWI selbst) untergeordnet ist. Auf diese Weise ist schnell eine sortierte Liste der Aufbauorganisation erstellt.

Diese drei LDAP-Objektklassen werden in *eGroup* durch die Klassen `de.awibremerhaven.egroup.eomodel.AwiPerson`, `de.awibremerhaven.egroup.eomodel.AwiGroup` und `de.awibremerhaven.egroup.eomodel.AwiPendingMember` repräsentiert. Sie wurden einmalig mit Hilfe des *Enterprise Objects Modeler – EOModeler* [13] erzeugt und dann weiter angepasst. Sie erlauben den Zugriff auf die Attributwerte, die das Framework aus dem Directory ausliest. Dabei kümmert sich das Enterprise Objects Framework um die Aktualität der Daten und einen ressourcenschonenden Zugriff. So werden nur die Einträge ausgelesen, die im Moment benötigt werden. Referenzierte Einträge werden erst bei Bedarf abgerufen. Dieses Verhalten wird als *lazy fetching* bezeichnet.

Prinzipiell kann Enterprise Objects auch die direkte Verbindung auf referenzierte Einträge, also andere Gruppen und Personen, herstellen. Dazu wurden im EOModeler many-to-many- bzw. one-to-many-Beziehungen definiert. Allerdings zeigt sich hier eine Schwäche des Frameworks. Als Primärschlüssel eines Datensatzes wird der Relative Distinguished Name angesehen (RDN), ausgehend vom aktuellen Base Distinguished Name (Base-DN). In den Attributen, z.B. `awiGroupHead`, stehen jedoch vollständige Distinguished Names (DN). Hier wird deutlich, dass Enterprise Objects näher an relationalen Datenbanken als an baumartig aufgebauten Verzeichnisdiensten orientiert ist.

Leider ist es nicht gelungen, in der Anwendung die eingelesenen Daten so zu manipulieren, dass entweder Einträge über ihren DN abrufbar sind oder die „Fremdschlüsseleinträge“ in RDNs umgesetzt werden. Somit musste der Ansatz, direkt auf referenzierte Einträge über eine many-to-many-Beziehung zugreifen zu können, fallen gelassen werden. Referenzierte Einträge müssen daher jeweils von der Anwendungslogik selbst aus dem Verzeichnisdienst ausgelesen werden.

Dieser Zugriff soll an einem kleinen Beispiel verdeutlicht werden. Ein Nutzer meldet sich an der Anwendung an. Dabei werden im Hintergrund alle Organisationseinheiten aus dem Directory ausgelesen. Sie stehen damit in der aktuellen Session ständig zur Verfügung. Der Nutzer erhält die Gruppen angezeigt in denen er Mitglied ist und kann weitere Informationen zu diesen Gruppen abrufen.

Nun werden aus den Werten der Attribute `awiGroupHead`, `awiGroupSecretary` usw., welche DNs enthalten, die UserIDs ausgelesen. Aus einem Eintrag `uid=`

mueller,ou=People,dc=awi... wird die UserID mueller. Der so bezeichnete Eintrag wird nun von der Persistenzschicht angefordert. Wurde er in dieser Session bereits verwendet, kann er erneut benutzt werden, andernfalls wird der Eintrag aus dem Directory ausgelesen.

Aus dem ausgelesenen Personeneintrag werden nun Name, Titel, Telefonnummer und weitere Daten in die HTML-Seite geschrieben und diese schließlich dem Browser zur Darstellung übersandt.

Die Umsetzung der DNs in UIDs (bzw. DNs in CNs für Gruppen) ist dem Umstand geschuldet, dass praktisch alle Attribute mehrere Einträge referenzieren können, z.B. mehrere Sekretärinnen. Um die LDAP-Zugriffe möglichst schnell abarbeiten zu können, wird aus mehreren UIDs ein OR-verknüpfter Suchfilter nach dem Muster

```
(&(objectclass=awiPerson)(|(uid=mueller)(uid=schneider)))
```

erzeugt. Es ist also nur eine Suchanfrage für mehrere Einträge notwendig. LDAP-Suchfilter gestatten nicht die Verwendung eines RDNs, daher muss hier die UID genügen. Diese ist jedoch im AWI eindeutig vergeben, daher ist es zulässig, lediglich diesen zur Identifizierung eines konkreten Personeneintrages zu nutzen.

Gleiches gilt für die Organisationseinheiten, die über einen CN eindeutig gekennzeichnet sind. Zur Umsetzung von DN in UID bzw. CN siehe auch Abschnitt [3.4.2.14](#).

## 3.4. Systembeschreibung

Bestimmte zentrale Klassen und Komponenten in *eGroup* enthalten wesentliche Teile der Funktionalität des Systems. Sie werden hier detailliert dargestellt.

Bereits jetzt sei auf die Quellcodes im Anhang [C](#) verwiesen.

### 3.4.1. Packages

Den Java Code Conventions [[33](#), Abschnitt 9] folgend befinden sich alle Klassen unterhalb des Packages `de.awibremerhaven.egroup`, im Folgenden verkürzend als `dae` bezeichnet. Dort sind sie in weitere Packages unterteilt.

Die WebObjects-spezifischen Klassen, also `Main`, `Application`, `Session` und `DirectAction` sind in `dae` untergebracht. Hier findet sich auch die globale Konfigurations-Klasse `Config`, siehe Abschnitt [3.4.2.5](#).

Die Darstellungsschicht, also die für das User Interface notwendigen Klassen, sind im Package `dae.pages` untergebracht. Sie sind ebenfalls durch WebObjects bedingt und enthalten alle notwendigen lokalen Variablen und Methoden, um dem User Interface eine Interaktion mit der Geschäftslogik zu erlauben.

Die HTML-Vorlagen für das User Interface liegen zweisprachig in den Verzeichnissen `English.lproj` und `German.lproj` und direkt im Projekthauptverzeichnis. Sie enthalten neben der üblichen HTML-Auszeichnung `<webobject>`-Tags, die von WebObjects zur Laufzeit interpretiert und mit Inhalt gefüllt werden.

Der Zugriff auf die Persistenzschicht ist über das Interface `dae.eomodel.ModelAccess` möglich. Es deklariert Methoden, um die Datenbasis abzufragen und Daten abzuspeichern. Für den schreibenden Zugriff steht die abstrakte Klasse `JNDIAccess` zur Verfügung.

Im gleichen Package liegen auch die Klassen `AwiGroup`, `AwiPerson` und `AwiPendingMember`, die Einträge des Directories repräsentieren. Sie gestatten den Zugriff auf die Attributwerte dieser Einträge. Wie bereits in Abschnitt 3.3.2 dargestellt, kann leider nicht direkt auf referenzierte Einträge zugegriffen werden.

Die einzelnen Komponenten, aus denen sich die Webseiten zusammensetzen können, liegen im Package `dae.components`. Sie sind nach Möglichkeit für exakt einen einzigen Zweck ausgelegt, z.B. zur Anzeige eines Gruppennamens (`AwiGroupDisplayNameComponent`), zur Berechnung der nächsten freien Seriennummer (`NextSerialNumberComponent`) oder auch zur Anzeige einer Gruppenliste mit Checkboxen vor den Gruppen wie in `ShowGroupsCheckboxComponent`.

In der Abschlussphase der Implementierung wurden Teile der Funktionalität in Interfaces in `dae.components.helpers` deklariert. Dort finden sich auch die Implementierungen dieser Funktionen. Sie sollten keine Abhängigkeiten zu anwendungsspezifischen Klassen aufweisen. So können zukünftig die konkreten Implementierungen in eine eigene Bibliothek ausgelagert und durch andere Anwendungen genutzt werden. Beispielhaft sei auf Abschnitt 3.4.2.12 verwiesen.

Die Zusammenhänge zwischen den Anwendungsschichten und ihren Schnittstellen sind in Abbildung 3.9 verdeutlicht. Die View-Komponente besteht aus von `WComponent` abgeleiteten Klassen in den Packages `dae.pages` und `dae.components`. Die Seiten (in der Abbildung blau gekennzeichnet) greifen dynamisch auf einzelne Komponenten (weiß) zu. Beide Teile nutzen Instanzen der Klasse `dae.Session`, um Werte zwischen den Seiten zu übergeben und auf das Datenmodell zuzugreifen, welches durch das Interface `ModelAccess` zugänglich ist.

Durch die so definierte Schnittstellenklasse `Session` bzw. das Interface `ModelAccess` ist es möglich, Teile der Anwendung neu zu erstellen, ohne die anderen Komponenten zu beeinflussen.

Der prinzipielle Ablauf einer Anfrage an das Directory ist in Abbildung 3.10 am Beispiel der Anzeige der Gruppenzugehörigkeiten dargestellt:

1. Ein Nutzer meldet sich bei *eGroup* an. Dabei wird eine neue `Session`-Instanz erzeugt. Während der Instanzierung von `Session` werden die vorhandenen Gruppen über eine `ModelAccess`-Implementierung aus dem Directory abgerufen und bereits sortiert nach Fachbereichen, Bereichen, Sektionen usw. in einer `Hashtable` abgelegt (Instanzvariable `Session.structuredOrganization`).
2. Der Nutzer wählt im Menü den Punkt „Mitgliedschaften“ aus. Dadurch wird bei `Session` nach dem Namen der Seite gefragt, die die Mitgliedschaften anzeigen kann und aus dem Rückgabewert die nächste Seite `OrgUnitsPage` erzeugt. Der Name dieser Seite ist in der Konfigurationsdatei unter dem Schlüssel `orgUnitsPage` abgelegt, siehe auch Abschnitt 3.4.2.5.

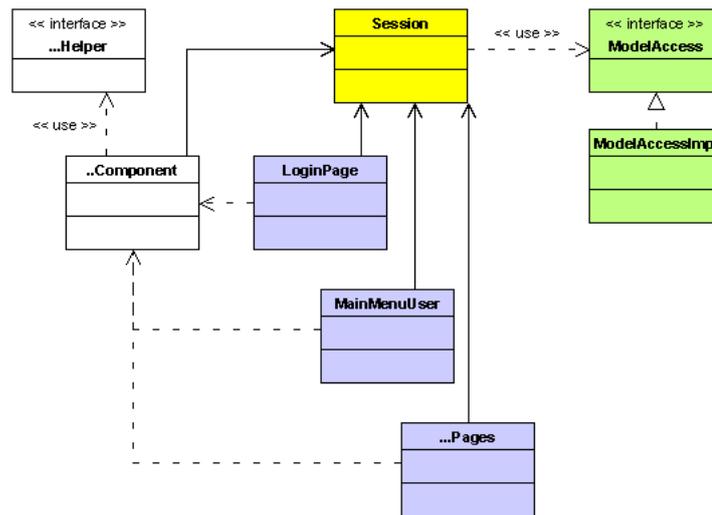


Abbildung 3.9.: Beziehungen zwischen den Komponenten der Anwendung

3. Die neue Seite `OrgUnitsPage` lässt sich nun aus der aktuellen `Session` den angemeldeten Nutzer geben und liest dessen Gruppenreferenzen aus (`AwiPerson.eduPerdonOrgUnitDM()`). Zusätzlich wird der gewünschte Teil der Organisationsstruktur, im Bild diejenigen Gruppen, die Fachbereiche sind, aus der `Session` geholt.
4. Nun wird `Session` veranlasst, aus den Gruppen diejenigen zu extrahieren, in denen der Nutzer Mitglied ist (`Session.getFilteredSubset()`). Zu diesem Zweck wird erneut auf die `ModelAccess`-Implementierung zurückgegriffen.
5. Innerhalb des Seitentemplates für `OrgUnitsPage` werden diese Gruppen in eine `WORepetition` gepackt. Diese `WebObjects`-Klasse iteriert über ein Array und gibt die einzelnen Einträge zurück, im Beispiel also eine Reihe von `AwiGroup`-Instanzen.
6. Der Nutzer erhält am Schluss eine Übersicht präsentiert, in welchen Fachbereichen, Sektionen etc. er explizit eingeschrieben ist.

Der Vorteil dieser Implementierung ist auf der einen Seite der schnelle Zugriff auf die Datenbasis, da alle Gruppen beim Anmelden ausgelesen werden und fortan im Cache der Persistenzschicht (siehe Abschnitt 3.2.3) zur Verfügung stehen. Auf der anderen Seite ergibt sich eine absolut freie Konfigurierbarkeit der Organisationsstruktur. Es bedarf lediglich einiger Einträge in der Konfigurationsdatei und einer Anpassung der darstellenden Komponente, um auf eine neue Struktur umzustellen oder zusätzliche Gruppenarten zu erlauben.

Das betrifft vor allem die Einträge `[type]Name` und `[type]Key` in der Konfigurationsdatei, wobei `[type]` ein Wert aus der Semikolon-separierten Liste `typesOf0us` ist. Näheres siehe auch unter Abschnitt 3.4.2.5.

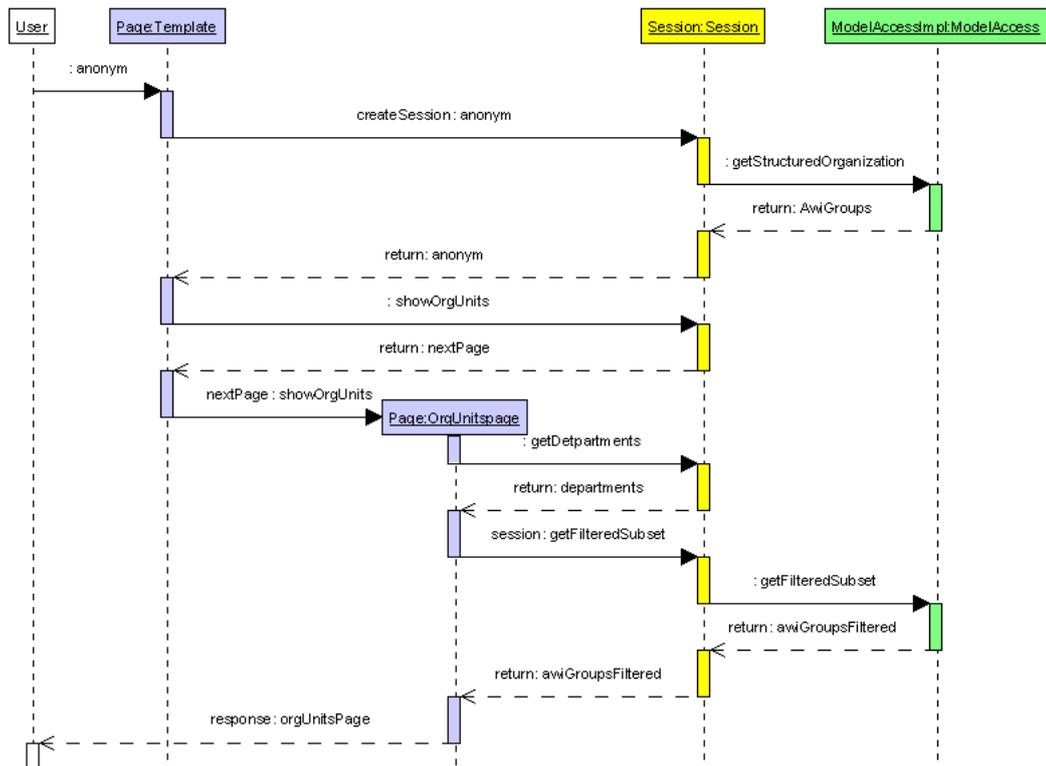


Abbildung 3.10.: Sequenzdiagramm – Anzeige der Gruppenzugehörigkeit

### 3.4.2. Ausgewählte Klassen

Dieser Teil der Arbeit widmet sich en détail einigen zentralen Klassen und Komponenten, die wesentliche Teile der Funktionalität bereitstellen. Aufgrund der Quellcodemenge können leider nicht alle Klassen betrachtet werden, daher kann dieser Abschnitt nur die Grundgedanken der Implementierung vermitteln. Es sei in diesem Zusammenhang ausdrücklich auf die Kommentare in den Quelltexten und die Testklassen hingewiesen.

#### 3.4.2.1. Application

Diese Klasse stellt den Startpunkt einer WebObjects-Anwendung mit der `main()`-Methode zur Verfügung. Hier werden auch einige grundlegende Einstellungen getroffen, z.B. die Verwendung von Cookies anstelle von URL-codierten SessionIDs.

Gerade wenn eine Anwendung auf verschiedenen Plattformen entwickelt wird und Textdateien verarbeitet werden, ist es unbedingt angeraten, ein Encoding der einzelnen Dateien und Klassen explizit festzulegen. Diese Festlegung wird ebenfalls in `Application` getroffen. Die Wahl fiel auf UTF-8 als universelle Zeichenkodierung. Alle Quelldateien, HTML-Vorlagen und Textdateien sind in diesem Encoding erstellt. Auf diese Weise können alle Umlaute ohne Umschreibungen (HTML-Entities) direkt verwendet werden. Bei der Weiterentwicklung sollte diese Einstellung unbedingt berücksichtigt werden. Eine Ausnahme von dieser Regel ist die Konfigurationsdatei `config.properties`. Diese Datei liegt gemäß Sprachspezifikation in ISO-8859-1 vor.

Die für die Verwendung von UTF-8 notwendigen Deklarationen, auch im HTML-Template, sind im Anhang unter Punkt [C.1](#) notiert.

