Management dynamischer Virtueller Organisationen in Grids

Dissertation

an der

Fakultät für Mathematik, Informatik und Statistik
der
Ludwig-Maximilians-Universität München

vorgelegt von

Michael Schiffers

Tag der Einreichung: 6. Juli 2007

Management dynamischer Virtueller Organisationen in Grids

Dissertation

an der

Fakultät für Mathematik, Informatik und Statistik
der
Ludwig-Maximilians-Universität München

vorgelegt von

Michael Schiffers

Tag der Einreichung: 6. Juli 2007
Tag des Rigorosums: 26. Juli 2007

1. Berichterstatter: **Prof. Dr. Heinz-Gerd Hegering**, Universität München
2. Berichterstatter: **Prof. Dr. Johann Schlichter**, Technische Universität München

Omnium enim rerum principia parva sunt.
(Aller Dinge Anfang ist klein.)
Cicero, De finibus bonorum et malorum (V, 58)

Danksagung

Diese Dissertation ist eigentlich die Geschichte einer mehrfachen (Wieder-) Entdeckung. Zum einen war es nach langen Jahren der Industrie die (Wieder-) Entdeckung der Wissenschaft, zum anderen war es die Erkenntnis, dass der „da draußen“ in der Industrie gewonnene Fundus an Erfahrungen und Erkenntnissen eben doch nicht nur auf schnelle Machbarkeit ausgerichtet ist und dass die Fragestellungen „hier drinnen“ in der Wissenschaft ungewöhnlich praxislastig sein können.

Natürlich sind solche (Wieder-) Entdeckungen stets Ergebnisse eines kollektiven Prozesses, an dem eine Vielzahl von Menschen Teil hatten. Und diesen Menschen möchte ich danken für ihre Beiträge zu dem, was hier nun in Form meiner Dissertation vorliegt. Mein ganz besonderer Dank gilt meinem Doktorvater, Prof. Dr. Heinz-Gerd Hegering, für die fordernde und fördernde Begleitung der Arbeit und den Freiraum, den er mir über die Jahre dafür ließ. Er wusste stets treffsicher, wann und wo er mir seine Kritik, aber auch seine Unterstützung, anzubieten hatte. Auch dem zweiten Berichterstatter, Prof. Dr. Johann Schlichter, gebührt für seine konstruktiven Anmerkungen ganz herzlicher Dank. Dann sind da alle Mitglieder des MNM-Teams, des Arbeitskreises *Grid* am Leibniz-Rechenzentrum und der VO-Management-Projekte der *D-Grid-Initiative* zu nennen, ohne deren Neugier, Kritik, Offenheit und Diskussionsbereitschaft viele Ideen keine Projektions- und Reibungsfläche gefunden hätten. Mein Dank geht deshalb an sie alle.

Dennoch wäre die Arbeit nicht möglich geworden ohne den Zuspruch meiner Familie, zu Beginn der Dissertation noch nicht ahnend, dass es zum Schluss zu einem familieninternen Promotions-Wettbewerb kommen würde. Ich danke deshalb meinem Vater und meinen Geschwistern, ganz besonders aber meiner Frau und meinen Kindern für die Zeit, die sie mir über Jahre für diese Arbeit eingeräumt haben.

München, im Juli 2007

Zusammenfassung

Seit Mitte der 1990er Jahre wird unter dem Grid-Problem allgemein das koordinierte Problemlösen und die gemeinschaftliche Nutzung von Ressourcen in dynamischen, multi-institutionellen, Virtuellen Organisationen verstanden. Das Konzept Virtueller Organisationen (VO) ist damit für Grids von zentraler Bedeutung. Intuitiv bestehen VOs aus Personen und/oder technischen Ressourcen autonomer realer Organisationen. Der für VOs typische Lebenszyklus impliziert zahlreiche, zum Teil neue, Anforderungen nicht nur an die Bereitstellung von Grid-Ressourcen, sondern insbesondere auch an das Management von VOs selbst. Fragen nach gezielter IT-Unterstützung in der Formation, dem Betrieb, und der Auflösung von VOs rücken in Grids immer mehr in den Vordergrund.

Trotz der drängenden Notwendigkeit eines auch gerade VOs als *managed objects* umfassenden, integrierten Grid-Management-Ansatzes, sind die Fragestellungen bezüglich der hierzu erforderlichen Architekturen, Plattformen und Betriebskonzepte noch weitgehend ungeklärt. Existierende Konzepte liegen bestenfalls für einzelne Teilaspekte vor (z.B. dem Mitgliedsmanagement). Bestätigt wird dies durch eine Analyse bestehender Architekturkonzepte, deren zum Teil erhebliche Defizite auf die aktuelle betriebliche Praxis im Grid-Management und den vereinfachend getroffenen Annahmen zu Lebensdauer, Gründungsprozess oder Kooperationsstruktur von VOs zurückzuführen sind.

Die Dissertation verfolgt das Ziel, eine VO-Managementarchitektur (VOMA), in der die Managementobjekte dynamische Virtuelle Organisationen sind, systematisch über eine umfangreiche Anforderungsanalyse und im Rahmen eines Model Driven Architecture (MDA)-Ansatzes zu entwickeln, um einerseits die Wiederverwendbarkeit von Modellen zu gewährleisten und andererseits flexibel hinsichtlich möglicher Einsatzszenarios zu bleiben. Im *Informationsmodell* der Architektur wird für alle am VO-Management beteiligten Rollen ein gemeinsames Verständnis über die auszutauschenden Managementinformationen festgelegt. Das *Organisationsmodell* identifiziert die am VO-Management beteiligten Rollen und ordnet ihnen entsprechende Handlungsdomänen zu. Im *Kommunikationsmodell* werden die spezifischen Anforderungen an die Kommunikationsmechanismen dieser Rollen spezifiziert. Im *Funktionsmodell* wird der Gesamtaufgabenkomplex des VO-Managements auf der Basis der anderen Teilmodelle in einzelne Funktionsbereiche gegliedert, die sich an VO-Lebenszyklen orientieren.

Während VOMA zunächst Plattform-unabhängig spezifiziert wird – und damit ein allgemeines Rahmenwerk liefert, muss die Architektur für einen realen Einsatz Plattform-spezifisch transformiert werden. Dies wird am Beispiel des Web Services Distributed Management (WSDM)-Rahmenwerkes gezeigt. Zudem wird geklärt, wie die Architektur in bestehende oder zukünftige Grid-Projekte integriert werden kann. Dazu wird VOMA um eine Infrastrukturkomponente (VOMA-I) erweitert, über die VOMA an Hand von Konfigurationsmustern in einem klassischen Manager/Agenten-Paradigma zum Einsatz gebracht werden kann. Die Tragfähigkeit des Konzeptes wird an Beispielen demonstriert.

Eine Zusammenfassung der erzielten Ergebnisse und ein Ausblick auf weiterführende Forschungsthemen runden die Arbeit schließlich ab.

Summary

Since the last decade the Grid problem has commonly been understood as coordinated problem solving and resource sharing in dynamic, multi-institutional virtual organizations. The concept of Virtual Organizations (VO) is thus central to Grids. Intuitively, VOs consist of individuals and/or resources „owned“ by autonomous real organizations. The lifecyle inherently associated with VOs poses several – mostly new – requirements regarding both the provisioning of Grid resources to the lifecycle phases and the management of VOs themselves. Additionally, current Grid scenarios demonstrate the emergence of requirements aiming at supporting the formation, the operation, and the termination of VOs in Grids.

Despite the obvious necessity of an integrated Grid management concept which looks at VOs as managed objects, adequate architectures, platforms, and operational concepts are still missing. Although existing concepts address partial aspects (e.g., membership management), the general questions, however, which mechanisms are needed for an efficient management of a VO lifecycle and how these can be embedded into a real organization's management solution remain unsolved. This is also reflected by an analysis of current practises in VO management and is mainly due to the underlying operational concepts and the tacit simplifying assumptions concerning the lifespan, the cooperation structure, and the formation processes.

This thesis closes this gap. It aims at developing a VO management architecture (VOMA) which looks at dynamic VOs as managed objects. The development follows a Model Driven Architecture (MDA) approach which is based on a comprehensive analysis of requirements systematically derived from real Grid scenarios. The MDA-approach has been selected in order to achieve both model reusability and operational flexibility.

The *information model* of the architecture determines the management information all roles participating in the management of VOs need to adhere to. The *organizational model* identifies the roles relevant to the VO management and their respective interactions. The *communication model* specifies the communication mechanisms between these roles and the *information model* the functional areas are determined along the VO lifecycle phases.

While VOMA is specified as a platform independent model – which thus provides a generic framework, the architecture must be transformed into a platform specific model. This is exemplified by mapping VOMA onto the Web Services Distributed Management (WSDM) framework. For a deployment into existing and future Grid projects VOMA will be extended by an infrastructure component (VOMA-I) which supports the roll-out into the classical manager/agents paradigm using configuration patterns. The VOMA concept is finally demonstrated by elaborated examples.

Both a summary of the achieved results and a collection of prospective further work finally complete the thesis.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Inhalt des Kapitels

Seit Mitte der 1990er Jahre wird unter dem Grid-Problem allgemein das „*koordinierte Problemlösen und die gemeinschaftliche Nutzung von Ressourcen in dynamischen, multi-institutionellen, virtuellen Organisationen*“ verstanden [Foster u. a., 2001]. Lag der Forschungsschwerpunkt anfänglich noch auf speziellen, auf konkrete Anwendungsfälle zugeschnittene Mechanismen zur Kopplung geographisch verteilter Supercomputer (Metacomputing), so hat sich der Fokus in den letzten Jahren auf die Sicherstellung von Interoperabilität, Integrierbarkeit und organisationsübergreifende Aggregation von Grid-Diensten (Service Grids [Krauter u. a., 2002]) sowie die Erfüllung komplexer Quality-of-Service (QoS)-Anforderungen, wie sie zur Lösung so genannter „Grand Challenges“ [Nelson, 2004]) erforderlich sind, verlagert [Foster u. Kesselman, 2004a]. Der aktuell festzustellende Boom von Grid-Projekten in Wissenschaft („e-Science“) [Berman u. a., 2003; Foster u. Kesselman, 2004b; Fox u. Walker, 2003; Gentzsch, 2005] und Industrie [Goyal, 2002; Joseph u. Fellenstein, 2004; Pietryka u. Srivastava, 2005; Sun Microsystems, 2003] deutet zudem darauf hin, dass – zumindest was den Teilbereich des *Resource Sharings* im oben genannten Grid-Problem angeht – auf eine mittlerweile akzeptabel stabile und robuste Grid-Middleware zurückgegriffen werden kann.

Gestützt werden diese Middleware-Ansätze durch eine Vielzahl international verabredeter Standards. So wird seit 1999 im Rahmen des Global Grid Forums (GGF)[1] [Global Grid

[1] GGF und die Enterprise Grid Alliance (EGA) haben sich Ende Juni 2006 zum Open Grid Forum (OGF) zusammengeschlossen.

Forum, 2005] die Entwicklung einer offenen Grid-Architektur vorangetrieben. Mit der Entscheidung des OGF zugunsten des Web Services Resource Frameworks (WSRF) [Czajkowski u. a., 2005] als Basis für zukünftige Standardisierungen wurden Grid-Dienste in das allgemeine Web Services-Framework[2] des World Wide Web Consortiums (W3C) eingebettet und in der ersten Version der Open Grid Services Architecture (OGSA)-Spezifikation [Foster u. a., 2006] dokumentiert. Im Gegensatz zum OGF fokussiert die internationale Globus Alliance [Globus Alliance, 2005] auf die Roadmap für das Globus Toolkit [Foster, 2005], das sich als de facto-Standard für OGSA- und WSRF-implementierende Grid-Middleware-Ansätze durchgesetzt hat.

Das Konzept der Virtuellen Organisation (VO) ist für Grids von zentraler Bedeutung und deshalb in der OGSA-Spezifikation an prominenter Stelle verankert. Intuitiv werden Virtuelle Organisationen aus Personen und/oder technischen Ressourcen autonomer realer Organisationen (*legal entities*) mit dem Ziel „rekrutiert", kooperativ und koordiniert zur Lösung eines (oder mehrerer) Probleme – dem eigentlichen Zweck der VO – beizutragen. Abbildung 1.1 zeigt dies exemplarisch in einem universitären Projektumfeld.