Zu beachten ist ferner, dass WebObjects für jede Komponente auch eine WOO-Datei anlegt, in der das Encoding für diese Komponente festgelegt wird. In der Standardinstallation wird hier das `MacOSRomanEncoding` definiert. Diese Datei kann jedoch problemlos gelöscht werden, vgl. Abschnitt [3.4.2.4](#).

#### 3.4.2.2. ApplicationDisabled

Es ist dem WOBuilder geschuldet, dass alle `import`-Anweisungen so genannte Wildcard-imports sind, also statt

```
1 import java.util.Hashtable;
import com.webobjects.appserver.WOComponent;

muss

1 import java.util.*;
import com.webobjects.appserver.*;
```

verwendet werden. Anderenfalls würde der WOBuilder keinen Zugriff auf die Methoden von benutzten Klassen, wie `NSArray` haben.

Durch die Verschiebung der beiden Klassen `Session` und `Application` aus dem Quellhauptverzeichnis in das Package `de.awibremerhaven.egroup` ist der WOBuilder

auch nicht mehr in der Lage, auf die Variablen und Methoden dieser beiden Klassen zuzugreifen. Daher wurden im Quellhauptverzeichnis zusätzlich zwei Klassen angelegt, die direkt die beiden WebObjects-Klassen erweitern. Siehe hierzu auch den Quelltext im Abschnitt [C.2](#).

Auf der anderen Seite stören sich die Testfälle für JUnit an diesen beiden zusätzlichen Klassen; die Tests brechen unvermittelt mit einer Fehlermeldung ab. Daher werden für die Testläufe diese beiden Klassen durch den Suffix `Disabled` aus dem allgemeinen Kontext entfernt.

Soll also eine Komponente mit dem `WOBuilder` entwickelt werden, sollten diese beiden Klassen wieder umbenannt werden.

#### 3.4.2.3. `AwILdapEntry`

Dieses Interface deklariert als einzige Methode `relativeDistinguishedName():String` und wird von allen Model-Klassen implementiert. Auf diese Weise ist es in `JNDIAccess.removeEntries(NSArray)` möglich, ohne weitere Unterscheidung der konkreten Typen auf den Relative Distinguished Name zuzugreifen und die betreffenden Einträge aus dem Directory zu entfernen.

#### 3.4.2.4. `AwipersonDisplayNameComponent`

Wie alle Komponenten in WebObjects besteht auch diese Komponente aus vier Dateien: der Java-Klasse, einem HTML-Template, einer WOD-Datei, die das Mapping zwischen Platzhaltern im HTML-Template und Methoden der Java-Klasse herstellt, sowie einer API-Datei, welche den Zugriff auf die Eigenschaften und Inhalte der Komponente durch andere Komponenten definiert. Die WOD-Datei ist eine einfach aufgebaute Textdatei, während die API eine XML-Struktur enthält. Eine fünfte Datei mit der Endung `WOO` kann problemlos gelöscht werden.

Für *eGroup* wurden mehrere Komponenten im Package `dae.components` definiert. Sie alle tragen den Suffix `Component`. `AwipersonDisplayNameComponent` wird benutzt, um den Namen einer Person aus mehreren Attributen zusammensetzen. Dafür muss der Komponente eine `Awiperson`-Instanz übergeben und über einen Wahrheitswert bestimmt werden, ob der Nachname vor dem Rufnamen stehen soll.

In Abhängigkeit davon liest die Komponente die Werte für `givenName`, `sn` und `personalTitle` oder `displayName` aus und erzeugt daraus eine Zeichenkette, die im zugehörigen HTML-Template angezeigt wird.

Dadurch wird in den verschiedenen Seiten die immer gleiche Konstruktion des Namens gespart. Sollten Änderungen notwendig werden, können diese an einer zentralen Stelle umgesetzt werden.

#### 3.4.2.5. `Config`

Die Klasse `Config` stellt die übergreifende Konstantenklasse dar und bietet Zugriff auf die in der Datei `resources/config.properties` niedergelegte Konfiguration der

Anwendung. Bei dieser Datei handelt es sich um eine *key-value*-Datei, die mit der `java.util.Properties`-Klasse erstellt wurde.

In der Konfigurationsdatei werden Passwörter für den LDAP-Zugriff, Servernamen, Seitennamen, reguläre Ausdrücke und Mailtexte abgelegt. Mit dieser zentralen Konfiguration folgt *eGroup* dem Prinzip „Konfigurieren statt integrieren“.

Neben einzelnen Konstanten ist es auch möglich, Listen und Tabellen zu erzeugen. Zu beachten ist dabei, dass `Properties.getProperty(String)` ausschließlich Strings verarbeitet, also die Werte ggf. in andere Typen umgewandelt werden müssen.

Um eine Liste zu erzeugen, werden die betreffenden Werte jeweils mit einem Semikolon getrennt eingetragen. Dieser String wird dann der Methode `Config.splitAtSemicolonToList(String):List` übergeben, die daraus eine `ArrayList` mit den einzelnen Strings erzeugt. Ein Beispiel ist die Methode `Config.getCreatePendingMemberFields()`. Sie liefert in einer Liste die Bezeichnungen der LDAP-Attribute zurück, die beim Abspeichern eines `awiPendingMember`-Eintrags berücksichtigt werden sollen.

Um eine Tabelle, genauer gesagt eine `Hashtable` zu erzeugen, werden zwei Dinge benötigt: Eine Liste mit Schlüsselwörtern und eine entsprechende Anzahl weiterer Einträge, welche die Werte zu den Schlüsselwörtern enthalten. Für ein Beispiel siehe `Config.getOrgStructureKeys():Hashtable`. Diese Methode erzeugt zunächst aus dem Wert für `typesOf0us` eine Liste mit Schlüsseln, um dann mit dieser Liste weitere Werte aus der Datei zu lesen. In diesem Fall wird jedem Schlüssel in der Liste der Suffix `Key` angehängt und der so referenzierte Wert aus der Datei gelesen.

Im konkreten Beispiel enthalten die Werte (z.B. `departmentKey`) Zeichenketten, die sich direkt in einem `EOQualifier` verwenden lassen, um die betreffenden Organisationseinheiten aus dem Directory zu lesen. Kommen also zu einem späteren Zeitpunkt weitere Organisationsstrukturen hinzu, müssen lediglich diese Werte in der Konfigurationsdatei angepasst bzw. erweitert werden, um auf die neuen Gruppen zugreifen zu können. Um sie auch darzustellen, muss natürlich auch, wie schon erläutert, die Präsentationsschicht angepasst werden.

Wird eine neue `Session`-Instanz erzeugt, werden auch sämtliche so ausgewählte Organisationseinheiten aus dem Directory abgerufen und in der Instanzvariable `Session.structuredOrganization:Hashtable` abgelegt, mit den erwähnten Schlüsseln und darunter jeweils einem `NSArray` aus `AWiGroups`.

Um auf die Werte zuzugreifen, sind die oben verwendeten Schlüssel in der Konfigurationsdatei zusätzlich noch einmal in einzelnen Werten mit dem Suffix `Name` abgelegt. So kann an beliebiger Stelle in der Anwendungslogik auf Teile der Organisationsstruktur zugegriffen werden, ohne die gesamte Organisation durchsuchen zu müssen.

Soll also eine Liste der Abteilungen ermittelt werden, so ist folgender Aufruf über eine `Session`-Instanz ohne weiteres möglich:

```
NSArray subdivisions = session.getStructuredOrganization().get(
    Config.getSubDivisionName());
```

Eine Besonderheit sind die Texte für eMail-Benachrichtigungen, wie z.B. unter `mailMessageSubscribe`. Hier werden die Platzhalter `##groupEN##` und `##group##` verwendet. Diese Platzhalter werden in einer `NotificationFactoryImpl`-Instanz unter Zuhilfenahme eines regulären Ausdrucks durch den englischen bzw. deutschen Gruppennamen ersetzt.

### 3.4.2.6. DirectAction

In `WebObjects` stellt diese Klasse Funktionalitäten außerhalb einer Session bereit. Hier wurden das An- und Abmelden eines Nutzers an der Anwendung implementiert. Dabei erzeugt `DirectAction.loginAction()` eine neue `Session`-Instanz, prüft die Authentizität des Nutzers und leitet ggf. zum Hauptmenü weiter.

Dagegen zerstört `logoutAction` die aktuelle Session und leitet an die Login-Seite weiter, wo eine Neuansmeldung möglich ist. Zu beachten ist hierbei, dass beim Abmelden alle nicht gespeicherten Änderungen verloren gehen. Daher ist jede Komponente selbst dafür verantwortlich, eine Speicherung veränderter Objekte zu veranlassen. Dazu sollten die Methoden in `Session` verwendet werden, siehe Abschnitt [3.4.2.13](#).

### 3.4.2.7. FunctionalAwiPersonsArrayComponent

Hierbei handelt es sich um ein Beispiel für eine Komponente, die selbst keine Ausgabe erzeugt, sondern lediglich einen Rückgabewert bereitstellt. Sie ist in der Lage, aus einer Gruppe unter einem bestimmten Attribut referenzierte Personeneinträge auszulesen und gibt diese in einem Array zurück.

Diese Komponente erwartet die Übergabe einer `AwiGroup`-Instanz und der Bezeichnung für ein LDAP-Attribut, wie sie vom `EOModeler` verwendet wird, also z.B. `awiGroupDeputy` oder `awiGroupWebMasterDN`. Das `NSArray`, welches die `AwiPerson`-Objekte enthält, kann in einer übergeordneten Komponente oder Seite aus der Variable `awiPersons` abgerufen werden und dort z.B. für eine Listendarstellung verwendet werden.

Dazu wird aus dem bezeichneten Attribut der Gruppe der Inhalt ausgelesen und intern in der Methode `getPeopleArray(NSArray)` verarbeitet. Da in den Feldern der Gruppenfunktionsträger auch EPPNs, siehe [\[25, Abschnitt 8\]](#), eingetragen sein können, findet hier eine Filterung über `Utilities.correctEmailAddress(String):boolean` statt. Die verbleibenden Einträge stellen konventionsgemäß Distinguished Names dar, über die Personeneinträge aus dem Directory geladen werden und über den Rückgabewert von `getAwiPersons():NSArray` zur Verfügung stehen.

### 3.4.2.8. InvitePeoplePage

Diese Seite nutzt einige der selbst erstellten Komponenten, unter anderem die nachstehend erläuterte `IsGroupOpenGroupCondComponent`, siehe Abschnitt [3.4.2.9](#).

Die Seite wird von der Template-Komponente eingefasst, welche den Titelbanner und das Menü zur Verfügung stellt. Im Seitentitel wird durch eine andere Komponente die Bezeichnung der Gruppe zur Verfügung gestellt, analog zum in 3.4.2.4 beschriebenen Verfahren. Danach folgt die Komponente SearchPeopleComponent. Sie blendet ein Suchfeld ein und gibt nach einer erfolgreichen Suche ein Array mit Personeneinträgen an die lokale Variable foundPeople zurück. Diese Verbindung zwischen Ergebnis der Komponentenoperation und der lokalen Variable wird im Binding Inspector hergestellt, im Bild 3.11 rot umrandet.

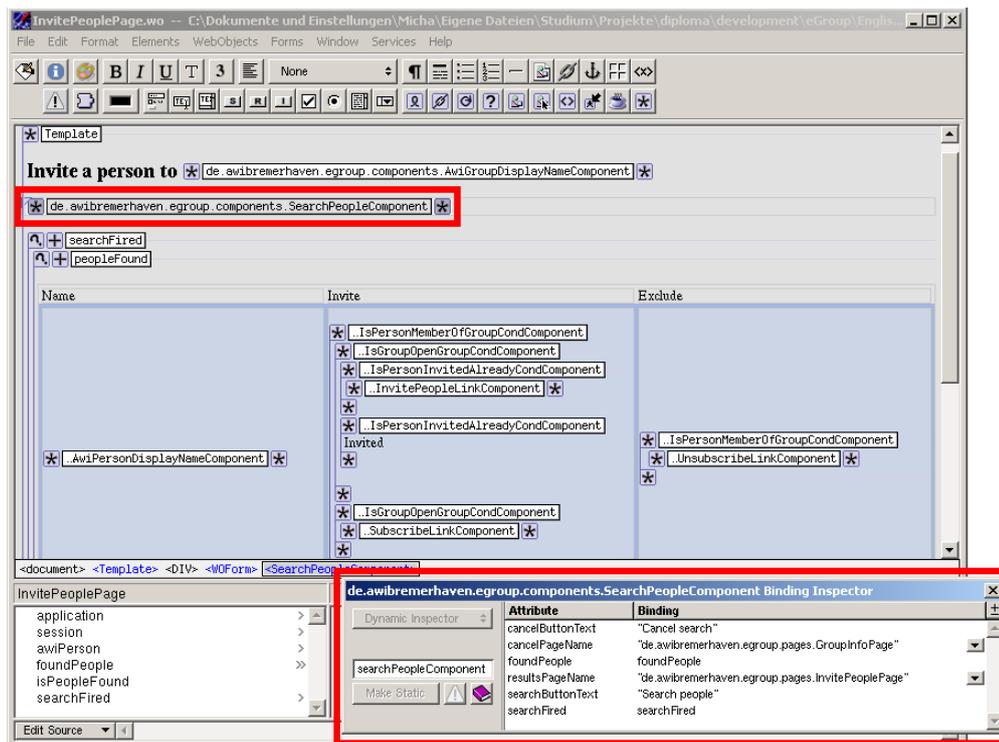


Abbildung 3.11.: InvitePeoplePage im WOBuilder

Über dieses Array wird innerhalb der Seite in einer `WORepetition` iteriert, die in dieser Ansicht nicht dargestellt wird. Für jede Person im Array wird eine Tabellenzeile generiert, die vor allem in der „Invite“-Spalte eine Reihe von konditionalen Komponenten enthält. In Abhängigkeit verschiedener Bedingungen, wie „offene Gruppe“, „Person bereits Mitglied“ und „Person bereits eingeladen“, werden im Frontend verschiedene Inhalte angezeigt, siehe Abbildung 3.12.

Im unteren Bereich der Seite wird eine Fehlermeldung erzeugt, sollten keine Personen gefunden werden.



Abbildung 3.12.: InvitePeoplePage im Frontend

#### 3.4.2.9. IsGroupOpenGroupCondComponent

Die Komponente `IsGroupOpenGroupCondComponent` ist in der Lage, einen Inhalt abhängig vom Ergebnis einer Bedingung und einer gesetzten booleschen Variable anzuzeigen. So wird in der Klasse entschieden, ob eine Gruppe eine offene Gruppe im Sinne der in *eGroup* genutzten Definition ist. Für beide möglichen Entscheidungswege kann der Komponente ein darzustellender Inhalt übergeben werden.

Die in *eGroup* implementierten konditionalen Komponenten verhalten sich ähnlich wie eine `WebObjects`-Komponente vom Typ `WOConditional`, allerdings wird die Negation durch einen zusätzlichen Parameter erreicht, der wahlweise auf `true` oder `false` gesetzt werden muss, z.B. über den `Binding Inspector` in `WOBuilder`.

Gerade in dieser Komponente zeigt sich die große Flexibilität von `WebObjects`. Sollte sich einmal die Definition einer offenen Gruppe grundlegend ändern, muss lediglich diese Komponente angepasst werden.

#### 3.4.2.10. JNDIAccess

Die abstrakte Klasse `JNDIAccess` stellt den schreibenden Zugriff auf das Directory zur Verfügung. Aufgrund der bereits beschriebenen Probleme wurde dieses zweite Zugriffsverfahren notwendig.

Alle Methoden dieser Klasse sind als `protected` deklariert, sie können daher maximal aus dem gleichen Package heraus aufgerufen werden. Um eine Trennung

zwischen Anwendungslogik und Datenbasis sicherzustellen, sollte der Zugriff ausschließlich über eine `ModelAccess`-Implementierung erfolgen.

Die drei schreibenden Methoden sind `insertEntries(NSArray)`, `updateEntries(NSArray)` und `removeEntries(NSArray)`. Sie können auf die anderen Methoden zurückgreifen, um aus den verschiedenen Datenobjekten Instanzen von `javax.naming.Attributes` zu erzeugen, die sich direkt in das Directory schreiben lassen. Dazu können Einstellungen aus der Konfigurationsdatei verwendet werden, um die zu speichernden Felder zu bestimmen, vgl. Abschnitt [3.4.2.5](#).

Die konkrete Implementierung dieser abstrakten Klasse in `JNDIAccessImpl` benutzt z.B. in `createAttributesPendingMember(AwiPendingMember):Attributes` die durch `Config.getCreatePendingMemberFields():List` definierten LDAP-Attribute, um auf die Werte einer `AwiPendingMember`-Instanz zuzugreifen. Siehe hierzu auch den Code-Ausschnitt im Anhang [C.18](#).

In `createAttributesGroup(NSArray):Attributes` findet sich ferner eine Umschreibung der sprachen-abhängigen Attribute, z.B. `awiGroupNameEN`, in die korrekten LDAP-Attribute statt, hier also in `awiGroupName;lang=en` [[47](#)]. Die Umsetzung wird durch die Methode `Utilitites.mapRFC2596Key(String):String` sichergestellt.

#### 3.4.2.11. ModelAccess

`ModelAccess` stellt den Zugang zur Datenbasis zur Verfügung. Es deklariert eine Reihe Methoden, um auf das Directory zuzugreifen.

Die konkrete Implementierung dieser Methoden ist in `ModelAccessImpl` zu finden. Den implementierten Methoden ist gemein, dass sie alle auf die Funktionalität von `ModelAccessImpl.fetchLdap(EOQualifier, entityName):NSArray` zurückgreifen, um Informationen aus dem Directory abzufragen. Diese Methode liefert ein sortiertes Array von `EOEnterpriseObject`-Instanzen zurück, wobei das Sortierkriterium in der Konfigurationsdatei abgelegt ist (Schlüssel `sortCriteria*`).

Um einen Nutzer an der Anwendung zu authentisieren, wird die Methode `isAuthenticationAllowed(String, String):boolean` verwendet. Sie erzeugt probeweise eine `JNDIAccessImpl`-Instanz mit dem Nutzernamen und Passwort des Nutzers. Wird bei diesem Vorgang keine `NamingException` geworfen, ist der Nutzer authentisiert und darf die Anwendung nutzen. Sollte eine Exception geworfen werden, wird der Zugriff verwehrt und eine Fehlermeldung angezeigt.

Für Änderungen am Directory stehen die Methoden `insertEnterpriseObject(EOEnterpriseObject)` und `removeEnterpriseObject(EOEnterpriseObject)` zur Verfügung. Beide Methoden legen das betreffende Objekt im `EditingContext` ab bzw. entfernen es aus diesem und rufen anschließend `storeModifications()` auf, die auch nach Update-Vorgängen aufgerufen werden sollte.

Sie ruft aus dem `EditingContext` die neuen, gelöschten und geänderten Objekte ab und übergibt sie an passende Methoden in eine `JNDIAccess`-Implementierung, siehe Abschnitt [3.4.2.10](#). Abschließend werden alle Objekte invalidisiert, also aus dem Cache entfernt, so dass beim nächsten Zugriff diese Objekte erneut aus dem Directory mit aktuellen Daten abgerufen werden.

Um Änderungen, die noch nicht ins Directory geschrieben wurden, rückgängig machen zu können, wurde die Methode `ModelAccess.revertModifications()` deklariert.

#### 3.4.2.12. `SerialNumberHelper`

Dieses Interface deklariert einige Methoden, um mit den Werten für das LDAP-Attribut `awiGroupSerialNumber` zu arbeiten. So sind Methoden definiert, welche die nächste freie Seriennummer ermitteln. Das ist jeweils von einem bestimmten Gruppentyp aus auf der selben Hierarchieebene oder eine Stufe darunter möglich. Diese Seriennummer wird dann fertig formatiert als Zeichenkette zurückgegeben.

Daneben existiert auch das Interface `CnHelper`, welches eine Methode für die Umwandlung einer formatierten Seriennummer in einen Common Name deklariert.

Eine konkrete Implementierung von `SerialNumberHelper` findet sich in der Klasse `SerialNumberHelperImpl`. Eine Instanzierung dieser Klasse folgt dem üblichen Muster:

```
SerialNumberHelper serialNumber = new SerialNumberHelperImpl(  
    structuredOrganization);
```

Diese Implementierung ermittelt die nächste freie Seriennummer anhand einer existierenden Organisationsstruktur, die ihr über den Konstruktor bekannt gemacht wird. Sollte hier eine andere Lösung notwendig werden, müsste lediglich eine neue Implementierung erstellt werden und die Instanzierung angepasst werden.

#### 3.4.2.13. `Session`

Das HTTP-Protokoll ist ein zustandsloses Protokoll, jede Anfrage steht also für sich isoliert. Es gibt in HTTP keine Möglichkeit, eine Sitzung aufzubauen und wieder zu beenden. Daher muss diese Funktionalität durch die Serveranwendung implementiert werden. Es existieren zwei Möglichkeiten dieses zu erreichen: Die Verwendung von Cookies oder angepassten URLs.

Beide Methoden beruhen auf der Verwendung zufällig generierter Schlüssel, so genannter Session-IDs. Diese werden entweder in einem Cookie auf der Clientseite abgelegt oder in die URL der Webseiten mit eingebettet. Auf diese Weise ist es möglich, eine Sitzung mit einem Nutzer zu beginnen, diese zu einem späteren Zeitpunkt wieder aufzunehmen und natürlich auch zu beenden. In `WebObjects` übernimmt die Klasse `Session` diese Funktionalität. Sie hält sitzungsbezogene Daten vor und gestattet den Zugriff auf die Datenbasis. Hier unterscheidet `WebObjects` sich z.B. von Jakarta Struts, da es dort kein explizites Session-Objekt gibt.

Sobald sich ein Nutzer an der Anwendung anmeldet, wird eine neue Sitzung erzeugt, die durch eine Instanz der Klasse `Session` dargestellt wird. `eGroup` verwendet Cookies, um die Session-ID zu speichern.

In dieser Klasse wird die zu verwendende Sprache ausgewählt. Dazu wird die Methode `getBrowserLanguage` verwendet, die den HTTP-Requestparameter `Accept-Language` auswertet. Standardsprache ist Englisch, es wird auf Deutsch umgeschaltet, wenn der Parameterwert mit „de“ beginnt. Darüber hinaus bietet sie keine tiefergehenden Funktionalitäten an, sondern gestattet lediglich Zugriff auf die Datenbasis und die lokal abgelegten Objekte, wie die Organisationsstruktur und den aktuell angemeldeten Nutzer.

Im Übrigen dient die `Session`-Klasse als Zwischenspeicher zwischen den unterschiedlichen Seiten, so dass diese weitgehend voneinander entkoppelt wurden.

#### 3.4.2.14. Utilities

Für immer wiederkehrende Tätigkeiten wurde die Klasse `Utilities` implementiert. Sie bietet Zugriff auf eine Reihe statischer Methoden, die vor allem Konvertierungen zwischen Common Name und Distinguished Name anbieten.

Darüber hinaus sind verschiedene Prüfmethode vorhanden, die z.B. eine eMail-Adresse erkennen oder das Vokabular für `eduPersonAffiliation` prüfen können. Letzteres wird für `eGroup` derzeit nicht verwendet.

Im Zusammenhang mit den vom `EOModeler` erzeugten Model-Klassen kommt den Methoden `createString(Object):String` und `createNSArray(Object):NSArray` eine besondere Bedeutung zu. Apples `EOModeler` ist nicht in der Lage zu erkennen, ob ein LDAP-Attribut nur einzelne Werte (`single-value`) oder mehrere Werte tragen kann. Tatsächlich gibt das Enterprise Objects Framework einzelne Werte als `Strings`, mehrfache Werte jedoch als `NSArray` zurück, unabhängig von der Definition im LDAP-Schema.

Daher werden diese beiden Methoden benutzt, um einen bestimmten Rückgabewert zu garantieren. Die Erstgenannte forciert die Rückgabe eines `String`s, selbst wenn mehrere Attributwerte gefunden wurden, die Letztgenannte gibt in jedem Fall ein `Array` zurück, das ggf. leer ist. Neben den explizit als `single-value` deklarierten Attributen gibt es im LDAP-Schema einige Attribute, die nur als einfache Attribute sinnvoll verwendet werden können, jedoch nicht entsprechend eingerichtet sind. Diese werden hier auf einen einzelnen `String` reduziert, in dem der erste gefundene Wert verwendet wird und alle weiteren Werte verworfen werden.

Im Einzelnen betrifft das die Attribute `awiGroupAcronym`, `awiGroupAffiliation`, `awiGroupAffiliationStatus`, `awiGroupDisplayAffiliation`, `awiGroupExternalURL`, `awiGroupInternalURL`, `awiGroupMail`, `awiGroupName`, `awiGroupSerialNumber` und `awiGroupVisibility` aus dem `awiGroup`-Schema, sowie `awiPersonExpirationFlag`, `displayName`, `givenName`, `l`, `personalTitle`, `sn` und `title` aus dem `awiPerson`-Schema.

Die Methode `createEOQualifierFromArray(NSArray, String)` erzeugt einen `EOQualifier`, der eine Reihe möglicher Werte in ein und demselben LDAP-Attribut sucht. Sie wird z.B. im Zusammenhang mit dem Auslesen referenzierter Personen in `FunctionalAwiPersonsArrayComponent` verwendet. Dabei geht es darum, alle Leiter, Vertreter usw. einer bestimmten Gruppe zu ermitteln. Das `NSArray`, welches dieser Methode übergeben wird, muss eine Reihe von Distinguished Names

enthalten, der String den Namen eines LDAP-Attributes wie vom EOModeler vergeben. Die Methode konstruiert daraus einen EQLQualifier, der genau diese Personen aus der Datenbasis abfragt. Aus

```
NSArray#1: uid=alice,ou=People,dc=awi-bremerhaven,dc=de  
NSArray#2: uid=bob,ou=People,dc=awi-bremerhaven,dc=de  
String: awiGroupHead
```

erzeugt diese Methode den Suchstring

```
(cn = 'alice') OR (cn = 'bob')
```

Der große Vorteil bei diesem Vorgehen liegt darin, dass auf diese Weise nur ein Suchauftrag an das Directory gegeben werden muss und sofort alle passenden Ergebnisse zurückgeliefert werden, und nicht für jeden referenzierten Eintrag eine eigene Suchanfrage gestartet werden muss.

#### 3.4.3. Einschränkungen

Während der Gespräche mit dem Rechenzentrum wurden viele Ideen für weitere Funktionen entwickelt. Manches wäre mit wenigen Handgriffen umzusetzen, anderes würde den Rahmen dieser Diplomarbeit weit hinter sich lassen.

In loser Reihenfolge hier fünf mögliche Erweiterungen:

**Freies Gruppenanlegen** Es ist in der derzeitigen Version auch für Administratoren nicht möglich, Gruppen ohne irgendwelche Vorgaben anzulegen. Offizielle Gruppen können nur ausgehend von einer übergeordneten Organisationseinheit erstellt werden und auch nur im Rahmen der derzeitigen Organisationsstruktur des AWI.

**Planung** *eGroup* ist derzeit keine Lösung für die Planung einer Aufbauorganisation. Dafür wäre es notwendig, Gruppen frei anzulegen und erst nach und nach in eine Reihenfolge zu bringen. Sobald sich die endgültige Struktur kristallisiert hätte, könnte sie festgeschrieben werden und nachträglich nicht mehr verändert werden.

**Alterung** Eine Alterung, wie in Abschnitt 2.5 erläutert, ist ebenfalls nicht implementiert.

**Mailinglisten** Möchte eine Gruppe eine eigene Mailingliste, so muss diese durch einen Administrator manuell eingerichtet werden. Für diesen Zweck ist eine automatische Generierung der entsprechenden Directory-Einträge mit Verwaltung zusätzlicher Abonnenten denkbar.

**Mehrsprachigkeit** Das Frontend von *eGroup* wurde zweisprachig in deutsch und englisch implementiert. Die Unterscheidung der Sprache wird über den vom Browser gesendeten Accept-Language-Parameter gesteuert. Der Benutzer sollte jedoch die Möglichkeit haben, jederzeit die Sprache wechseln zu können.

## 4. Der Abschluss

Eine Investition in Wissen  
bringt noch immer die besten  
Zinsen.

---

(Benjamin Franklin)

Der am Beginn der Arbeit aufgestellte Zeitplan hat von den drei zur Verfügung stehenden Monaten etwa 5 Wochen für die Implementierung vorgesehen. Dieser Zeitrahmen war für die gestellte Aufgabe ausreichend, wurde jedoch teilweise durch die verschiedenen Probleme in Teilen überstrapaziert. *eGroup* bietet nun eine Basis für zukünftige Erweiterungen, die durch die Architektur auch problemlos möglich sind.

Während dieser 5 Wochen entstanden ca. 6800 Zeilen aktiven Java-Codes in 80 Klassen, zuzüglich der HTML-Templates, Stylesheets, Konfigurationsdateien, Prototypen etc. Viele dieser Zeilen wurden automatisch generiert, tragen aber dennoch zur Komplexität der Codebasis bei. Der ursprünglich gewählte Ansatz des Test Driven Development wurde durch größere Schwierigkeiten mit WebObjects stark belastet. Viele Erkenntnisse konnten erst in der Schlussphase der Entwicklung eingesetzt werden. Dennoch erfüllt *eGroup* die gestellten Anforderungen. Es muss sich nun im praktischen Einsatz bewähren.

### Danksagung

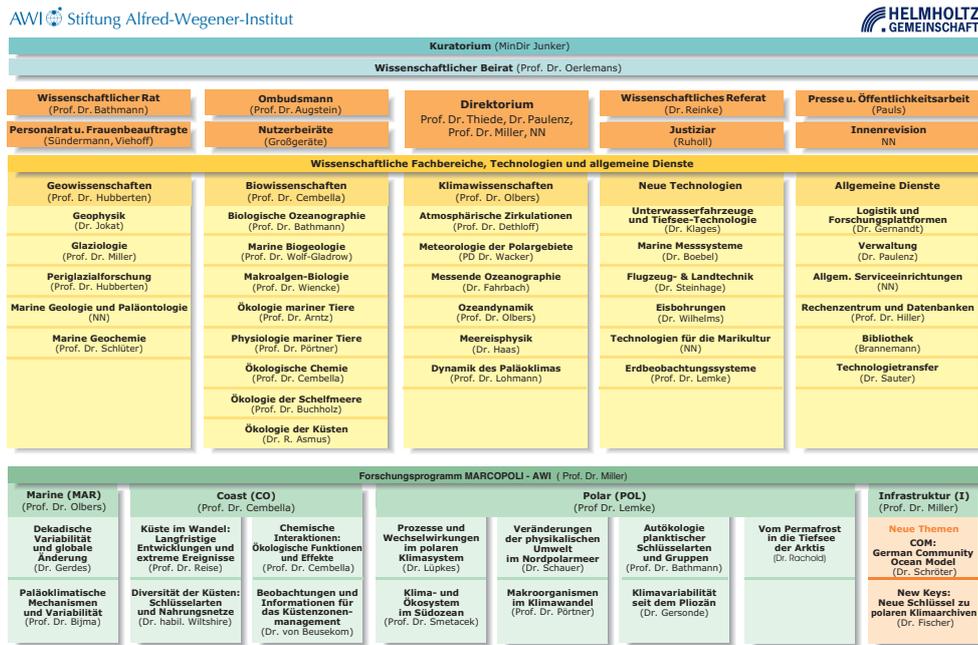
Die Erarbeitung der für mich neuen Technologien, insbesondere die Datenhaltung in einem LDAP-Server und die Arbeit mit WebObjects, sind wertvolle Erfahrungen. Ich habe neue Erkenntnisse gewinnen können und nehme viele Anregungen für zukünftige Projekte mit.

Die Arbeit am Alfred-Wegener-Institut, die Einbindung in ein spannendes und innovatives Umfeld, hat mir viel Freude bereitet.

Ich danke Siegfried Makedanz vom Alfred-Wegener-Institut und Gert Veltink von der Fachhochschule für die ausgezeichnete Betreuung während der Bearbeitung dieser Diplomarbeit. Meinen Korrekturlesern, Anke Battermann, Anja Gerstenberger, Dorothea und Till Conzelmann und meinen Eltern, danke ich für konstruktive Kritik, viele wertvolle Hinweise und mentale Unterstützung. Ohne euch wäre das Ergebnis dieser Arbeit sicher ein anderes.

Mein Dank geht auch an SWITCH [35] für die freundliche Bereitstellung der Abbildungen 2.4 und 2.5.

# A. Organisationsstruktur AWI



Stand Juni 2005

Abbildung A.1.: Organigramm AWI

## B. Organisationsstruktur MARCOPOLI

### MARCOPOLI

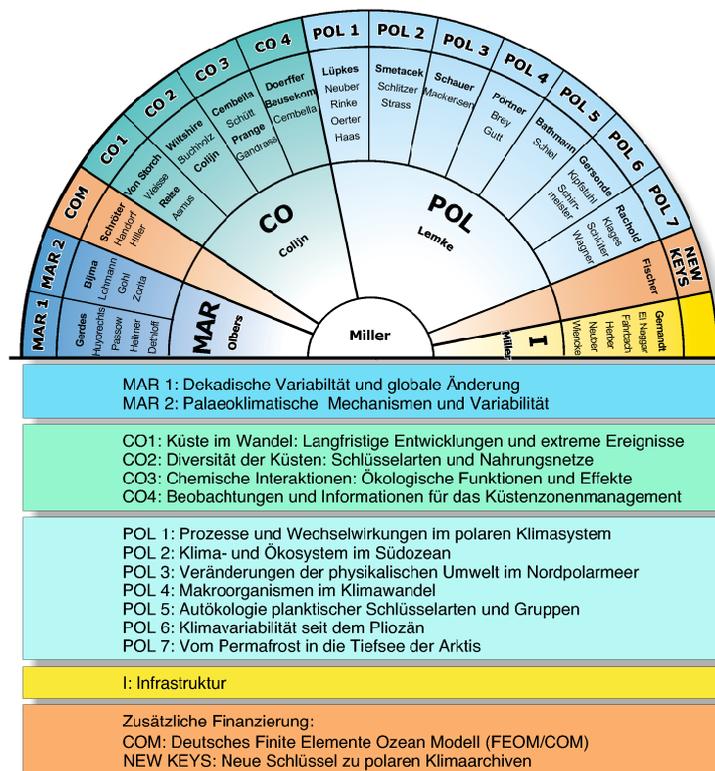


Abbildung B.1.: Organigramm MARCOPOLI

## C. Quellcodes

### C.1. Application

#### C.1.1. Java-Klasse

```

1 public class Application extends W0Application {
    /**
     * The main-method of eGroup. Creates a new Application and sets
     * the
     * URL-encoding to UTF-8.
5     *
     * @param argv
     *     Command-line parameters.
     */
    public static void main(String[] argv) {
10     W0Application.main(argv, Application.class);
        W0Request.setDefaultURLEncoding("UTF-8");
    }

    /**
15     * Main constructor of this application. We force the System-
        setting for
     * file-encoding to be utf-8. WebObjects reads all
        external
     * files, like the HTML-templates and wod-files, with the
        encoding specified
     * by System.getProperty("file.encoding"). This is
        a
     * system-dependent setting, but we have to be
20     * system-independent, since the development was done under
        windows, but the
     * deployment may be on Linux or Mac.
     */
    public Application() {
        super();
25     System.setProperty("file.encoding", "utf-8");
    }

    /**
     * Here we set the output-encoding to utf-8. Every HTTP-response
     * will be
30     * with utf-8.