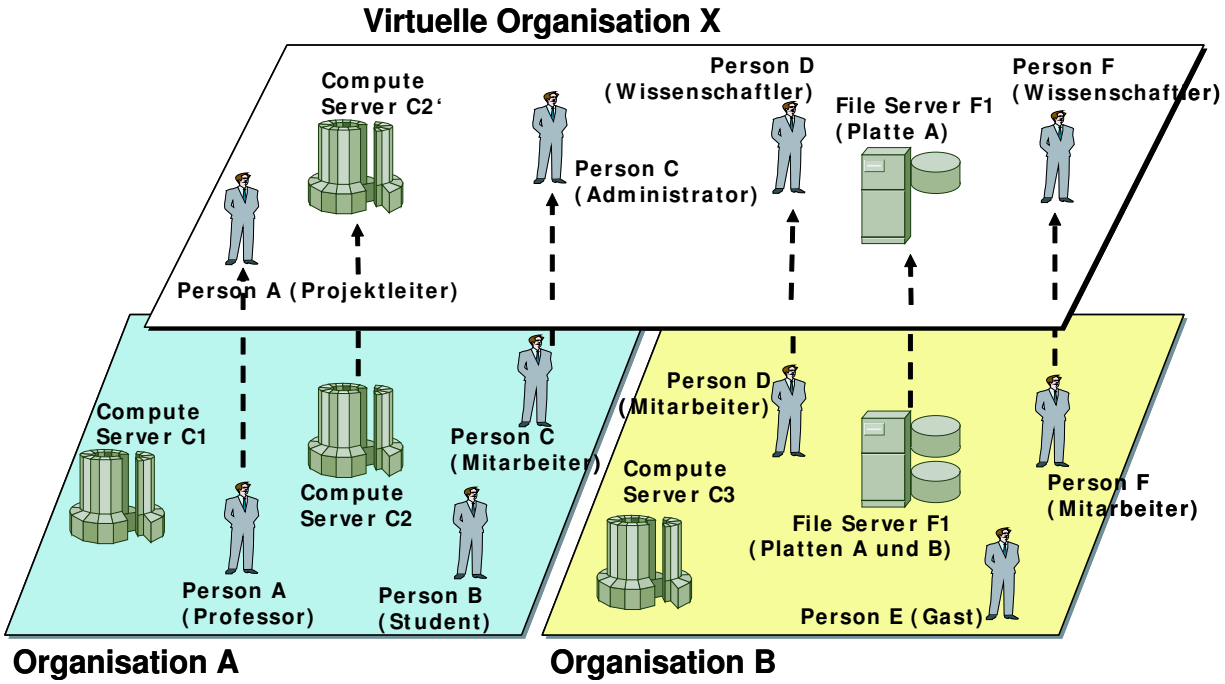


Abbildung 1.1: Typische Virtuelle Organisation nach [Foster u. Childers, 2005]

Virtuelle Organisationen sind daher zweckorientiert, einer gemeinsamen Interessenslage verpflichtet, die die „Geschäftsgrundlage" der VO bildet. Anders als reale Organisationen sind sie jedoch a priori zeitlich befristet angelegt, mit einer hohen Dynamik sowohl in ihrer

[2]Für einen ausgezeichneten Überblick über die hier relevanten Web Services-Standards per September 2005 siehe `http://www.innoq.com/soa/ws-standards/poster/Web_Services_Standards_09_2005.pdf`.

Zusammensetzung (Struktur) als auch in den internen und externen Interaktionsmustern (Prozesse).

Der für Virtuelle Organisationen typische Lebenszyklus impliziert zahlreiche – zum Teil neue – Anforderungen nicht nur an die Bereitstellung von Grid-Ressourcen und -Diensten, sondern insbesondere auch an das Management von Virtuellen Organisationen selbst [Hegering, 2005b]. Fragen nach einer IT-gestützten Formation, Evolution und Auflösung Virtueller Organisationen, nach adäquaten Policy-Mechanismen und deren Durchsetzbarkeit oder nach organisationsübergreifenden Workflow-Kompositionen und deren Einbettung in konfliktfreie Co-Management-Lösungen rücken mehr und mehr in den Vordergrund, nicht zuletzt wegen der auftretenden Dynamik [NGG2 Expert Group, 2004]. Für Ad-Hoc-Grids [Smith u. a., 2004], für Grids mit hohen nicht-funktionalen Anforderungen (Dependable Grids [Nguyen-Tuong u. a., 2004]) und für nachhaltige Grid-Infrastrukturen wie in der D-Grid-Initiative [Gentzsch, 2005] sind diese Fragen gar kritisch.

Trotz der drängenden Notwendigkeit eines auch gerade Virtuelle Organisationen als „*managed objects*“ umfassenden, integrierten Grid-Management-Ansatzes im Sinne [Hegering u. a., 1999], sind die dazu erforderlichen Architekturen, Plattformen, Betriebskonzepte und Maßnahmen noch weitgehend ungeklärt oder liegen bestenfalls konzeptionell vor, wie die Aktivitäten der Enterprise Grid Alliance (EGA) [Enterprise Grid Alliance, 2006][3] oder die ausgesprochen rudimentären OGSA-Ansätze [Foster u. a., 2006]) zeigen. Welche konkreten Mechanismen jedoch für das Management dynamischer Virtueller Organisationen notwendig sind und welche Routineaufgaben wie und unter welchen Randbedingungen automatisierbar sind, stellen nach wie vor offene Fragen dar. Diesem Themenkomplex widmet sich die vorliegende Arbeit.

1.1 Motivation und Zielsetzung

Grid-Technologien haben sich in den letzten Jahren von anfänglich proprietären Speziallösungen zu service-orientierten Grids auf der Basis offener Standards entwickelt [Foster u. Kesselman, 2004a]. Heute sind Grids prinzipiell dadurch gekennzeichnet, dass diverse, nicht notwendigerweise funktional disjunkte Computing-, Speicher- oder andere Problemlösungsdienste von rechtlich autonomen Dienstanbietern auf den ihnen jeweils „gehörenden“ Ressourcen-Pools angeboten werden. Für eine koordinierte Lösung komplexer Probleme können dann einige dieser Dienstanbieter *ausgewählte* Ressourcen und Dienste für eine *befristete* Periode an eine Virtuelle Organisation zur Nutzung durch fremde Dienstnehmer innerhalb eines vereinbarten Rahmens delegieren. Dieses Problem der bedarfsgerechten Gründung verlässlicher und skalierbarer Virtueller Organisationen in dynamischen, offenen, in der Regel häufig kompetitiven, Umgebungen ist Bestandteil einer um-

[3] Siehe auch Fußnote 1 auf Seite 1.

fassenderen Fragestellung, die in dieser Arbeit informell als das VO-Management-Problem in Grids bezeichnet wird, nämlich das

> „Management von Lebenszyklen Virtueller Organisationen über einer Service-orientierten Grid-Infrastruktur“.

VO-Management adressiert damit nicht nur die *Formation* Virtueller Organisationen, sondern auch deren operativen *Betrieb* und deren *Auflösung*, dies alles unter Berücksichtigung von Zielkonformität trotz mangelnder Durchsetzbarkeit in der Bereitstellung „realer“ Ressourcen und Dienste.

VO-Management unterscheidet sich vom allgemeinen Grid-Management durch seinen Fokus auf Virtuelle Organisationen statt auf Grid-Infrastrukturen. Obwohl dennoch das VO-Management-Problem von fundamentaler Bedeutung für ein effektives und effizientes Grid-Management ist, ist es bis heute nur marginal behandelt worden. Erst in jüngster Zeit (siehe Kapitel 4) wird vereinzelt versucht, das VO-Management-Problem einer ganzheitlichen Betrachtung zuzuführen. Folgerichtig zeigen die heutigen Ansätze eine Reihe erheblicher **Defizite**, die im wesentlichen aus der aktuellen betrieblichen Grid-Praxis und den ihr zugrundeliegenden impliziten oder expliziten Annahmen resultieren.

Annahme langlebiger und geschlossener VOs. Bisherige Grid-Projekte beruhen auf dem Modell langlebiger, geschlossener Pools von Ressourcen und Nutzergruppen. So sind beispielsweise das LHC-Grid [LCG, 2005], das DEISA-Supercomputing-Grid [DEISA, 2006], das NASA Information Power Grid [IPG, 2005] oder das GriPhyN-Grid [GriPhyN, 2005] auf mehrere Jahre angelegt und nur „der Community bekannten“ Wissenschaftlern und Ingenieuren zugänglich. Solche Einschränkungen sind allerdings für dynamische Grids wie dem FireGrid [Berry u. a., 2005] oder dem MEDIGrid [Eftichidis, 2005] unzulässig, da dadurch nicht nur der Zugang zu Grid-Technologien für andere Benutzergruppen künstlich beschränkt wird, sie favorisieren auch ein (quasi-) statisches VO-Modell, das transiente, kurzlebige Kollaborationen nicht zulässt. VOs werden in diesen Umgebungen in der Regel zudem *manuell* gebildet und betrieben, eine Tool-Unterstützung besteht – wenn überhaupt – nur für Einzelaspekte (zum Beispiel der VO Membership Service (VOMS) [Alfieri u. a., 2003] zur Regelung von Zugriffsrechten oder Workflow-Management-Systeme [Yu u. Buyya, 2005] zur Kopplung von Grid-Diensten). Eine *dynamische* Formation Virtueller Organisationen wird jedoch nicht unterstützt, sie wird – im Gegenteil – in der Regel als vollzogen postuliert.

Annahme manueller Administration. Der administrative Aufwand für die Formation einer VO ist nicht-trivial und für ein Individuum oder eine Organisation ohne entsprechend ausgebildete Spezialisten und adäquate Tool-Unterstützung kaum zu bewältigen, wie das folgende Beispiel zeigt.

Beispiel: Möchte ein Wissenschaftlerteam spontan für einen kurzen Zeitraum

eine Reihe von Simulationsexperimenten durchführen, so ist dies mit den heutigen Grid-Konzepten zwar durchführbar, der dazu notwendige administrative Aufwand reduziert den Nutzen jedoch nicht unerheblich. Formal muss die Gruppe nämlich zunächst eine VO etablieren, dann die VO-spezifischen Policies definieren, die individuellen Rechte und Verantwortlichkeiten beschreiben und die erforderlichen administrativen Rechte zuordnen. Schließlich muss der Administrator innerhalb der VO die entsprechenden *Grid-Credentials* [Grimm u. Pattloch, 2006] kreieren. Zusätzlich muss jede an der VO partizipierende Organisation die geeignete Grid-Middleware unterstützen und die von ihr angebotenen Dienste als Teil dieser Middleware veröffentlichen. Erst danach sind die Nutzer in der Lage, innerhalb des Teams im Kontext eigener Rechte zu interagieren.

Nur mit einer weitgehenden Automatisierung des Formationsprozesses dynamischer Virtueller Organisationen durch organisations*übergreifende* Workflows bleibt der Aufwand beherrschbar. Eine generisch ausgerichtete Menge von Diensten zur Unterstützung des Administrationsaspektes des VO-Management-Problems (*capabilities* in [Foster u. a., 2006]) ist allerdings bisher nicht betrachtet worden.

Annahme geringer Dynamik. Die weitaus meisten heutigen Grid-Projekte gehen von der Annahme stabiler, persistenter und langfristig angelegter VOs mit selten wechselnden Nutzern und Ressourcen aus. Dies führt in der Regel zu unzulässigen Vereinfachungen und proprietären Speziallösungen. Beispielsweise stellen VOs prinzipiell kollaborative Gruppen dar, in denen sich die einzelnen Mitglieder gegenseitig vertrauen. Ein Konzept von Vertrauen und Reputation wird für geschlossene VOs jedoch stets implizit wegen des unterlegten Formationsprozesses angenommen. Konsequenterweise können Grid-Ressourcen und -Nutzer in solchen VOs nicht anonym sein, sie gehorchen vielmehr a priori festgelegten, wohl-etablierten Autorisierungs- und Authentifizierungs-Mechanismen, die keine spontanen Adaptionen zulassen. Obwohl in der letzten Zeit diverse Sicherheitsarchitekturen für Grids diskutiert wurden (siehe etwa [Alfieri u. a., 2003]), basieren diese nach wie vor sämtlich auf der oben dargestellten Annahme geringer Dynamik. Ein adaptives Sicherheitsmodell, das eine zur Laufzeit inkrementelle VO-Evolution unterstützen würde, fehlt.