```

```

32  *
   * @see com.webobjects.appserver.WOApplication#
   *   createResponseInContext(com.webobjects.appserver.WOContext)
   */
34  public WOResponse createResponseInContext(WOContext context) {
35      WOResponse response = super.createResponseInContext(context);
      response.setContentEncoding(_NSUtilities.UTF8StringEncoding);

      return response;
  }
40  /**
   * <p>We are forcing the form-values to be decoded with utf-8.</
   *   p>
   *
   * @see com.webobjects.appserver.WOApplication#dispatchRequest(
   *   com.webobjects.appserver.WORequest)
45  */
   public WOResponse dispatchRequest(WORequest woRequest) {
       String uri = woRequest.uri();
       woRequest.setContentEncoding(_NSUtilities.UTF8StringEncoding);
       woRequest.setFormValueEncodingDetectionEnabled(true);
50      woRequest.setDefaultFormValueEncoding(_NSUtilities.
           UTF8StringEncoding);
   }
}

```

Listing C.1: Application.java (Auszug)

### C.1.2. UTF-8-Deklaration im HTML-Template

```

1  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "
   http://www.w3.org/TR/html4/loose.dtd">
   <html>
     <head>
       <title>eGroups</title>
5      <meta http-equiv="Content-Type" content="text/html; charset=
         UTF-8">
     </head>
     <body>
       ...
     </body>
10 </html>

```

Listing C.2: Template.html (Auszug)

## C.2. ApplicationDisabled

```

1  public class ApplicationDisabled extends de.awibremerhaven.egroup.
   Application {

```

}

Listing C.3: ApplicationDisabled.java

## C.3. AwiLdapEntry

```

1  /**
   * This is the base of all LDAP-entries. It declares a method to
   * identify
   * an entry.
   */
5  public interface AwiLdapEntry {
   /**
    * Returns the relative distinguished name of a LDAP-entry.
    *
    * @return The relative distinguished name
10  */
   public String relativeDistinguishedName();
   }

```

Listing C.4: AwiLdapEntry.java

## C.4. AwiPersonDisplayNameComponent

### C.4.1. HTML-Template

```

1  <WEBOBJECT NAME=displayName></WEBOBJECT>

```

Listing C.5: AwiPersonDisplayNameComponen.html

### C.4.2. Binding

```

1  displayName: W0String {
   value = displayName;
   }

```

Listing C.6: AwiPersonDisplayNameComponen.wod

### C.4.3. Java-Klasse

```

1  /**
   * This component is used to show the complete name of a person.
   * Construct by givenName surName {@link #surNameFirst} <code>==
   * false </code> or
   * surName, personalTitle givenName {@link #surNameFirst} <code>==
   * true </code>.
5  */
   public class AwiPersonDisplayNameComponent extends W0Component {
   private AwiPerson awiPerson;

```

```

private boolean surNameFirst;
10
11 /**
   * Returns the formatted name of the local awiPerson.
   *
   * @returns The formatted name of the local awiPerson.
15 */
public String getDisplayName() {
    String sn = this.getAwiPerson().sn();
    String personalTitle = this.getAwiPerson().personalTitle();
    String givenName = this.getAwiPerson().givenName();
20    String title = this.getAwiPerson().personalTitle();

    StringBuilder builder = new StringBuilder();

    if (this.isSurNameFirst() == true) {
25        builder.append(sn);
        builder.append(", ");

        if (!((personalTitle == null) || (personalTitle.equals("none
            ") || personalTitle
                .equals("null")))) {
30            builder.append(title);
            builder.append(" ");
        }
        builder.append(givenName);
    }
35    else {
        String name = awiPerson.displayName();
        if (title != null) {
            builder.append(title);
            builder.append(" ");
40        }

        if (name != null) {
            builder.append(name);
        }
45    }

    return builder.toString();
}

50 /**
   * Stores the awiPerson whose name should be printed.
   *
   * @param awiPerson The person whose name should be printed.
   */
55 public void setAwiPerson(AwiPerson awiPerson) {
    this.awiPerson = awiPerson;
}

```

```

    }
58
    /**
60     * Set this to true if you'd like to get the
    * surname first, like "Smith, Ph.D. John", or to
    * false if you'd like to get "John Smith".
    *
    * @param surNameFirst true for surname first.
65     */
    public void setSurNameFirst(boolean surNameFirst) {
        this.surNameFirst = surNameFirst;
    }
}

```

Listing C.7: AwiPersonDisplayNameComponen.java (Auszug)

## C.5. Config

```

1 public class Config {
    /**
    * Returns the order that should be used to display the address,
    * phone number etc. of an AwiGroup in the {@link Config#
    * getGroupInfoPage()}.
    *
5     * @return A List containing the value from config.xml#
    * addressPriority.
    */
    public static List getAddressPriority() {
        String property = (String) props.getProperty("addressPriority"
        );
        List prio = splitAtSemicolonToList(property);
10
        return prio;
    }

    /**
15     * Returns the LDAP-username of this application.
    *
    * @return The LDAP-username of this application.
    */
    public static String getApplicationUserName() {
20     return props.getProperty("applicationUserName");
    }

    /**
25     * 

* Returns a map where you find the LDAP-attributes that are
    * used with RFC2596 language extensions (e.g. ldapAttributeId;
    * lang-en).
    *


    * </p>

```

```

* <p> Practically , you'll receive a list of eomodel-used
  attribute ids (awiGroupName and awiGroupNameEN) and the
  corosponding LDAP-attribute ID (awiGroupName;lang-de and
28 * </p>
*
30 * @return A mapping to translate EO-model language-depending
  attributes to LDAP-attributes , according to RFC2596.
* @see de.awibremerhaven.egroup.eomodel.Utilities#mapRFC2596Key
  (String) for implementation example.
*/
33 public static Hashtable getMapRFC2596() {
  Hashtable keys = new Hashtable();
35   List types = Config.getMapRFC2596Keys();
  for (int i = 0; i < types.size(); i++) {
    String type = (String) types.get(i);
    String key = props.getProperty(type);
    keys.put(type, key);
40
    String keyEN = props.getProperty(type + "EN");
    keys.put(type + "EN", keyEN);
  }
  return keys;
45 }

/**
* Returns a table with the types of groups as keys and a <code>
  boolean</code> as value to indicate the possibility of a
  subgroup.
*
50 * @return A Hashtable indicating the possiblities of subgroups.
*/
public static Hashtable getInferiorGroupPossiblities() {
  List keys = getTypesOf0us();
  Hashtable possibilities = new Hashtable(keys.size());
55
  for (int i = 0; i < keys.size(); i++) {
    String key = (String) keys.get(i);
    String possibleString = props.getProperty(key + "
      InferiorPossible");
    Boolean possible = new Boolean(possibleString);
60   possibilities.put(key, possible);
  }

  return possibilities;
}
65 }

```

Listing C.8: Config.java (Auszug)

## C.6. DirectAction

```

1  /**
   * This class could be used to trigger direct actions from pages (
   *   WComponents). See the WebObjects documentaion.
   */
   public class DirectAction extends WODirectAction {
5   private static final NSDictionary noWOSID = new NSDictionary(
       Boolean.FALSE, "wosid");

   /**
    * Terminates the Session and returns the defaultpage.
    *
10  * @return The defaultpage , usually Main.
    */
   public WComponent logoutAction() {
       Session session = (Session)session();
       userLog.info("[ " + session.getUser().uid() + " ]_logged_out");
15
       session.terminate();
       WORedirect mainPage = (WORedirect) pageWithName("WORedirect");
       mainPage.setUrl(context().directActionURLForActionNamed("
           default",
           noWOSID));
20
       return mainPage;
   }

   /**
25  * Registers the user in the session. If she/he is not allowed
       to bind to
       * the Directory we return with this page and an error message (
       using the
       * value of {@link Session#isAuthenticationAllowed(String,
           String)}).
       *
       * @return The next page that should be presented to the user.
30  */
   public WComponent loginAction() {
       log.debug("Processing_loginAction");

       String uid = request().stringFormValueForKey("uid");
35  String password = request().stringFormValueForKey("password");

       boolean isAllowed = ((Session) session()).
           isAuthenticationAllowed(uid,
           password);

```

```

40     log.debug("User_with_uid_is_allowed:" + uid + " " + isAllowed)
        ;
41     userLog.info("[ " + uid + " ]_logged_in");

        if (isAllowed == true) {
            return pageWithName(Config.getUserMainMenuPage());
45     }
        else {
            Main mainPage = (Main) pageWithName("Main");
            mainPage.setAuthenticationAllowed(isAllowed);
            return mainPage;
50     }
    }
}

```

Listing C.9: DirectAction.java (Auszug)

## C.7. FunctionalAwiPersonsArrayComponent

### C.7.0.1. HTML-Template

Da diese Komponente keine Ausgabe erzeugt, ist das HTML-Template leer.

### C.7.1. Binding

```

1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
    <wodefinitions>
        <wo class="FunctionalAwiPersonsArrayComponent"
            wocomponentcontent="false">
5          <binding name="awiGroup"/>
            <binding name="awiPersons"/>
            <binding name="ldapAttributeId"/>

            <validation message="&apos;awiGroup&apos;_is_a_required_
                binding">
10             <unbound name="awiGroup"/>
            </validation>

            <validation message="&apos;awiPersons&apos;_must_be_bound_to_a
                _settable_value">
                <unsettable name="awiPersons"/>
15             </validation>

            <validation message="&apos;ldapAttributeId&apos;_is_a_required
                _binding">
                <unbound name="ldapAttributeId"/>
                </validation>
20     </wo>

```

```
</wodefinitions>
```

Listing C.10: FunctionalAwiPersonsArrayComponent.api

### C.7.2. Java-Klasse

```

1  /**
   * This component returns an array of AwiPersons in {@link #
   *   getAwiPersons()} from a given group, as found in the LDAP
   *   attribute named in {@link #ldapAttributeId}.
   *
   */
5  public class FunctionalAwiPersonsArrayComponent extends
   WComponent {
   /** The group you'd like to get the officials of. */
   private AwiGroup awiGroup;

   /**
10  * An array of awiPersons as found under the attribute named
   * by {@link #ldapAttributeId}.
   * @TypeInfo de.awibremerhaven.egroup.eomodel.AwiPerson
   */
   private NSArray awiPersons;

15  /** The attribute where to find the persons (their DN's). */
   private String ldapAttributeId;

   /**
20  * Returns an array of awiPersons from the stored LDAP attribute
   *
   * @return An array of AwiPersons.
   * @TypeInfo de.awibremerhaven.egroup.eomodel.AwiPerson
   */
25  public NSArray getAwiPersons() {
   log.debug("Returning_persons_from_ldap-attribute");
   String ldapAttributeId = this.getLdapAttributeId();
   Object dnsObject = this.getAwiGroup().valueForKey(
       ldapAttributeId);
   NSArray peopleDns = Utilities.createNSArray(dnsObject);

30  NSArray awiPersons = this.getPeopleArray(peopleDns);

   return awiPersons;
   }

35  /**
   * <p>
   * This method separates references AwiPersons from EPPNs in
   * appropriate ldap-attributes.

```

```

* </p>
40 * <p>
* For example, awiGroupSecretary could contain the three values
   "uid=user,ou=People...", "user@example.com" and "uid=test,
   ou=People...". In this case this method will return you an
   array containing the references AwiPersons (uid=user and uid
   =test), but will also call the {@link #setEppns(NSArray)}
   with the found eppns (user@example.com).
* </p>
* <p> This way you're able to use a WORepetition to iterate
   over the returned array and to iterate over the array as
   returned by {@link #getEppns()}. </p>
*
45 * @param ldapValues An array of values as found in some ldap-
   attribute, like awiGroupHead, that could contain EPPNs.
* @return An array of AwiPersons which are referenced by some
   values and, via {@link #getEppns()} an array of found EPPNs.
*/
48 protected NSArray getPeopleArray(NSArray ldapValues) {
   NSMutableArray dns = new NSMutableArray();
50
   for (int i = 0; i < ldapValues.count(); i++) {
       String entry = ldapValues.objectAtIndex(i).toString();
       if (!Utilities.correctMailAddress(entry) == true) {
           log.debug("Found_DN:_" + entry);
55           dns.addObject(entry);
       }
   }
   EOQualifier qualifier = Utilities.createEOQualifierFromArray(
       dns, Config.getFetchCriteriaPeople());
60   NSArray ldapPersons = session.fetchLdap(qualifier, Config
       .getAwiPersonEntityName());

   return ldapPersons;
   }
65 }

```

Listing C.11: FunctionalAwiPersonsArrayComponent.java (Auszug)

## C.8. InvitePeoplePage

### C.8.1. HTML-Template

```

1 <webobject name="template">
   <div class="userInterface">
       <h2><WEBOBJECT NAME=titleOpenGroupCond>Invite</WEBOBJECT><
           WEBOBJECT NAME=titleClosedGroupCond>Subscribe</WEBOBJECT>
           Colleagues <WEBOBJECT NAME=groupNameComponent></WEBOBJECT></
           h2>
       <!-- Search dialogue -->

```

```

5 <webobject name="searchForm">
6   <webobject name="searchPeopleComponent"></webobject>
</webobject>
<p>
  <!-- displays if search was triggered-->
10 <webobject name="searchFiredCond">
  <!-- displays if people found -->
  <webobject NAME=peopleFoundCond>
    <table>
      <thead>
15        <tr>
          <td class="nameColumn">Name</td>
          <td><WEBOBJECT NAME=tableHeaderOpenGroupCond>
            Invite</WEBOBJECT><WEBOBJECT NAME=
              tableHeaderClosedGroupCond>Subscribe</WEBOBJECT
                ></td>
          <td>Remove</td>
        </tr>
20      </thead>
      <tbody>
        <!-- Iterates over the found people -->
        <webobject name="Repetition1">
          <tr>
25            <td>
              <!-- Shows the name of each single person -->
              <webobject name="awiPersonDisplayNameComponent
                "></webobject>
            </td>
            <td>
30              <!-- If person isn't member of the group shows
                this -->
              <WEBOBJECT NAME=memberOfGroupCondComponent>
                <!-- Open group? -->
                <WEBOBJECT NAME=openGroupCond>
                  <WEBOBJECT NAME=personIsInvitedComponent>
35                    <span class="linkButtons">
                      <WEBOBJECT NAME=
                        invitePeopleLinkComponentComponent>
                      </WEBOBJECT>
                    </span>
                  </WEBOBJECT>
                  <WEBOBJECT NAME=
                    personIsNotInvitedComponent>Invited</
                      WEBOBJECT>
40                </WEBOBJECT>
                <!-- Closed group? -->
                <WEBOBJECT NAME=closedGroupCond>
                  <span class="linkButtons">

```

```

                                <WEBOBJECT NAME=subscribeComponent2></
                                WEBOBJECT>
45                                </span>
                                </WEBOBJECT>
                                </WEBOBJECT>
                                </td>
                                <td>
50                                <!-- Person is member already -->
                                <webobject name="isMemberComponent">
                                    <span class="linkButtons">
                                        <webobject name="unsubscribeComponent"></
                                        webobject>
55                                    </span>
                                    </webobject>
                                </td>
                                </tr>
                                </webobject>
                                </tbody>
60                                </table>
                                </webobject>
                                <!-- No persons were found. -->
                                <WEBOBJECT NAME=peopleNotFound>No persons found.</
                                WEBOBJECT>
                                </webobject>
65                                </p>
                                <p>
                                    <!-- Back link -->
                                    <webobject name="Hyperlink1">Back to groups information page
                                    </webobject>
70                                </p>
                                </div>
                                </webobject>

```

Listing C.12: InvitePeoplePage.html

### C.8.2. Binding

```

1  Hyperlink1: W0Hyperlink {
    pageName = "de.awibremerhaven.egroup.pages.GroupInfoPage";
}

5  Repetition1: W0Repetition {
    item = awiPerson;
    list = foundPeople;
}

10 awiPersonDisplayNameComponent: de.awibremerhaven.egroup.components
    .AwipersonDisplayNameComponent {
    awiPerson = awiPerson;
    surNameFirst = true;
}