Annahme zentraler Kontrolle. Grid Computing ist generell durch das Fehlen einer zentralen administrativen Kontrollinstanz gekennzeichnet. Dies erschwert das Management im Vergleich zu klassischen, zentral verwalteten, Systemen nicht unerheblich. So werden beispielsweise die Durchsetzbarkeit von Policies, die dynamische Verhandlung von Service Level Agreements (SLA) oder die Überwachung globaler Ressourcen-Zustände wegen der sich häufig überlappenden (und verändernden) Management-Verantwortlichkeiten (*Co-Management*) problematisch. Gerade wegen dieser Schwierigkeiten werden in vielen aktuellen Grid-Projekten immer noch zentrale (obwohl

durchaus verteilte) Managementkonzepte etabliert, die auf „forcierten Homogenisierungen“ verschiedener Ebenen [Garschhammer u. Schiffers, 2005] aufbauen.

Annahme der VO-Existenz. Als zeitlich befristete Entitäten unterliegen VOs per se Lebenszyklen, die zur Zeit in keinem Grid-Projekt komplett unterstützt werden. Selbst eine qualitative Analyse bestehender *Grid-fremder* VO-Projekte [Camarinha-Matos, 2005] zeigt, dass auch in diesem Kontext die wenigsten Projekte einen vollständigen Lebenszyklus-Support leisten. Insbesondere die Formations- und Auflösungsphasen werden nur unzureichend abgedeckt. Stattdessen werden VOs stets als existent postuliert. Eine ganzheitliche Sicht auf den kompletten Lebenszyklus einer VO ist allerdings im Kontext dynamischer VOs von erheblicher Bedeutung.

Annahme fester Kooperationsstrukturen. Virtuelle Organisationen lassen sich in ihrer internen Struktur nach der Kooperationsstruktur der beteiligten realen Organisationen kategorisieren, die neben den erforderlichen Informationsflüssen auch die wichtigen Entscheidungsrelationen (*governance*) definiert [Katzy u. Löh, 2003]. In Lieferketten (*supply chains*) interagieren die VO-Mitglieder entlang einer Prozesskette, in Sterntopologien fungiert typischerweise ein Mitglied als zentraler Ansprechpartner (*hub-and-spoke*), in projekt-orientierten Konstellationen sind hingegen typische Peer-to-Peer-Interaktionen zu finden. Obwohl Beispiele dieser Kooperationsmuster auch in Grids zu finden sind, manchmal sogar in hybrider Form, sind diese Muster doch immer stabil: VOs sind normalerweise strukturell nicht rekonfigurierbar. Rekonfigurierbarkeit ist bei VOs jedoch immer dann von Bedeutung, wenn sich die „VO-Mission“ ändert oder im Fehlerfall.

Annahme intra-organisatorischer Managementaspekte. Aktuelle Grid-Projekte basieren sämtlich auf der Annahme bereits existierender VOs (siehe oben). Konsequenterweise beschränkt sich die VO-Managementfunktionalität auf intra-organisationale Abläufe. VO-Managementdienste adressieren deshalb vornehmlich isolierte Fragestellungen wie das Management der Mitgliedschaft zu Virtuellen Organisationen oder die Delegation von Ressourcen zu Virtuellen Organisationen. Ein notwendiger umfassenderer Managementansatz, der einerseits VOs selbst als „*managed objects*“ betrachtet und andererseits das Management von Föderationen dynamischer Virtueller Organisationen unterstützt (Inter-Grid), ist jedoch nicht in Sicht. Die Gründe dafür sind vor allem einer fehlenden OGSA-konformen Gesamtkonzeption im VO-Management zuzuschreiben. Aktuelle „OGSA-Capabilities“ gemäß [Foster u. a., 2006] füllen diese Lücke auch nicht.

Zusammenfassend kann festgehalten werden, dass sich aktuelle VO-Managementansätze zur Zeit überwiegend in dem in Abbildung 1.2 markierten Bereich bewegen, wobei die Graustufen einen Hinweis auf die Ausprägung der Dimensionen geben. Wesentliche Eigenschaften dynamischer Virtueller Organisationen werden entweder gar nicht oder nur unzureichend berücksichtigt (siehe auch Kapitel 4), eine Integration in existierende Grid-Standards ist zudem komplett offen. Als primäre Defizite ergeben sich daraus:

(i) VOs wurden bisher nicht als „*managed objects*“ behandelt. Eine Abdeckung des gesamten Lebenszyklus einer VO, die Unterstützung transienter, kurzlebiger VOs und die Unterstützung sich überlappender VOs fehlen damit komplett.

(ii) Eine adäquate VO-Managementarchitektur und deren Teilmodelle nach [Hegering u. a., 1999] wurde bisher nicht konzipiert.

(iii) VO-Managementdienste wurden im Rahmen OGSA-konformer Werkzeuge bisher nicht realisiert.

(iv) Eine saubere Methodik zur Bereitstellung einer VO-Managementarchitektur fehlt.

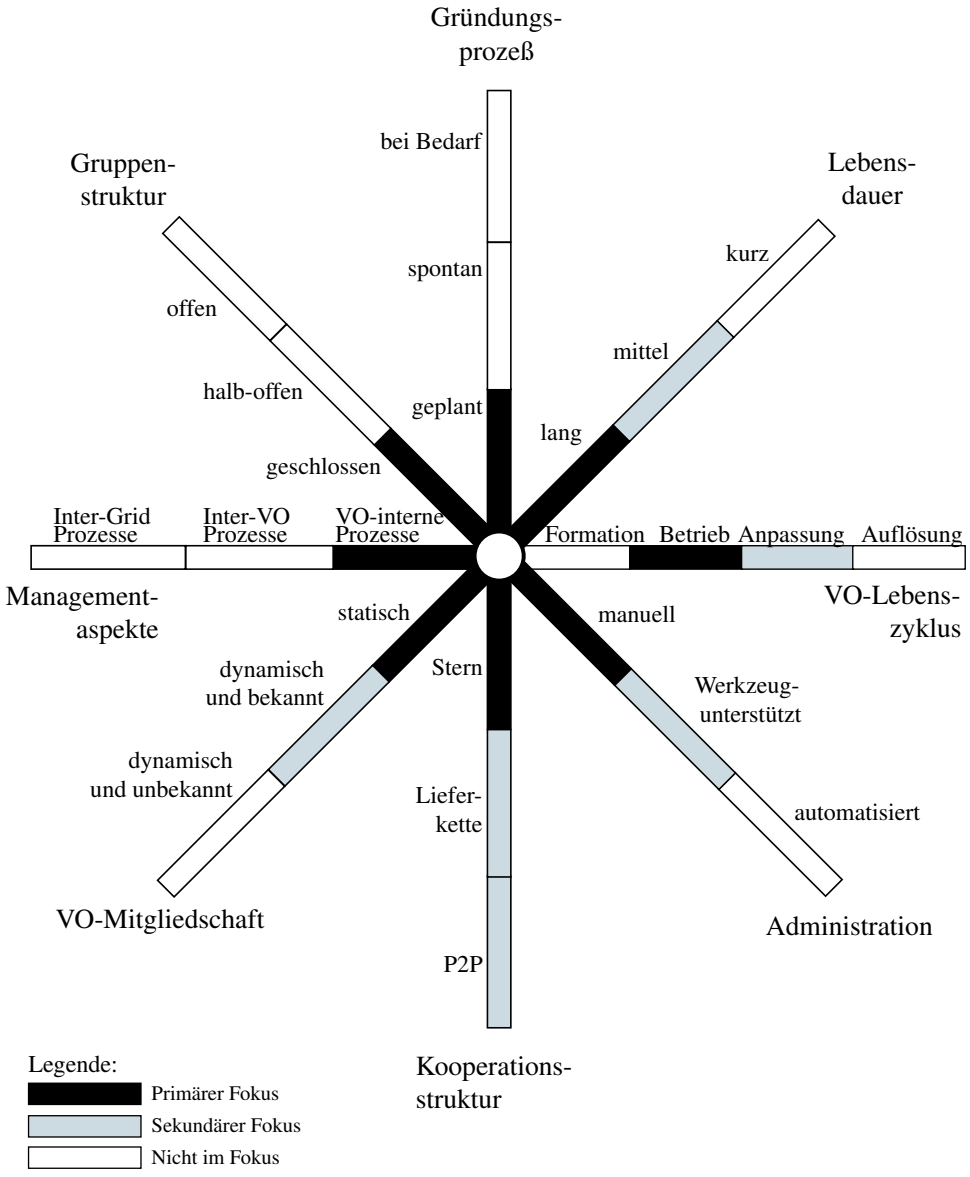


Abbildung 1.2: Fokus aktueller VO-Management-Ansätze

Vor diesem Hintergrund ist es das **Ziel dieser Arbeit**, eine Architektur zur Beschreibung der grundlegenden Konzepte und Festlegungen, wie sie für das Management Virtueller Organisationen in Grids notwendig sind, zu spezifizieren. Die Architektur wird einerseits dazu dienen, die Beschreibung Virtueller Organisationen als Managementobjekte zu ermöglichen, die involvierten Rollen und deren Interaktionsformen zu definieren, die notwendigen Kommunikationsvorgänge zu beschreiben und die erforderlichen Managementfunktionalitäten Plattform-unabhängig bereitzustellen. Anderseits wird die Architektur aber auch eine Methodik induzieren, die ein ganzheitliches Management Virtueller Organisationen erst ermöglicht. Damit werden nicht nur die angesprochenen Defizite überwunden, es wird auch die für Grids notwendige Flexibilität erreicht. Abbildung 1.3 zeigt den angestrebten Sollzustand.

Bevor die konkreten Fragestellungen dieser Arbeit diskutiert werden, sind einige Bemerkungen angebracht:

Bemerkung 1: Modellierungsproblematik. Der Entwurf von Managementanwendungen für dynamische Virtuelle Organisationen aus Basisdiensten ist u.a. deshalb schwierig, weil er sich an *lokalen*, nicht notwendigerweise kooperativen, Interessen und Anforderungen autonomer Ressourcen/Service-Provider orientieren muss („*locality over globality*“ [Kurowski u. a., 2004]), gleichzeitig aber mit dem Fokus auf den VO-Lebenszyklus ein *globales* Ziel verfolgt. Hier treten Interaktionsphänomene auf, die mit bisherigen Modellierungstechniken nicht oder nur sehr umständlich zu bewältigen sind und daher insgesamt wenig verstanden sind. Beispielsweise versagen traditionelle Modellierungsansätze in der Regel, sobald die Fortpflanzung von Wirkungen lokaler Management-Interventionen über mehrere Organisationen hinweg, ob gewollt oder ungewollt, studiert werden soll.

Beispiel: Ein Konsortium von Klimaforschern verwendet Grid-Technologien für die exakte Vorhersage von Tropenstürmen auf der Basis großflächig verteilter Wetterstationen und hochentwickelter Klimamodelle. Die Zielsetzung dieser VO geht einher mit starken Fehlertoleranz-Anforderungen (die Grid-Anwendung darf nicht terminieren während der Verfolgung eines schweren Sturmes) und einer hohen Dynamik in der Orchestrierung von Diensten und der Bereitstellung von Ressourcen. Diese werden typischerweise über Priorisierungsregelungen im Rahmen lokaler und globaler Policies erfüllt. Zur Durchsetzung solcher Policies können laufende Grid-Jobs für die Freigabe reservierter oder belegter Ressourcen unterbrochen werden, was wiederum Prozesse in realen Organisationen oder innerhalb anderer VOs beeinflussen kann (nach [Foster u. a., 2004]).

Die Unüberschaubarkeit solcher in der Regel nicht explizit dargestellter Einflüsse reduziert die Praxistauglichkeit traditioneller Modellierungsmethoden gerade in groß-