```

```

    }
14
15 closedGroupCond: de.awibremerhaven.egroup.components.
    IsGroupOpenGroupCondComponent {
    awiGroup = session.orgUnit;
    useContentIfOpenGroup = false;
    }

20 invitePeopleLinkComponentComponent: de.awibremerhaven.egroup.
    components.InvitePeopleLinkComponent {
    awiGroup = session.orgUnit;
    linkString = "Invite";
    personToInvite = awiPerson;
    user = session.user;
25   typeOfRequest = "invitation";
    }

    isMemberComponent: de.awibremerhaven.egroup.components.
    IsPersonMemberOfGroupCondComponent {
    awiGroup = session.orgUnit;
30   awiPerson = awiPerson;
    useContentIfMember = true;
    }

    memberOfGroupCondComponent: de.awibremerhaven.egroup.components.
    IsPersonMemberOfGroupCondComponent {
35   awiGroup = session.orgUnit;
    awiPerson = awiPerson;
    useContentIfMember = false;
    }

40 openGroupCond: de.awibremerhaven.egroup.components.
    IsGroupOpenGroupCondComponent {
    awiGroup = session.orgUnit;
    useContentIfOpenGroup = true;
    }

45 peopleFoundCond: WOConditional {
    condition = peopleFound;
    }

    personIsInvitedComponent: de.awibremerhaven.egroup.components.
    IsPersonInvitedAlreadyCondComponent {
50   awiGroup = session.orgUnit;
    awiPerson = awiPerson;
    useContentIfAlreadyInvited = false;
    }

```

```

55 personIsNotInvitedComponent: de.awibremerhaven.egroup.components.
    IsPersonInvitedAlreadyCondComponent {
56     awiGroup = session.orgUnit;
        awiPerson = awiPerson;
        useContentIfAlreadyInvited = true;
    }
60 peopleNotFound: WOConditional {
    condition = isPeopleFound;
    negate = true;
    }
65 searchPeopleComponent: de.awibremerhaven.egroup.components.
    SearchPeopleComponent {
        cancelButtonText = "Cancel";
        cancelButtonPageName = "de.awibremerhaven.egroup.pages.GroupInfoPage";
        foundPeople = foundPeople;
70     resultsPageName = "de.awibremerhaven.egroup.pages.
        InvitePeoplePage";
        searchButtonText = "Search";
        searchFired = searchFired;
    }

75 subscribeComponent2: de.awibremerhaven.egroup.components.
    SubscribeLinkComponent {
        awiGroup = session.orgUnit;
        awiPerson = awiPerson;
        linkText = "Add";
    }
80 unsubscribeComponent: de.awibremerhaven.egroup.components.
    UnsubscribeLinkComponent {
        awiGroup = session.orgUnit;
        awiPerson = awiPerson;
        linkText = "Remove";
85 }

```

Listing C.13: InvitePeoplePage.wod (Auszug)

### C.8.3. Java-Klasse

```

1  /**
    * <p>This page is used to invite people to a group. The template
      will insert a search component ({@link SearchPeopleComponent})
      and return the results of the search via {@link #
        getFoundPeople()}.</p>
    *
    * <p>These results will be used by a component that displays the
      found persons , e.g. using a WORepetition-component.</p>
5  */

```

```

public class InvitePeoplePage extends WOComponent {
7   /**
   * Will contain the found persons.
   * @TypeInfo de.awibremerhaven.egroup.eomodel.AwiPerson
10  */
   private NSArray foundPeople;

   /**
   * Creates a new instance of this page.
15  *
   * @param context The context this page resides in.
   */
   public InvitePeoplePage(WOContext context) {
       super(context);
20   this.session = (Session) context.session();
   }

   /**
   * Returns <code>true</code> if the user clicked
25  * the search-button.
   *
   * @return <code>true</code> if the user clicked the search-
       button.
   */
   public boolean getSearchFired() {
30   return searchFired;
   }

   /**
   * Set this to <code>true</code> if the user clicked
35  * the search-button.
   *
   * @param newSearchFired <code>true</code> if the user clicked
       the search-button.
   */
   public void setSearchFired(boolean newSearchFired) {
40   searchFired = newSearchFired;
   }

   /**
   * Returns the found people, aka instances of AwiPerson.
45  *
   * @return The found poeple.
   * @TypeInfo de.awibremerhaven.egroup.eomodel.AwiPerson
   */
   public NSArray getFoundPeople() {
50   return foundPeople;
   }
}

```

```

54  /**
55   * Returns true if some people were found.
56   *
57   * @return true if some people were found.
58   */
59  public boolean isPeopleFound() {
60      if ((this.getFoundPeople() != null)
61          && (this.getFoundPeople().count() > 0)) {
62          return true;
63      }
64      else {
65          return false;
66      }
67  }
68  }

```

Listing C.14: InvitePeoplePage.java (Auszug)

## C.9. IsGroupOpenGroupCondComponent

### C.9.1. HTML-Template

```

1  <WEBOBJECT NAME=contentOpenGroupCond>
2    <WEBOBJECT NAME=isOpenGroupCond>
3      <WEBOBJECT NAME=openContent></WEBOBJECT>
4    </WEBOBJECT>
5  </WEBOBJECT>
6  <WEBOBJECT NAME=contentClosedGroupCond>
7    <WEBOBJECT NAME=isClosedGroupCond>
8      <WEBOBJECT NAME=closedContent></WEBOBJECT>
9    </WEBOBJECT>
10 </WEBOBJECT>

```

Listing C.15: IsGroupOpenGroupCondComponent.html

### C.9.2. Binding

```

1  closedContent: WComponentContent {
2  }
3
4  contentClosedGroupCond: WConditional {
5    condition = useContentIfOpenGroup;
6    negate = true;
7  }
8
9  contentOpenGroupCond: WConditional {
10   condition = useContentIfOpenGroup;
11 }
12
13 isClosedGroupCond: WConditional {

```

```

    condition = isOpenGroup;
15  negate = true;
16  }

    isOpenGroupCond: W0Conditional {
    condition = isOpenGroup;
20  }

    openContent: W0ComponentContent {
    }

```

Listing C.16: IsGroupOpenGroupCondComponent.wod

### C.9.3. Java-Klasse

```

1  /** This component determines whether a given group is open for
    public subscription or not. */
    public class IsGroupOpenGroupCondComponent extends W0Component {
    /** The AwiGroup that should be checked. */
    private AwiGroup awiGroup;
5
    /**
    * Returns <code>>true</code> if the given group is open for
    public subscriptions , <code>>false</code> otherwise .
    *
    * @return <code>>true</code> if the given group is open for
    public subscriptions , <code>>false</code> otherwise .
10  */
    public boolean isOpenGroup() {
    String groupRdn = this.getAwiGroup().relativeDistinguishedName
    ();
    if (groupRdn.contains(Config.getDefaultInternalGroupDnPrefix()
    )) {
    return true;
15  }
    else {
    return false;
    }
    }
20
    /**
    * Sets the AwiGroup.
    *
    * @param awiGroup
    * The awiGroup to set.
25  */
    public void setAwiGroup(AwiGroup awiGroup) {
    this.awiGroup = awiGroup;
    }
30

```

```

32  /**
   * Sets the value of {@link #useContentIfOpenGroup}.
   *
   * @param useContentIfOpenGroup
35  *     The value of {@link #useContentIfOpenGroup}.
   */
37  public void setUseContentIfOpenGroup(boolean
      useContentIfOpenGroup) {
      this.useContentIfOpenGroup = useContentIfOpenGroup;
   }
40 }

```

Listing C.17: IsGroupOpenGroupCondComponent.java (Auszug)

## C.10. JNDIAccess

```

1  /**
   * Provides the write access to the Directory.
   */
public class JNDIAccessImpl extends JNDIAccess {
5  /**
   * <p>Creates Attributes directly from an AwiGroup-instance.
     Uses {@link Config#getCreateInternalGroupGUIFields()} to
     access the fields in the group and to store the values in
     the Attribute-object. </p>
   * <p>Note: This method is suitable not only for internal groups
     , but also for official groups.</p>
   *
   * @param awiGroup An AwiGroup that should be transferred to an
     Attributes-instance.
10  * @return An Attributes-instance , ready for storing in the
     Directory.
   */
protected Attributes createAttributesGroup(final AwiGroup
      awiGroup) {
      Attributes attrsInsert = new BasicAttributes();
      List insertGroupFields = Config.
        getCreateInternalGroupGUIFields();
15
      String dn = getDnGroup(awiGroup);
      String ou = Utilities.convertDnToCn(dn);

      /* Iterate over all fields that might contain updated data */
20  for (int k = 0; k < insertGroupFields.size(); k++) {
      String insertAttribute = (String) insertGroupFields.get(k);
      String languageAttribute = Utilities.mapRFC2596Key(
        insertAttribute);

      Attribute attr = new BasicAttribute(languageAttribute);
25

```

```

        NSArray insertValue = Utilities.createNSArray(awiGroup.
            valueForKey(insertAttribute));
27     if (insertValue.count() > 0) {
        /* Iterate over mutlivalued-attributes */
        for (int l = 0; l < insertValue.count(); l++) {
30         String singleLine = insertValue.objectAtIndex(l).
            toString();
            attr.add(singleLine);
        }

        attrsInsert.put(attr);
35     }
}

Attribute objectClasses = new BasicAttribute("objectClass");
objectClasses.add(0, "awiGroup");
40 objectClasses.add(0, "groupOfUniqueNames");
objectClasses.add(0, "organizationalUnit");
objectClasses.add(0, "top");
attrsInsert.put(objectClasses);

45 Attribute ouAttr = new BasicAttribute("ou");
ouAttr.add(0, ou);

attrsInsert.put(ouAttr);
return attrsInsert;
50 }

/**
 * This method transfers an {@link AwiPendingMember}-instance
 * into an
 * Attributes-instance.
55 *
 * @param pendingMember The AwiPendingMember-entry that shall be
 * stored in the Directory.
 * @return Attributes, built from the AwiPendingMember-instance.
 */
protected Attributes createAttributesPendingMember(final
    AwiPendingMember pendingMember) {
60 Attributes attrsInsert = new BasicAttributes();
List insertPendingMemberFields = Config.
    getCreatePendingMemberFields();
String dn = getDnPendingMember(pendingMember);

for (int i = 0; i < insertPendingMemberFields.size(); i++) {
65 String insertAttribute = (String) insertPendingMemberFields.
    get(i);
Attribute attr = new BasicAttribute(insertAttribute);

```

```

        String value = Utilities.createString(pendingMember.
            valueForKey(insertAttribute));
68
        attr.add(value);
70        attrsInsert.put(attr);
    }

    Attribute objectClass = new BasicAttribute("objectClass");
    objectClass.add("awiPendingMember");
75
    attrsInsert.put(objectClass);

    return attrsInsert;
}
80
/**
 * Creates the context by using the given dn and password. You
 * may pass
 * empty Strings to create an anonymous bind.
 *
85 * @param dn The DN that will be used for the binding.
 * @param password The password of the user.
 * @return An connection to the LDAP.
 * @throws AuthenticationException If the given dn is not
 *         allowed to bind.
 * @throws NamingException If something is wrong with the
 *         connection.
90 */
protected InitialDirContext createLDAPContext(String dn, String
    password)
    throws AuthenticationException, NamingException {
    Hashtable ldapEnvironment = this.getLdapEnvironment();

95    ldapEnvironment.put(Context.SECURITY_PRINCIPAL, dn);
    ldapEnvironment.put(Context.SECURITY_CREDENTIALS, password);

    InitialDirContext ctx = new InitialDirContext(ldapEnvironment)
        ;

100    return ctx;
}

/**
 * <p>Inserts the set of given {@link de.awibremerhaven.egroup.
 * eomodel.AwiGroup}s into the directory.</p>
105 * <p> It will process all attributes that are defined in {@link
 * Config#getCreateInternalGroupGUIFields()} and will also map
 * language–depending attributes to RFC2596–compliant
 * attributes , using {@link Utilities#mapRFC2596Key(String)}.

```

```

    </p>
106     *
    * @param insertedEntries An array of AwiGroups that should be
      inserted.
    * @throws NamingException If LDAP-operation fails.
    */
110 protected void insertEntries(final NSArray insertedEntries)
      throws NamingException {

    /* Iterate over all entries that should be updated */
    for (int i = 0; i < insertedEntries.count(); i++) {
      Attributes attrsInsert = new BasicAttributes();
115      String dn = null;

      if (insertedEntries.objectAtIndex(i) instanceof AwiGroup) {
        AwiGroup awiGroup = (AwiGroup) insertedEntries.
          objectAtIndex(i);
        attrsInsert = createAttributesGroup(awiGroup);
120        dn = getDnGroup(awiGroup);
      }
      else if (insertedEntries.objectAtIndex(i) instanceof
        AwiPendingMember) {
        AwiPendingMember pendingMember = (AwiPendingMember)
          insertedEntries.objectAtIndex(i);
        attrsInsert = createAttributesPendingMember(pendingMember)
          ;
125        dn = getDnPendingMember(pendingMember);
      }

      ctx.createSubcontext(dn, attrsInsert);
    }
130 }

/** Modifies an awiGroup-entry by using the fields named in {
    @link Config#getModifyGroupFields() }.
    *
    * @param awiGroup The group to be modified.
135 * @return Attributes that can be bound to the Directory.
    */
protected Attributes modifyAttributesGroup(AwiGroup awiGroup) {
  Attributes attrsUpdate = new BasicAttributes();

140  List modifyGroupFields = Config.getModifyGroupFields();

  /* Iterate over all fields that might contain updated data */
  for (int k = 0; k < modifyGroupFields.size(); k++) {
    String modifyAttribute = (String)modifyGroupFields.get(k);
145    String languageAttribute = Utilities.mapRFC2596Key(
      modifyAttribute);
  }
}

```

```

147     Attribute attr = new BasicAttribute(languageAttribute);

        NSArray modifyValue = Utilities.createNSArray(awiGroup.
            valueForKey(modifyAttribute));
150     for (int l = 0; l < modifyValue.count(); l++) {
        String singleLine = modifyValue.objectAtIndex(l).toString
            ();
        attr.add(singleLine);
    }

155     attrsUpdate.put(attr);
    }
    return attrsUpdate;
}

160 /**
    * Modifies an awiPerson-entry by using the fields named in {
    * @link Config#getModifyPeopleFields() }.
    *
    * @param awiPerson The person to be modified.
    * @return Attributes that could be bound to the Directory.
165 */

protected Attributes modifyAttributesPerson(AwiPerson awiPerson)
{
    Attributes attrsUpdate = new BasicAttributes();
    List modifyPersonFields = Config.getModifyPeopleFields();
170     for (int i = 0; i < modifyPersonFields.size(); i++) {
        String modifyAttribute = (String) modifyPersonFields.get(i);
        Attribute attr = new BasicAttribute(modifyAttribute);
        NSArray modifyValue = Utilities.createNSArray(awiPerson.
            valueForKey(modifyAttribute));

175     for (int l = 0; l < modifyValue.count(); l++) {
        String singleLine = modifyValue.objectAtIndex(l).toString
            ();
        attr.add(singleLine);
    }

180     attrsUpdate.put(attr);
    }
    return attrsUpdate;
}

185 /**
    * Removes the given AwiLdapEntries from the directory.
    *

```

```

    * @param deleted An array of objects implementing the {@link
      AwiLdapEntry}-Interface.
190  * @throws NamingException If the removal fails.
    */
192  protected void removeEntries(final NSArray deleted) throws
      NamingException {
      for (int i = 0; i < deleted.count(); i++) {
          AwiLdapEntry entry = (AwiLdapEntry) deleted.objectAtIndex(i)
              ;
195      String rdn = entry.relativeDistinguishedName();
          String dn = Utilities.createDnFromRdn(rdn);
          ctx.unbind(dn);
      }
  }
200  /**
    * <p>Updates the given array of {@link AwiGroup} in the
      directory. Iterates over the array and updates all
      attributes that are named by {@link Config#
      getModifyGroupFields()}.</p>
    * <p>It will map language-dependent attributes, using {@link
      Utilities#mapRFC2596Key(String)} to RFC2596 compliant
      attributes, e.g. awiGroupNameEN to awiGroupName;lang-en.</p>
    *
205  * @param updatedEntries An array of AwiGroups that should be
      updated.
    * @throws NamingException If something fails during update.
    */
  protected void updateEntries(final NSArray updatedEntries)
      throws NamingException {
      /* Iterate over all entries that should be updated */
210  for (int i = 0; i < updatedEntries.count(); i++) {
          String dn = null;
          Attributes attrsUpdate = null;

          if (updatedEntries.objectAtIndex(i) instanceof AwiGroup) {
215      AwiGroup awiGroup = (AwiGroup) updatedEntries.
              objectAtIndex(i);
              dn = Utilities.createDnFromRdn(awiGroup.
                  relativeDistinguishedName());
              attrsUpdate = modifyAttributesGroup(awiGroup);
          }
          else if (updatedEntries.objectAtIndex(i) instanceof
              AwiPerson) {
220      AwiPerson awiPerson = (AwiPerson) updatedEntries.
              objectAtIndex(i);
              dn = Utilities.createDnFromRdn(awiPerson.
                  relativeDistinguishedName());
              attrsUpdate = modifyAttributesPerson(awiPerson);
          }
      }
  }

```

```

    }
224
225    try {
        ctx.modifyAttributes(dn, DirContext.REPLACE_ATTRIBUTE,
            attrsUpdate);
    }
    catch (NoSuchAttributeException exc) {
        /* Skipping non-set attribute */
230    }
    }
}
}

```

Listing C.18: JNDIAccessImpl.java (Auszug)