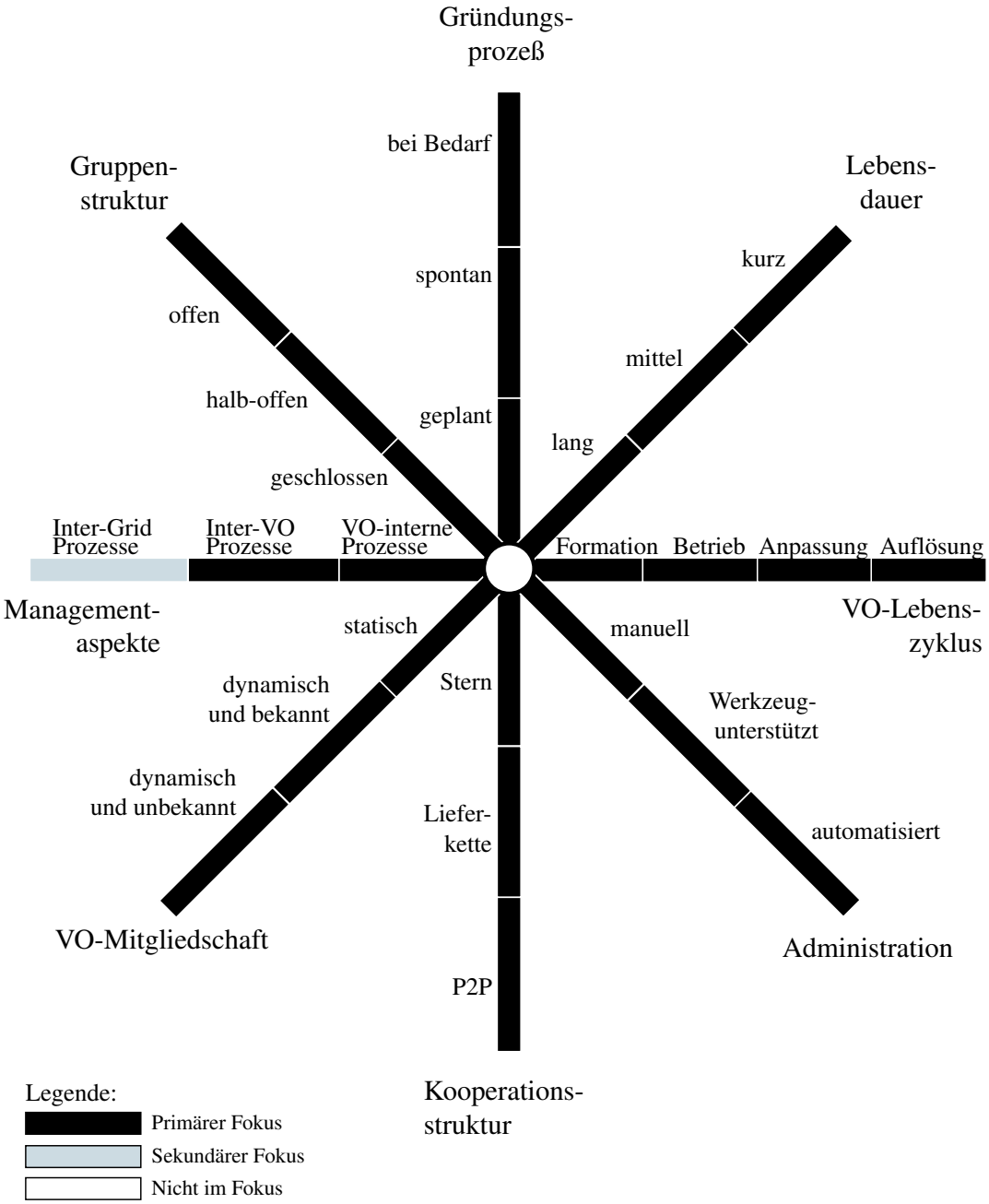


Abbildung 1.3: Fokus dieser Arbeit

skalierten Umgebungen wie Grids. Eine *inkrementelle restriktive Entwurfsstrategie*, der das Paradigma zugrundeliegt, alles zu erlauben und nur die Situationen auszuschließen, die z.B. die Systemsicherheit gefährden oder der Erfüllung der „VO-Mission“ entgegenstehen, ist hier möglicherweise vielversprechender als der traditionelle Ansatz. Eine Lösung wird in [Wedig, 2004] vorgestellt, wird hier aber nicht weiter vertieft, da die Bewertung von Modellierungsstrategien an sich hier nicht im Vordergrund stehen.

Bemerkung 2: Definitionsproblematik. VOs werden zur Zeit unter verschiedenen Blickwinkeln diskutiert. Häufig werden VOs als Mengen von (Grid-) Ressourcen (siehe etwa [Treadwell, 2006]) oder als einfache Liste von Mitgliedern (z. B. [Norman, 2006]) betrachtet. Die in diesem Zusammenhang naheliegende Anwendbarkeit von Verfahren des Grid Resource Managements [Nabrzyski u. a., 2004] für ein konzertiertes Management von Ressourcen ist allerdings noch nicht nachgewiesen worden. Werden VOs hingegen als reiner Dienstanbieter begriffen (zum Beispiel [Kürümlüoglu u. a., 2005]), sind entsprechende Managementkonzepte aus der Organisationstheorie [Camarinha-Matos u. a., 2005] und dem IT-Service-Management (siehe etwa [Köhler, 2004; Schmidt, 2001]) möglicherweise hilfreich. Eine wieder andere Sicht adressiert VOs als Dienst an sich [Dreo Rodošek u. a., 2005]. Unter diesem Blickwinkel werden für das Management dynamischer VOs Methoden des IT-Service-Managements [Dreo Rodošek u. Hegering, 2004; Dreo Rodošek, 2002] oder des Web Services Managements [OASIS, 2006g] verwendet. Obwohl die Adäquatheit jeder dieser Betrachtungsebenen für einzelne Szenarios durchaus bestätigt wurde oder Gegenstand aktueller Forschungsprojekte ist, ist der allgemeine Nachweis ihrer Tragfähigkeit für organisationsübergreifende, groß-skalierte und dynamische Grid-Umgebungen, wie sie in dieser Arbeit betrachtet werden, noch nicht erbracht. Eine Konsolidierung der verschiedenen Ansätze erscheint jedoch notwendig. Für eine weitergehende Diskussion der Definitionsproblematik wird auf Kapitel 2.1.3 verwiesen.

Nach diesen Vorbemerkungen können nun im nachfolgenden Abschnitt die in dieser Arbeit untersuchten Fragestellungen vorgestellt werden. Die Vorgehensweise zur Lösung der Fragestellungen wird in Abschnitt 1.3 erläutert. Eine Abgrenzung von verwandten Forschungsarbeiten folgt im Abschnitt 1.4. Die wesentlichen Ergebnisse der Arbeit werden schließlich im Abschnitt 1.5 zusammengestellt.

1.2 Fragestellungen

Zur genaueren Eingrenzung des Forschungsschwerpunktes dieser Arbeit ist es zunächst erforderlich, die Vielschichtigkeit des VO-Managementproblems (siehe Seite 4) zu beleuchten. Dazu dienen in erster Näherung die Achsen in Abbildung 1.3, weitere Gesichtspunkte werden im Kapitel 3 adressiert. Die folgenden Darstellungen dienen zwar der Eingrenzung des Themenkreises, es soll allerdings nicht daraus abgeleitet werden, dass die diskutierten Management*anwendungen* der Gegenstand dieser Arbeit sind. Vielmehr werden sie späteren Kapiteln als Anwendungsfälle dienen.