## C.11. ModelAccess

```

1  /**
   * This is your access to the Directory. It provides methods for
   * fetching data.
   */
public class ModelAccessImpl implements ModelAccess {
5  /** The database context used to access the data. */
    private EOEditingContext editingContext;

    /**
     * Creates a new ModelAccessImpl using the given
     * EOEditingContext. This will be
10    * an anonymous bind. For a named bind you have to modify the
     * EOModel.
     *
     * @param editingContext The EOEditingContext, as found in a
     * WOSession.
     * @throws AuthenticationException If wrong credentials were
     * given.
     * @throws NamingException If something is wrong with the LDAP-
     * connection.
15    */
    public ModelAccessImpl(EOEditingContext editingContext)
        throws AuthenticationException, NamingException {
        this.editingContext = editingContext;

20    jndi = new JNDIAccessImpl(Config.getApplicationUserName(),
        Config.getApplicationUserPassword());
    }

    /**
     * @see de.awibremerhaven.egroup.eomodel.ModelAccess#
     * closeConnection()
25    */
}

```

```

public void closeConnection() {
27     try {
        jndi.close();
    }
30     catch (NamingException exc) {
        jndi = null;
    }
    try {
        editingContext.reset();
35     }
    catch (ArrayIndexOutOfBoundsException exc) {
        log.warn("Error_at_closing_connection:_ " + exc);
    }
}
40
/**
 * @see de.awibremerhaven.egroup.eomodel.ModelAccess#fetchLdap(
 *     com.webobjects.eocontrol.EOQualifier, java.lang.String)
 */
public NSArray fetchLdap(EOQualifier qualifier, String
    entityName) throws IllegalArgumentException {
45     if (qualifier != null) {
        NSArray array = new NSMutableArray();
        NSArray sortedArray = new NSMutableArray();
        EOFetchSpecification fetchSpec = new EOFetchSpecification(
            entityName, qualifier, null);

50     try {
        /* Fetches the entries */
        array = editingContext.objectsWithFetchSpecification(
            fetchSpec);

        EOSortOrdering sortOrder = null;
55     /* SortOrderung for AwiGroups */
        if (entityName.equals(Config.getAwiGroupEntityName())) {
            sortOrder = EOSortOrdering.sortOrderingWithKey(Config.
                getSortCriteriaGroups(), EOSortOrdering.
                    CompareCaseInsensitiveAscending);
        }

60     /* SortOrderung for AwiPeople */
        else if (entityName.equals(Config.getAwiPersonEntityName(
            ))) {
            sortOrder = EOSortOrdering.sortOrderingWithKey(Config.
                getSortCriteriaPeople(), EOSortOrdering.
                    CompareCaseInsensitiveAscending);
        }

65     else if (entityName.equals(Config

```

```

        .getAwiPendingMemberEntityName())) {
67         sortOrder = EOSortOrdering.sortOrderingWithKey(Config.
            getSortCriteriaPendingMembers(), EOSortOrdering.
                CompareCaseInsensitiveAscending);
        }

70         /* Sort now... */
        sortedArray = EOSortOrdering.sortedArrayUsingKeyOrderArray
            (array, new NSArray(sortOrder));
        }
        catch (IllegalArgumentException exc) {
            throw new IllegalArgumentException("You have to specify '
75             + Config.getAwiGroupEntityName() + "'_or_'
                + Config.getAwiPersonEntityName()
                + "'_for_the_parameter_'entityName'!");
        }

80         return sortedArray;
        }
        else {
            return null;
        }
85     }

    /**
     * @see de.awibremerhaven.egroup.eomodel.ModelAccess#
        getAwiGroupByCn(java.lang.String)
90     */
    public AwiGroup getAwiGroupByCn(String cn) throws
        NamingException {
        NSArray groups = this.fetchLdap("cn='" + cn + "'", Config.
            getAwiGroupEntityName());
        if (groups.count() != 1) {
            throw new NamingException(groups.count() + "_AwiGroup_with_
                the_cn_" + cn + "_found,_but_expected_to_find_exactly_1_
                AwiGroup.");
95         }
        else {
            AwiGroup group = (AwiGroup) groups.objectAtIndex(0);
            return group;
        }
100    }

    /**
     * @see de.awibremerhaven.egroup.eomodel.ModelAccess#
        getAwiPersonByUid(java.lang.String)
    */

```

```

105 public AwiPerson getAwiPersonById(String uid) throws
      NamingException {
106     NSArray persons = this.fetchLdap("uid='" + uid + "'", Config.
          getAwiPersonEntityName());
      if (persons.count() != 1) {
          throw new NamingException(persons.count()
              + "_AwiPersons_with_the_uid_" + uid
110         + "_found,_but_expected_to_find_only_1_AwiPerson.");
      }
      else {
          AwiPerson person = (AwiPerson) persons.objectAtIndex(0);
          return person;
115     }
  }

  /**
   * @see de.awibremerhaven.egroup.eomodel.ModelAccess#
       getFilteredSubset(com.webobjects.foundation.NSArray, com.
       webobjects.foundation.NSArray)
120  */
  public NSArray getFilteredSubset(NSArray subsetOfGroups, NSArray
      rdns) {
      int size = rdns.count();
      StringBuffer qualifierString = new StringBuffer();
      for (int i = 0; i < size; i++) {
125         qualifierString.append("relativeDistinguishedName_like_%s");
          if (i < size - 1) {
              qualifierString.append("_OR_");
          }
      }
130
      EOQualifier qualifier = EOQualifier.
          qualifierWithQualifierFormat(qualifierString.toString(),
              rdns);
      NSArray filteredSubset = EOQualifier.
          filteredArrayWithQualifier(subsetOfGroups, qualifier);

      return filteredSubset;
135  }

  /**
   * @see de.awibremerhaven.egroup.eomodel.ModelAccess#
       getStructuredOrganization()
   */
140 public Hashtable getStructuredOrganization() {
      Hashtable organization = new Hashtable();

      Hashtable orgStructureKeys = Config.getOrgStructureKeys();
      Enumeration names = orgStructureKeys.keys();

```

```

145
146     while (names.hasMoreElements()) {
        String name = (String) names.nextElement();
        String orgKey = (String) orgStructureKeys.get(name);

150         NSArray subStructure = this.fetchLdap(orgKey, Config.
            getAwiGroupName());
        log.debug("For_" + name + "_I've_found_these_qty:" +
            subStructure.count());
        organization.put(name, subStructure);
    }
    return organization;
155 }

/**
 * @see de.awibremerhaven.egroup.eomodel.ModelAccess#
 *     insertEnterpriseObject(com.webobjects.eocontrol.
 *     EOEnterpriseObject)
 */
160 public void insertEnterpriseObject(EOEnterpriseObject
    insertObject) throws NamingException {
    editingContext.insertObject(insertObject);
    this.storeModifications();
}

165 /* (non-Javadoc)
 * @see de.awibremerhaven.egroup.eomodel.ModelAccess#
 *     isAuthenticatedAllowed(java.lang.String, java.lang.String)
 */
public boolean isAuthenticatedAllowed(String uid, String
    password) {
    boolean allowed = false;
170     try {
        JNDIAccess tried = new JNDIAccessImpl(Utilities.
            createDnFromUid(uid), password);
        allowed = true;
        tried.close();
    }
175     catch (NamingException exc) {
        allowed = false;
    }

    return allowed;
180 }

/**
 * @see de.awibremerhaven.egroup.eomodel.ModelAccess#
 *     removeEnterpriseObject(com.webobjects.eocontrol.
 *     EOEnterpriseObject)

```

```

    */
185 public void removeEnterpriseObject(EOEnterpriseObject
        removeObject) throws NamingException {
        editingContext.deleteObject(removeObject);
        this.storeModifications();

    }

190 /**
    * @see de.awibremerhaven.egroup.eomodel.ModelAccess#
        revertModifications()
    */
    public void revertModifications() {
195     log.debug("Reverting Changes");
        editingContext.revert();
    }

200 /**
    * @see de.awibremerhaven.egroup.eomodel.ModelAccess#
        storeModifications()
    */
    public void storeModifications() throws NamingException {
205     NSArray updated = editingContext.updatedObjects();
        NSArray inserted = editingContext.insertedObjects();
        NSArray deleted = editingContext.deletedObjects();

        if ((updated != null) && (updated.count() > 0)) {
210             jndi.updateEntries(updated);
        }
        if ((inserted != null) && (inserted.count() > 0)) {
            jndi.insertEntries(inserted);
        }
        if ((deleted != null) && (deleted.count() > 0)) {
215             jndi.removeEntries(deleted);
        }

        editingContext.invalidateAllObjects();
    }
220 }

```

Listing C.19: ModelAccessImpl.java (Auszug)

## C.12. SerialNumberHelper

```

1 /**
    * This helper declares methods to deal with
        awiGroupSerialNumbers.
    */
    public interface SerialNumberHelper {

```

```

5
6  /**
   * This method returns the next free serial number below the
   *   given serialNumber, or, according to 
   *   typeOfGroup at the same level.
   *
   * @param serialNumber An existing serial number, from a group
   *   at the same level or above the new group.
10  * @param typeOfGroup The type of the group you need a new
   *   serial number for.
   * @return The next free serial number.
   */
public abstract int getNextFreeSerialNumber(final int
    serialNumber, final String typeOfGroup);

15  /**
   * Returns the next serialNumber below the given group.
   *
   * @return Returns the nextSerialNumberInferior.
   */
20  public abstract int getNextFreeInferiorSerialNumber(final
    AwiGroup superiorGroup, final String typeOfGroup);

   /**
   * Returns the next free serialNumber at the same level.
   *
   * @return Returns the nextSerialNumberSameLevel.
25  */
public abstract int getNextFreeSameLevelSerialNumber(final
    AwiGroup awiGroup, final String typeOfGroup);

   /**
30  * Formats a serialNumber to fit the given type of group.
   *
   * @param typeOfGroup The type of group, see {@link Config#
   *   getSerialNumberTypes()}.
   * @param newSerial The number you'd like to format, e.g. 60101.
   * @return A formatted serialNumber, e.g. 006.01.01
35  */
public abstract String formatSerialNumber(final int serialNumber
    , String typeOfGroup);
}

```

Listing C.20: SerialNumberHelper.java (Auszug)

## C.13. Session

```

1  /**
   * <p>Holds an application-Session, the emulation of a HTTP-
   *   session.</p>

```

```

* <p>It provide access to the model, the business logic and is
  repsonsible for passing objects from one page to another.</p>
4 * <p>Every page is adviced to ask this class for the next page.
  This enables you to replace a page by simple returning another
  page (or, more general, return another page's name. String
  are much better to test than WOComponents...)</p>
5 * <p>If you're interested in automatic testing see {@link de.
  awibremerhaven.egroup.SessionTest} for simulating a Session.</
  p>
  */
7 public class Session extends WOSession {
  /**
   * Determines wether a user is allowed to use this application
   or not. It's used to display the menu.
10  */
  private boolean isAllowed;

  /** The access to the database-package facade-class. */
  private ModelAccess model;
15

  /**
   * Contains <strong>all </strong> awiGroups, categorized
   following the definitions in config.xml.
   */
  private Hashtable structuredOrganization;
20

  /** The actual user that created this session. */
  private AwiPerson user;

  /**
25  * <p>Creates a new Session-object.</p>
   * <p>Along the way it initializes the model-layer (creates the
   access to theDirectory).</p>
   * <p>Finally it reads the whole organization structure from the
   Directory ({@link ModelAccessImpl#getStructuredOrganization
   ()}).</p>
   *
   * @throws NamingException If something is wrong with the
   connection. This error is not repairable, so it may stops
   the application.
30  * @throws AuthenticationException If invalid credentials were
   used.
   */
  public Session() throws NamingException, AuthenticationException
  {
    super();

35    model = new ModelAccessImpl(this.defaultEditingContext());
    structuredOrganization = model.getStructuredOrganization();
  }

```

```

38     this.setStoresIDsInCookies(true);
        this.setStoresIDsInURLs(false);
40     }

    /**
     * Within this method we set the language the pages will be
     * displayed in.
     *
     * @see com.webobjects.appserver.WOSession#awake()
     */
45     public void awake() {
        super.awake();
        this.setLanguages(new NSArray(this.getBrowserLanguage()));
50     }

    /**
     * This method checks the browser languages. If it contains "
     * German" it returns "German", otherwise "English".
     *
     * @return <code>German</code> or <code>English</code>,
     *         depending on the Accept-Language-field in the HTTP-request.
     *         Defaults to English.
     */
55     private String getBrowserLanguage() {
        String browserLanguage = new String("English");

60         NSArray languages = context().request().browserLanguages();

        if (languages.containsObject("German")) {
            browserLanguage = "German";
        }

65         log.debug("Browser_wishes_this_language:_" + browserLanguage);

        return browserLanguage;
    }

70     /**
     * <p>Returns the Hashtable containing the categorized
     * organizational units.</p>
     *
     * <p>You are adviced to access the categories by using the
     * Strings similiar to the ones from {@link Config#
     * getDepartmentName()} and {@link Config#getSectionName()}</p>
     * >.
75     *
     * @return All categorized groups.
     */

```

```

public Hashtable getStructuredOrganization() {
79     return structuredOrganization;
80 }

/**
 * Inserts the given group into the directory and gives you the
 * name of the next page. Will refetch the organization's
 * structure and refetch the given group.
 *
85 * @param awiGroup The group that should be inserted into the
 *     directory.
 * @return The name of the next page, see {@link Config#
 *     getModifyGroupPage()}.
 * @throws NamingException If something goes wrong while
 *     inserting.
 */
public String insertAwiGroupInDirectory(final AwiGroup awiGroup)
    throws NamingException {
90     this.insertIntoDirectory(awiGroup);
    this.setStructuredOrganization(model.getStructuredOrganization
        ());

    String groupRdn = awiGroup.relativeDistinguishedName();
    String groupCn = Utilities.convertDnToCn(groupRdn);
95     AwiGroup refetched = model.getAwiGroupByCn(groupCn);
    this.setOrgUnit(refetched);

    return Config.getGroupInfoPage();
}
100

/**
 * Returns whether the user is allowed to login or not.
 *
 * @return Returns true if the user is allowed, <
 *     code>false otherwise.
105 */
public boolean isAllowed() {
    return isAllowed;
}

110 /**
 * Returns whether a user, identified by a uid and his password,
 * is allowed to use this application.
 *
 * @param uid The uid as found in the Directory.
 * @param password The password of this user.
115 * @return true if the user is allowed, false
 *     if he is not allowed.
 */

```

```

public boolean isAuthenticationAllowed(String uid, String
    password) {
118     boolean isAllowed = model.isAuthenticationAllowed(uid,
        password);

120     if (isAllowed == true) {
        try {
            AwiPerson awiPerson = model.getAwiPersonByUid(uid);
            this.setUser(awiPerson);
        }
125     catch (NamingException exc) {
        log.warn("Wrong_uid_used:_" + uid);
        isAllowed = false;
        }
    }
130     this.setAllowed(isAllowed);
    return isAllowed;
}

/**
135  * Returns <code>true</code> if the actual user has superuser-
    rights (entitlement urn:awi:awi-bremerhaven.de:application:
    eGroup:admin:all is set. Returns <code>>false</code>
    otherwise.
    *
    * @return Returns <code>true</code> if the actual user has
        superuser-rights, <code>>false</code> otherwise.
    */
public boolean isSuperUser() {
140     boolean isSuperUser = false;

    NSArray entitlements = this.getUser().eduPersonEntitlement();
    if (entitlements.containsObject(Config.getSuperUserEntitlement
        ())) {
145         isSuperUser = true;
    }
    else {
        isSuperUser = false;
    }

150     return isSuperUser;
}

/**
    * Proceeds with the page that is able to modify the given group
    *
155  * @param awiGroup A modified AwiGroup.

```

```

    * @return The next page's name, see {@link Config#
      getModifyGroupPage()}.
158 */
159 public String modifyGroup(AwiGroup awiGroup) {
160     this.setOrgUnit(awiGroup);
    return Config.getModifyGroupPage();
}

/**
165 * Remove a number of EOEnterpriseObjects from the model.
    *
    * @param toBeRemoved An array of objects that will be removed.
    * @throws NamingException If something goes wrong during
      removal.
    */
170 public void removeEnterpriseObjects(final NSArray toBeRemoved)
    throws NamingException {
    for (int i = 0; i < toBeRemoved.count(); i++) {
        EOEnterpriseObject object = (EOEnterpriseObject) toBeRemoved
            .objectAtIndex(i);
        model.removeEnterpriseObject(object);
    }
175 }

/**
    * Gives you the name of a page that is able to show you a
      particular AwiGroup (its name, its head etc.).
    *
180 * @param orgUnit This AwiGroup will be stored in {@link #
      orgUnit} to be accessible by the next page.
    * @return The name of the GroupInfoPage.
    */
public String showGroupInformation(AwiGroup orgUnit) {
    this.setOrgUnit(orgUnit);
185
    return Config.getGroupInfoPage();
}

/**
190 * This method will lead you to the page that's designed to show
      you the members of the actual group ({@link #orgUnit}).
    *
    * @param awiGroup The Group of which the members should be
      shown.
    * @return The name of the next page, see {@link Config#
      getShowGroupMembersPage()}.
    */
195 public String showGroupMembers(AwiGroup awiGroup) {
    String nextPage = Config.getShowGroupMembersPage();

```

```

    this.setOrgUnit(awiGroup);
198
    return nextPage;
200 }

/**
 * Stores the modifications of the given group in the directory.
 *
205 * @param awiGroup The modified group.
 * @return The next page, {@link Config#getGroupInfoPage()}.
 * @throws NamingException If something goes wrong during LDAP-
 *         operation.
 */
public String storeAwiGroup(AwiGroup awiGroup) throws
    NamingException {
210     this.setOrgUnit(awiGroup);
    this.storeModifications();
    return Config.getGroupInfoPage();
}

215 /**
 * Calling {@link ModelAccessImpl#storeModifications()} to write
 * changes to the directory.
 *
 * @throws NamingException If something goes wrong on updating
 *         the directory.
 */
220 public void storeModifications() throws NamingException {
    model.storeModifications();
}

225 /**
 * This method invokes appropriate actions to subscribe a person
 * to an AwiGroup.
 *
 * @return The name of the next page, see {@link Config#
 *         getSubscribePage()}.
 */
230 public String subscribeToNewGroup(Object modifyEntity) {
    this.setLdapAttributeId(Config.getGroupMembersCriteria());
    this.modifyEntity = modifyEntity;
    return Config.getSubscribePage();
}

235 /**
 * Terminates this Session-instance. Closes the binding to the
 * model and
 * terminates.
 */

```

```

    public void terminate() {
240     this.logoutFromLDAP();
        super.terminate();
    }
}

```

Listing C.21: Session.java (Auszug)

## C.14. Utilities

```

1  /**
   * This class provides basic transformations and other useful
   * stuff to handle the EOModel.
   */
   public class Utilities {
5     /**
      * This gets an Object and tries to determine the internal type
      * of the contents. Valid types are <code>null</code> (empty
      * NSArray will be created), a single String (a NSArray
      * containing only this String will be created) or a NSArray if
      * the attribute is multi-valued (a NSArray with all the
      * attributes will be created).
      *
      * @param singleOrMultiValue The content of an attribute from
      * the Directory. Can be multi- or single-valued.
      * @return A NSArray containing the contents of the attribute.
      * Can containing only one element.
10     */
     public static NSArray createNSArray(Object singleOrMultiValue) {
        NSArray array;
        if (singleOrMultiValue == null) {
15         array = new NSArray();
        }
        else if (singleOrMultiValue instanceof String) {
            array = new NSArray(singleOrMultiValue);
        }
        else if (singleOrMultiValue instanceof NSArray) {
20         array = (NSArray) singleOrMultiValue;
        }
        else {
            throw new ClassCastException("Unable_to_cast_the_result_to_a
                _valid_class_(String_or_NSArray).");
        }
25     return array;
    }

    /**
     * This method takes an array of DNs, chops the Base-DN and
     * returns the RDNs. It should take care of spaces between the

```

```

    attributes , but it can't catch all circumstances. So keep
    your data as clean as you can.
30  *
    * @param dns An array containing the DNSs.
    * @return The same array , but without the Base-DNs.
    */
34 public static NSArray convertDnToRdn(NSArray dns) {
35     int size = dns.count();
    NSMutableArray rdns = new NSMutableArray(size);

    for (int i = 0; i < size; i++) {
        String dn = dns.objectAtIndex(i).toString();

40         String rdn = dn.replaceAll(",_*(\" + Config.
            getGroupsContainer() + "|" + Config.getPeopleContainer()
            + ") ,_*(\" + Config.getBaseDn() , "");

        rdns.addObject(rdn);
    }
45     return rdns;
}

/**
 * <p>Creates a single String from a model-attribute. Since we
 * don't know wether the returning object is null, a NSArray or
 * a String, but we'd like to have a single String-value we
 * need to parse that attribute and to force it to become a
 * String.</p>
50  *
 * <p>If the given Object is a String containing a minimum of 1
 * character we return this String. If it's equal to <code>null
 * </code> or empty we return <code>null</code> If it's an
 * array (NSArray or NSMutableArray) we return the first non-
 * empty String-value in that array. If it contains an non-
 * String-value(s) we throw an IllegalArgumentException.</p>
 *
 * @param arrayOrString A String or a NSArray or a NSMutablArray
 *
 * @return <code>null</code> if arrayOrString was not an
 * instance of String or NSArray (and NSMutableArray).<br/> <
 * code>null</code> if the String or array was empty or the
 * String-length was 0.<br/>
55  *     A String of the first non-null-Srting-value in an
 *     array.
 * @throws IllegalArgumentException If the given argument is not
 * an instance of NSArray or String. (Hint: this exception is
 * a RuntimeException, so you're responsible for passing the
 * right type of argument!)
 */

```

```

public static String createString(Object arrayOrString) throws
    IllegalArgumentException {
59     String result = null;
60
    /* the null-value */
    if (arrayOrString == null) {
        result = null;
    }
65     /* A String, null, length=0 or length > 0 */
    else if (arrayOrString instanceof String) {
        if (arrayOrString.toString().length() == 0) {
            result = null;
        }
70     else {
        result = arrayOrString.toString();
    }
    }
    /* An array */
75     else if (arrayOrString instanceof NSArray) {
        NSArray array = (NSArray) arrayOrString;

        /* An array with length > 0 */
        if (array.count() > 0) {
80             int size = array.count();

            for (int i = 0; i < size; i++) {
                Object content = array.objectAtIndex(i);

85                 /* A String at position i in that array */
                if ((content != null) && (content instanceof String)) {
                    String contentString = content.toString();

                    /* A "lengthy" String at position 0 */
90                     if (contentString.length() > 0) {
                        result = contentString;
                        break;
                    }
                }
95             }
        }
        /* Value has length = 0 or was null */
        else {
            result = null;
100        }
    }
    /* Anything else than String or NSArray */
    else {
        throw new IllegalArgumentException(
105            "This_method_is_only_valid_for_'NSArray'_or_'String'");
    }
}

```

```

    }
107     return result;
    }

110  /**
   * Converts a given array of DNs to CNs by calling {@link #
   *   convertDnToCn(String)}.
   *
   * @param dns An array of DNs.
   * @return The rest of the DNs, some CNs.
115  */
  public static NSArray convertDnToCn(NSArray dns) {
    int size = dns.count();
    NSMutableArray cns = new NSMutableArray(size);

120    for (int i = 0; i < size; i++) {
      String dn = dns.objectAtIndex(i).toString();
      log.debug("Convert_this_dn:_" + dn);

      String cn = Utilities.convertDnToCn(dn);
125      cns.addObject(cn);
    }
    return cns;
  }

130  /**
   * Converts a given DNs to a RDNs by cutting the basedn ({@link
   *   Config#getBaseDn()}).
   *
   * @param dn A DNs.
   * @return The rest of the DN, a CN.
135  */
  public static String convertDnToRdn(String dn) {
    String rdn = dn.replaceAll(",_?" + Config.getBaseDn(), "");

    return rdn;
140  }

  /**
   * Converts a Dn to a CN by chopping the BaseDN {@link #
   *   convertDnToRdn(String)} and the people-/groupscontainer ({
   *   @link Config#getPeopleContainer()}, {@link Config#
   *   getGroupsContainer()}).
   *
145  * @param dn A DN.
   * @return The resulting CN.
   */
  public static String convertDnToCn(String dn) {
    String rdn = convertDnToRdn(dn);

```

```

150
151     String cn = rdn.replaceAll(",_*(\" + Config.getGroupsContainer
        () + "|" + Config.getPeopleContainer() + "|" + Config.
        getGroupsContainer().toLowerCase() + "|" + Config.
        getPeopleContainer().toLowerCase() + "\")", "");
        cn = cn.replaceAll("(cn=|uid=)", "");

        return cn;
155     }

    /**
     * <p>Checks a String whether it contains a well-formed email-
        address.</p>
     *
     * <p><strong>Note:</strong> This method only provides a rude
        check. The localpart is only checked if it is present.</p>
     *
     * @param mailAddress The String that should be checked.
     * @return <code>>true</code> if <code>mailAddress</code> contains
        a valid email-address, <code>>false</code> otherwise.
     */
165     public static boolean correctMailAddress(String mailAddress) {

        if (mailAddress.matches("^.+?@[0-9a-z]([-.]?[0-9a-z])*\\.?[a-z
            ]{2,5}$")) {
            return true;
        }
170     else {
            return false;
        }

    }

175
    /**
     * <p>Creates an EOQualifier from iterating over the given array
        (containing DNs) by using the given attribute as filter-
        argument.</p>
     *
     * <p>Example: awi-2005,ou=Groups,dc=.... and awi-2005-05,ou=...
        with attribute=cn will result in ((cn='awi-2005') OR (cn='
        awi-2005-05')).</p>
180     *
     * <p>This method could be used to fetch some groups that are
        referenced in eduPersonOrgUnitDN by their DNs.</p>
     *
     * @param values An array of DNs.
     * @param attribute The attribute that should be used by the
        filter.
185     * @return An EOQualifier mathing the given arguments.

```

```

*/
187 public static EOQualifier createEOQualifierFromArray(NSArray
    values, String attribute) {
    NSArray cns = Utilities.convertDnToCn(values);

190    StringBuffer qualifierString = new StringBuffer();
    int size = cns.count();

    for (int i = 0; i < size; i++) {
        qualifierString.append(attribute);
195        qualifierString.append("=_%s");
        if (i < size - 1) {
            qualifierString.append("_OR_");
        }
    }

200    EOQualifier qualifier = EOQualifier.
        qualifierWithQualifierFormat(qualifierString.toString(),
            cns);

    return qualifier;
}

205 /**
 * Extracts DNs from an array of values as found in the
 * directory. Usually this array contains DNs and
 * eduPersonPrincipalNames. Uses {@link #correctMailAddress(
 * String)} to determine whether an entry is an eppn or not; if
 * not the entry will be returned.
 *
 * @param ldapValues An array of values from the Directory.
210 * @return An array of entries that are not eppns.
 */
public static NSArray getDnsFromLdapAttribute(NSArray ldapValues
    ) {
    NSMutableArray dns = new NSMutableArray();

215    for (int i = 0; i < ldapValues.count(); i++) {
        String entry = ldapValues.objectAtIndex(i).toString();
        if (Utilities.correctMailAddress(entry) == false) {
            dns.addObject(entry);
        }
220    }
    return dns;
}

/**
225 * Creates a DN from a given UID using {@link #createDnFromRdn(
 * String)} and {@link #createRdnFromUid(String)}.

```

```

227  *
    * @param uid The uid that should be converted.
    * @return A dn constructed from the given uid.
    */
230  public static String createDnFromUid(String uid) {
    return createDnFromRdn(createRdnFromUid(uid));
  }

  /**
235  * Creates a DN from a given dn using {@link #createDnFromRdn(
    String)} and {@link #createRdnFromCn(String)}.
    *
    * @param uid The uid that should be converted.
    * @return A dn constructed from the given uid.
    */
240  public static String createDnFromCn(String cn) {
    return createDnFromRdn(createRdnFromCn(cn));
  }

  /**
245  * Creates an rdn from the given uid by appending uid= and {
    @link Config#getPeopleContainer()},
    *
    * @param uid The uid that should be surrounded by uid= and the
    container.
    * @return A rdn constucted from the given uid.
    */
250  public static String createRdnFromUid(String uid) {
    StringBuffer rdn = new StringBuffer();
    rdn.append("uid=");
    rdn.append(uid);

255  return rdn.toString();
  }

  /**
    * Creates a dn from the given rdn by appending {@link Config#
    getBaseDn()}.
260  *
    * @param rdn The rdn that should be completed to a dn.
    * @return A dn constructed from the given rdn.
    */
    public static String createDnFromRdn(String rdn) {
265  if (rdn.startsWith("awiPending")) {
    StringBuilder dn = new StringBuilder();
    dn.append(rdn);
    dn.append(",");
    dn.append(Config.getPendingMembersContainer());
270  dn.append(",");
  }

```

```

    dn.append(Config.getBaseDn());
272     return dn.toString();
    }
    else if (rdn.startsWith("cn")) {
275     StringBuilder dn = new StringBuilder();
        dn.append(rdn);
        dn.append(",");
        dn.append(Config.getGroupsContainer());
        dn.append(",");
280     dn.append(Config.getBaseDn());
        return dn.toString();
    }
    else if (rdn.startsWith("uid")) {
285     StringBuilder dn = new StringBuilder();
        dn.append(rdn);
        dn.append(",");
        dn.append(Config.getPeopleContainer());
        dn.append(",");
        dn.append(Config.getBaseDn());
290     return dn.toString();
    }
    else {
        throw new IllegalArgumentException(
295         "Need a RDN starting with uid, cn or awiPending");
    }
}

/**
 * This creates an rdn from the given cn by surrounding it with
 * cn= and {@link Config#getGroupsContainer()}.
300 *
 * @param groupCn The cn of a group.
 * @return The rdn constructed from the given cn.
 */
public static String createRdnFromCn(String groupCn) {
305     StringBuffer rdn = new StringBuffer();
        rdn.append("cn=");
        rdn.append(groupCn);

        return rdn.toString();
310 }

/**
 * Takes a look at {@link de.awibremerhaven.egroup.Config#
 * getMapRFC2596()} to map language-dependant attributes (like
 * awiGroupNameEN) RFC2596-compliant attributes (awiGroupName;
 * lang-en).
 *

```

```

315  * @param localLdapAttributeId The name of an LDAP-attribute as
      * defined in {@link Config#createInternalGroupGUIFields()}
      * or {@link Config#createInternalGroupPage()}.
316  * @return The name of the attribute as used in the Directory.
      */
318  public static String mapRFC2596Key(String localLdapAttributeId)
      {
          Hashtable mapping = Config.getMapRFC2596();
320
          String mapped = (String) mapping.get(localLdapAttributeId);
          if (mapped != null) {
              return mapped;
          }
325  else {
              return localLdapAttributeId;
          }
      }

330  /**
      * Returns a timestamp with the pattern YYYYMMDDHHmmSSmsms (year
      * , month,
      * day, hour, minute, second, milliseconds).
      *
      * @return A timestamp.
335  */
  public static String getTimeStamp() {
      Calendar cal = Calendar.getInstance();
      int year = cal.get(Calendar.YEAR);
      int month = cal.get(Calendar.MONTH) + 1;
340  int day = cal.get(Calendar.DAY_OF_MONTH);
      int hour = cal.get(Calendar.HOUR_OF_DAY);
      int min = cal.get(Calendar.MINUTE);
      int sec = cal.get(Calendar.SECOND);
      int msec = cal.get(Calendar.MILLISECOND);

345
      DecimalFormat twoDigits = new DecimalFormat("00");
      String monthFormatted = twoDigits.format(month);
      String dayFormatted = twoDigits.format(day);
      String hourFormatted = twoDigits.format(hour);
350  String minFormatted = twoDigits.format(min);
      String secFormatted = twoDigits.format(sec);

      DecimalFormat fourDigits = new DecimalFormat("0000");
      String yearFormatted = fourDigits.format(year);
355  String msecFormatted = fourDigits.format(msec);

      StringBuilder timeStamp = new StringBuilder();
      timeStamp.append(yearFormatted);
      timeStamp.append(monthFormatted);

```

```
360     timeStamp.append(dayFormatted);
361     timeStamp.append(hourFormatted);
        timeStamp.append(minFormatted);
        timeStamp.append(secFormatted);
        timeStamp.append(msecFormatted);
365
        return timeStamp.toString();
    }
}
```

Listing C.22: Utilities.java (Auszug)



## D. Glossar

AAI	Authentisierungs- und Autorisierungsinfrastruktur. Verwaltung und Bereitstellung von Identifikationsdiensten und Berechtigungen.
API	Application Programming Interface. Schnittstelle zu Funktionen einer Software.
AWI	Stiftung Alfred-Wegener-Institut für Polar- und Meeresforschung, Bremerhaven
Base-DN	Definiert eine Suchbasis im Verzeichnis. Ausgehend von dieser wird der RDN verwendet.
CN	Common Name. Ist der 'übliche' Name eines Eintrages im Verzeichnisdienst, z.B. der Nachname einer Person oder der Kurzname einer Gruppe.
DANTE	Delivery of Advanced Network Technology to Europe. Betreibt europäische Forschungsnetzwerke [1].
DFN	Deutsches Forschungsnetz.
DFN-CERT	DFN Computer Emergency Response Team Services GmbH. Überwacht das DFN auf Sicherheitslücken und tauscht Informationen mit ähnlichen nationalen und internationalen Einrichtungen aus.
DFN-PCA	DFN Policy Certification Authority. Zentrale Zertifizierungsinstanz des Deutschen Forschungsnetzes.
DIT	Directory Information Tree. Datenstruktur, die von einem Verzeichnisdienst benutzt wird.
DN	Distinguished Name. Bezeichnet einen Eintrag im Verzeichnisdienst eindeutig.

e-Science	Bezeichnet allgemein die kollaborative wissenschaftliche Nutzung von verteilten Ressourcen über Institutionsgrenzen hinweg.
EO	Enterprise Objects. Framework im Lieferumfang von WebObjects. Bietet einen transparenten Datenzugriff auf eine Datenbasis.
EOModeler	WebObjects-Anwendung zur Erstellung des Datenmodells, den Beziehungen zwischen den Entitäten und der Modell-Klassen [12].
G-WiN	Gigabit-Wissenschaftsnetz des DFN [17].
GÉANT	Von 26 europäischen Netzwerken, der Europäischen Kommission und DANTE betriebenes Gigabit-Netzwerk [19].
GÉANT2	Siebente Generation des Europäischen Forschungsnetzwerkes [2].
Geschlossene Gruppe	Für diese Gruppen ist eine Genehmigung erforderlich, um Mitglied zu werden oder austreten zu können. Die Gruppenverwalter können Kollegen zwangsweise einschreiben.
Grid	Bezeichnung für verteilte Rechen- und Datenressourcen. Diese Rechnerverbünde werden vor allem für aufwendige Berechnungen genutzt.
IPv6	Internet Protocol Version 6. Nächste Version des Internet Protokoll in Schicht 3 des OSI/ISO-Referenzmodells (Netzwerkschicht) mit einem theoretischen Adressraum von $2^{128}$ IP-Adressen.
JNDI	Java Naming and Directory Interfac. Java Standard-API zum Zugriff auf Verzeichnisdienste.
LDAP	Lightwight Directory Access Protocol. Protokoll zur Abfrage von Verzeichnisdiensten, definiert in RFC 2251 [48].
LDAP-Klasse	Schema für einen Eintrag in einem Verzeichnisdienst. Eine Klasse besteht aus mehreren Attributen. Attribute können vererbt werden.
LDIF	LDAP Data Interchange Format, textuelles Format zum Austausch von Verzeichniseinträgen [18].
MARCOPOLI	Forschungsprogramm des AWI mit der GKSS-Forschungszentrum Geesthacht GmbH zu den Themen Meer, Küsten, Polargebiete und Infrastruktur.

NREN	National Research and Education Network. Nationales Forschungsnetzwerk, z.B. DFN G-WiN [17] oder SWITCHlambda [37].
NSArray	Apple-Implementierung ähnlich einem <code>Vector</code> , unveränderlich ( <code>immutable</code> ).
Offene Gruppe	Diese Gruppen darf jeder Mitarbeiter anlegen und sich selbst zu derartigen Gruppen hinzufügen und wieder entfernen. Der Gruppensprecher kann seine Kollegen zu seiner Gruppe einladen.
ORM	Objekt-relationales Mapping. Abbildung einer relationalen Datenstruktur auf Objekte, oft auf einfache JavaBeans.
Project Builder	WebObjects-Anwendung zur Verwaltung eines WO-Projekts.
QoS	Quality of Service. Unter diesem Begriff werden Netzdienste und Infrastrukturen zusammengefasst, die einem Dienstonutzer eine bestimmte Dienstqualität garantiert, z.B. eine definierte Bandbreite in einem Netzwerk.
RDN	Relative Distinguished Name. Bezeichnet einen Eintrag im Verzeichnisdienst ausgehend von einem Base-DN eindeutig.
SWITCH	Schweizerisches Forschungsnetzwerk.
TERENA	Trans-European Research and Education Networking Association. Interessenvertretung und Dachorganisation der europäischen NRENs.
Verzeichnisdienst	Datenbank, in welcher die Informationen in einer Baumstruktur vorliegen. Die Einträge sind durch einen DN eindeutig gekennzeichnet. Die möglichen Attribute sind in Schemata definiert und können mehrfach besetzt sein. Leseoperationen können sehr schnell bearbeitet werden und durch Indizierung der Attribute zusätzlich unterstützt werden. Der Dienst wird mit dem LDAP-Protokoll abgefragt.
VO	Virtuelle Organisation. Oft zeitlich beschränkter Zusammenschluss voneinander unabhängiger Organisationen oder Teile einer Organisation, z.B. zur Bearbeitung eines bestimmten Projektes.
WO	WebObjects. Suite von Frameworks und Anwendungen zur Erstellung von komponentenbasierten Anwendungen.

WOBuilder	WebObjects-Anwendung zur Erstellung von Komponenten für die Darstellungsschicht, Definition von Schnittstellen für diese Komponenten und deren HTML-Templates.
WOD	Diese Datei enthält das Mapping zwischen den <webobject>-Tags und Methoden in der Anwendung. Die Art und das Verhalten der eingebetteten Komponenten wird hier festgelegt.

# Abbildungsverzeichnis

2.1. Das Alfred-Wegener-Institut (Neubau) . . . . .	9
2.2. SunFire 6800 und Cray XD1 . . . . .	10
2.3. IBM Regatta P655 . . . . .	10
2.4. Dienstzugriff ohne AAI . . . . .	13
2.5. Dienstzugriff mit AAI . . . . .	14
3.1. Möglichkeiten der Mitgliedschaft . . . . .	35
3.2. Generelle Anwendungsfälle . . . . .	37
3.3. Anwendungsfall: Verwaltung der eigenen Mitgliedschaften . . . . .	38
3.4. Anwendungsfall: Verwaltung von Gruppeninformationen . . . . .	38
3.5. Frontend KollegInnen einladen . . . . .	39
3.6. Frontend Homepage . . . . .	40
3.7. Anwendungsfall: Mitgliederverwaltung . . . . .	40
3.8. Ablauf Einschreibung . . . . .	42
3.9. Beziehungen zwischen den Komponenten der Anwendung . . . . .	49
3.10. Sequenzdiagramm – Anzeige der Gruppenzugehörigkeit . . . . .	50
3.11. InvitePeoplePage im WOBuilder . . . . .	55
3.12. InvitePeoplePage im Frontend . . . . .	56
A.1. Organigramm AWI . . . . .	62
B.1. Organigramm MARCOPOLI . . . . .	63

# Listings

2.1. Beispiel für statische Gruppen . . . . .	18
2.2. Beispiel für dynamische Mailgruppen . . . . .	18
2.3. Beispiel für Forward Referencing . . . . .	19
2.4. Beispiel für <i>spatial groups</i> . . . . .	19
2.5. Beispiel für abstrakte Bezeichner . . . . .	20
C.1. Application.java (Auszug) . . . . .	64
C.2. Template.html (Auszug) . . . . .	65
C.3. ApplicationDisabled.java . . . . .	65
C.4. AwiLdapEntry.java . . . . .	66
C.5. AwiPersonDisplayNameComponen.html . . . . .	66
C.6. AwiPersonDisplayNameComponen.wod . . . . .	66
C.7. AwiPersonDisplayNameComponen.java (Auszug) . . . . .	66
C.8. Config.java (Auszug) . . . . .	68
C.9. DirectAction.java (Auszug) . . . . .	70
C.10. FunctionalAwiPersonsArrayComponent.api . . . . .	71
C.11. FunctionalAwiPersonsArrayComponent.java (Auszug) . . . . .	72
C.12. InvitePeoplePage.html . . . . .	73
C.13. InvitePeoplePage.wod (Auszug) . . . . .	75
C.14. InvitePeoplePage.java (Auszug) . . . . .	77
C.15. IsGroupOpenGroupCondComponent.html . . . . .	79
C.16. IsGroupOpenGroupCondComponent.wod . . . . .	79
C.17. IsGroupOpenGroupCondComponent.java (Auszug) . . . . .	80
C.18. JNDIAccessImpl.java (Auszug) . . . . .	81
C.19. ModelAccessImpl.java (Auszug) . . . . .	87
C.20. SerialNumberHelper.java (Auszug) . . . . .	92
C.21. Session.java (Auszug) . . . . .	93
C.22. Utilities.java (Auszug) . . . . .	100

## Literaturverzeichnis

- [1] DANTE – *Delivery of Advanced Network Technology to Europe*. <http://www.dante.net>
- [2] GÉANT2. [www.geant2.net](http://www.geant2.net)
- [3] *Hibernate*. <http://www.hibernate.org>
- [4] ICANN – *Internet Corporation for Assigned Names and Numbers*. <http://www.icann.net>
- [5] *Internet2 Middleware Initiative*. <http://middleware.internet2.edu/>
- [6] *Norddeutscher Verbund für Hoch- und Höchstleistungsrechnen (HLRN)*. <http://www.hlrn.de>
- [7] *Shibboleth Project*. <http://shibboleth.internet2.edu/>
- [8] *Spring Framework*. <http://www.springframework.org/>
- [9] *JUnit 3.8.1*. <http://www.junit.org/index.htm>. Version: August 2002
- [10] *Struktur der Stiftung Alfred-Wegener-Institut, Stand 4.1.2005*. Januar 2005
- [11] *Signaturgesetz*. BGBl I 2001, S. 876, Fassung vom 4.1.2005
- [12] Apple Computer Inc.: *Enterprise Objects (Manual)*. 2005. [http://developer.apple.com/documentation/WebObjects/Enterprise\\_Objects/EnterpriseObjects.pdf](http://developer.apple.com/documentation/WebObjects/Enterprise_Objects/EnterpriseObjects.pdf)
- [13] APPLE INC.: *WebObjects 5.2*. <http://www.apple.com/webobjects/>
- [14] BARTON, Tom: *Practices in Directory Groups / University of Memphis und Internet2 Middleware Architecture Committee for Education, Directory Working Group*. Version: Oktober 2002. <http://middleware.internet2.edu/dir/groups/internet2-mace-dir-groups-best-practices-200210.htm>. – Empfehlung. – Elektronische Ressource

- [15] BUNDESMINISTERIUM FÜR BILDUNG UND FORSCHUNG: *Bekanntmachung über die Förderung von Forschungsvorhaben auf dem Gebiet „e-Science und Grid-Middleware zur Unterstützung wissenschaftlichen Arbeitens“ im Rahmen der deutschen D-Grid-Initiative. Call 2004: „Community-Grids“ und „Grid-Middleware-Integrationsplattform“*. Bekanntmachung BMBF Referat 522. [http://iwrwww1.fzk.de/dgrid/foerderung/e-Science\\_Call.pdf](http://iwrwww1.fzk.de/dgrid/foerderung/e-Science_Call.pdf). Version: 12.08.2004
- [16] CUNNINGHAM & CUNNINGHAM, INC.: *Test Driven Development*. <http://c2.com/cgi/wiki?TestDrivenDevelopment>. Version: 20.04.2005
- [17] DEUTSCHES FORSCHUNGSNETZWERK DFN: *Gigabit-Wissenschaftsnetz (G-WiN)*. <http://www.dfn.de/content/gigabitwissenschafts/>
- [18] GOOD, G.: *The LDAP Data Interchange Format (LDIF)*. RFC 2849, Juni 2000
- [19] GÉANT: *Topology Map*. [http://www.geant.net/upload/pdf/Topology\\_Oct\\_2004.pdf](http://www.geant.net/upload/pdf/Topology_Oct_2004.pdf). Version: Oktober 2004
- [20] HEGERING, Heinz-Gerd ; HILLER, Wolfgang ; MASCHUW, Reinhard ; REINEFELD, Alexander ; RESCH, Michael: *D-Grid: Auf dem Weg zur e-Science in Deutschland*. [http://iwrwww1.fzk.de/dgrid/intern2/D-Grid\\_Strategie\\_17-12-03b.pdf](http://iwrwww1.fzk.de/dgrid/intern2/D-Grid_Strategie_17-12-03b.pdf). Version: 2003
- [21] HOWES, T. ; SMITH, M.: *The LDAP URL Format*. RFC 2255. <http://www.ietf.org/rfc/rfc2255.txt>. Version: Dezember 1997
- [22] HUNT, Andrew ; THOMAS, David: *Der Pragmatische Programmierer*. München : Carl Hanser Verlag, 2003. – ISBN 3-446-22309-6
- [23] HUNT, Andrew ; THOMAS, David: *The Pragmatic Starter Kit Series*. Bd. 2: *Pragmatic Unit Testing In Java with JUnit*. The Pragmatic Bookshelf, 2004. – ISBN 0-9745140-1-2
- [24] INTERNET2: *Abilene Network*. <http://abilene.internet2.edu>
- [25] INTERNET2 MIDDLEWARE ARCHITECTURE COMMITTEE FOR EDUCATION, DIRECTORY WORKING GROUP: *DRAFT Revision of eduPerson Specification*. Version: Dezember 2004. <http://www.nmi-edit.org/eduPerson/draft-internet2-mace-dir-eduperson-00.pdf>. – Spezifikation Draft Revision, MACE-DIR-Groups
- [26] INTERNET2 MIDDLEWARE ARCHITECTURE COMMITTEE FOR EDUCATION, DIRECTORY WORKING GROUP: *Grouper*. <http://middleware.internet2.edu/dir/groups/grouper/>. Version: April 2005. – MACE-DIR-Groups
- [27] MENDIS, Ravi: *WebObjects Developer's Guide*. Indianapolis : Sams Publishing, 2002. – ISBN 0-672-32326-5
- [28] OBJECTSTYLE: *Cayenne*. <http://www.objectstyle.org/cayenne/>

- [29] OBJECTSTYLE: *WOLips release 1.1.0.102 RC 2*. <http://www.objectstyle.org/woproject/index.html>. Version: Januar 2005
- [30] ORACLE CORPORATION: *TopLink*. <http://www.oracle.com/technology/products/ias/toplink/index.html>
- [31] SHIBBOLETH PROJECT: *A High-Level Technical Introduction to Shibboleth*. <http://shibboleth.internet2.edu/shib-tech-intro.html>. Version: 2005
- [32] SUN MICROSYSTEMS: *J2EE JavaServer Faces*. <http://java.sun.com/j2ee/javaserverfaces/>
- [33] SUN MICROSYSTEMS: *Code Conventions for Java Programming Language*. Version: 1999. <http://java.sun.com/docs/codeconv/html/CodeConvTOC.doc.html>. – Spezifikation
- [34] SWITCH: *Geschäftsbericht*. <http://www.switch.ch/de/about/SWITCH-GB2003.pdf>. Version: 2003
- [35] SWITCH NETSERVICES: *Authentication and Authorization Infrastructure (AAI)*. <http://www.switch.ch/aai/>
- [36] SWITCH NETSERVICES: *SWITCHmobile*. <http://www.switch.ch/mobile/>
- [37] SWITCH NETWORK: *SWITCHlambda*. <http://www.switch.ch/network/switchlambda/>
- [38] TERENA: *Mission and Objectives Statement*. <http://www.terena.nl/about/mission.html>. Version: 2002
- [39] TERENA: *Activity Plan 2005*. <http://www.terena.nl/about/actplan2005.pdf>. Version: 2004
- [40] TERENA SECRETARIAT: *TERENA creates new Middleware Task Force*. In: *News from TERENA* 156 (2004), September. <http://www.terena.nl/news/2004/newsflash156.pdf>
- [41] TERENA SECRETARIAT: *6DISS project will deliver the IPv6 message worldwide*. In: *News from TERENA* 170 (2005), April. <http://www.terena.nl/news/2005/newsflash170.pdf>
- [42] THE APACHE SOFTWARE FOUNDATION: *Apache Ant*. <http://ant.apache.org>
- [43] THE APACHE SOFTWARE FOUNDATION: *Apache Struts Web Application Framework*. <http://struts.apache.org/>
- [44] THE APACHE SOFTWARE FOUNDATION: *Jakarta Tapestry*. <http://jakarta.apache.org/tapestry/>

- [45] THE APACHE SOFTWARE FOUNDATION: *Jakarta Turbine Web Application Framework*. <http://jakarta.apache.org/turbine/>
- [46] THE APACHE SOFTWARE FOUNDATION: *Apache Logging Services – log4j 1.2.9*. <http://logging.apache.org/log4j/docs/>. Version: November 2004
- [47] WAHL, M. ; HOWES, T.: *Use of Language Codes in LDAP*. RFC 2596. <http://www.ietf.org/rfc/rfc2596.txt>. Version: Mai 1999
- [48] WAHL, M. ; HOWES, T. ; KILLE, S.: *Lightweight Directory Access Protocol (v3)*. RFC 2251. <http://www.ietf.org/rfc/rfc2251.txt>. Version: Dezember 1997
- [49] WIKIPEDIA: *Agile Softwareentwicklung*. [http://de.wikipedia.org/wiki/Agiler\\_Prozess](http://de.wikipedia.org/wiki/Agiler_Prozess). Version: 16.06.2005
- [50] WIKIPEDIA: *Polnische Notation*. [http://de.wikipedia.org/wiki/Polnische\\_Notation](http://de.wikipedia.org/wiki/Polnische_Notation). Version: 22.05.2005
- [51] WIKIPEDIA: *Model View Controller*. <http://de.wikipedia.org/wiki/MVC>. Version: 29.06.2005
- [52] WORLD WIDE WEB CONSORTIUM: *Cascading Style Sheets, level 2*. <http://www.w3.org/TR/REC-CSS2/>. Version: 1998
- [53] WORLD WIDE WEB CONSORTIUM: *HTML 4.01 Specification*. <http://www.w3.org/TR/html4/>. Version: 1999

## **Erklärung**

Die Diplomarbeit entstand in Zusammenarbeit mit einer Institution / Firma / Person außerhalb der Fachhochschule Oldenburg/Ostfriesland/Wilhelmshaven.

Soweit meine Rechte berührt sind, erkläre ich mich einverstanden, dass die Diplomarbeit Angehörigen der Fachhochschule Oldenburg/Ostfriesland/Wilhelmshaven für Studium/Lehre/Forschung uneingeschränkt zugänglich gemacht werden kann.

## **Eidesstattliche Versicherung**

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Diplomarbeit bis auf die offizielle Betreuung selbst und ohne fremde Hilfe angefertigt habe und die benutzten Quellen und Hilfsmittel vollständig angegeben sind.

Michael Holtermann  
Bremerhaven, den 11. Juli 2005