VO-Lebenszyklus. Virtuelle Organisationen werden stets für eine begrenzte Lebensdauer gegründet und unterliegen einem Lebenszyklus, der mit der Identifizierung der „richtigen“ Mitglieder beginnt und über die eigentliche Formation, den Betrieb, eine Anpassungsphase schließlich zur Terminierung und Auflösung führt. Das Management

dynamischer VOs, wie es in dieser Arbeit untersucht wird, muss sich also der Herausforderung stellen, wie die jeweiligen Phasenübergänge zu bewältigen sind. Dazu sind nicht nur Mechanismen zur Ableitung des aktuellen Zustands einer VO (als *managed object*) und zur Identifizierung möglicher Phasentransitionen notwendig, sondern es ist auch darzustellen, welche Prozesse in den einzelnen Phasen wann anzustoßen sind. So muss bei der Terminierung und Auflösung Virtueller Organisationen hinterfragt werden, wie und unter welchen Umständen dies geschehen kann, wem warum welche Ergebnisse wann gehören und wie die im Laufes des Lebenszyklus gewonnenen Daten verwertet werden (zum Beispiel zum Aufbau von Vertrauensbeziehungen oder für Audits). Die Ausweitung der Betrachtung auf den kompletten Lebenszyklus führt damit zur Notwendigkeit, den Status einer VO überwachen und darstellen zu können sowie die für den Fortschritt des VO-Lebenszyklus notwendigen Phasentransitionen orchestrieren bzw. choreographieren zu können[4]

Lebensdauer. Obwohl VOs heute typischerweise für mehrere Jahre etabliert werden, sind kurzlebig ausgelegte VOs (für die Dauer von Tagen oder kürzer) nicht nur denkbar, im NextGrid-Projekt [NGG2 Expert Group, 2004] ist es gar ein Standardszenario. Bei kürzer werdenden Lebensdauern werden in der Regel standardisierte Managementmuster verstärkt in den Vordergrund rücken, da diese den Automatisierungsanforderungen am ehesten genügen [Vogel, 2004]. Zu klären ist allerdings, welche Managementfunktionalitäten damit abgedeckt werden können, wie diese Muster klassifiziert werden können, wie sie identifiziert werden können und welche Aspekte im Lebenszyklus einer VO damit bis zu welchem Grad gemanagt[5] werden können.

Gründungsprozess. VOs werden nicht notwendigerweise von langer Hand geplant obwohl dies heute der Regelfall ist. Sie werden in Zukunft dynamisch gegründet, bei Bedarf oder als Reaktion auf Schlüsselereignisse asynchroner und synchroner Art [Dreo Rodošek u. a., 2005]. Werden VOs jedoch kurzfristiger gegründet, stellt sich ganz generell die Frage, wie die für das Ziel der VO „richtigen“ Ressourcen, Dienste und Mitglieder gefunden werden können, eventuell sogar verpflichtet werden können? Kann man ihnen trauen? Wie und wo werden umgekehrt von den möglichen Kandidaten deren „Fähigkeiten“ publiziert und bewertet? Ebenso ist zu klären, ob und wie Formationsereignisse klassifiziert werden können und welche charakteristischen Eigenschaften die einzelnen Klassen insbesondere bezogen auf ihren Managementauftrag besitzen. Welche Workflows müssen initiiert werden? Lassen sich existierende Konzepte wie das aus dem Grid-Resource Management bekannte Service Negotiation and Acquisition Protocol (SNAP) [Czajkowski u. a., 2004] adaptieren?

[4]Während die *Orchestrierung* von Diensten eine zentrale Kontrolle zur Überwachung des Prozessablaufes vorsieht, bezieht sich die *Choreographie* von Diensten auf eine verteilte Art der Aufgabenbearbeitung, in der es mehrere kontrollierende Instanzen geben kann bzw. der Kontrollfluss von jedem ausführenden Dienst automatisch an den nächsten Dienst weitergegeben wird [Janssen, 2003]. Als Oberbegriff wird in dieser Arbeit der Terminus „Konzertierung“ verwendet.

[5]Trotz einiger sprachlicher Bedenken verwenden wir hier die germanisierte Form *managen* des englischen Begriffes *to manage* weil er sich weitläufig so durchgesetzt hat.

Gruppenstruktur. Geschlossene Benutzergruppen können ihre Koordinationsmechanismen und die Verteilung von Rechten und Verantwortlichkeiten innerhalb der Gruppe wegen des *per constructionem* angenommenen gegenseitigen Vertrauens a priori festlegen und durchsetzen. Von daher tendieren diese Gruppen zu weniger formalen Gründungs- und Betriebsmaßnahmen als offene Benutzergruppen, für die schon allein ein adäquates Trustmanagement nicht trivial ist [Hommel u. Reiser, 2005]. Die Öffnung von Benutzer- und Ressourcengruppen bedeutet deshalb für das VO-Management die Einführung eines effektiven und effizienten Trustmanagements, das auch die Anonymität von Ressourcen und Nutzern unterstützt. Ganz allgemein stellt sich die Frage nach der Gewährleistung von Verlässlichkeit von VOs im Sinne [Avižienis u. a., 2001]: Wie können Ressourcen und Nutzer integriert werden, die weder „gridifiziert“ sind noch die „Gepflogenheiten“ der VO kennen?

Managementaspekte. VO-Management findet heute überwiegend in der Betriebsphase statt und fokussiert auf die Beherrschbarkeit der innerhalb einer VO gegebenen Mitgliedschaftsdynamik. Es ist jedoch immanenter Bestandteil des Grid-Konzeptes, dass sowohl Nutzer als auch Ressourcen zu mehreren VOs gleichzeitig gehören können [Foster u. a., 2001]. Dies induziert nicht nur die Notwendigkeit eines VO-übergreifenden Managements im selben Grid (siehe auch Bemerkung 1 zur Fortpflanzung von Wirkungen auf Seite 8), sondern auch eines VO-Managements, das Grid-übergreifend agiert (Inter-Grid). Dazu wird es erforderlich sein, VOs generisch zu modellieren und adäquate Managementarchitekturen im Sinne [Hegering u. a., 1999] über offene Standards bereitzustellen.

VO-Mitgliedschaft. Mitgliedschaften in VOs sind heute fast immer (quasi-)statisch, da die den VOs zugrunde liegenden Projekte im Rahmen einer weitgehend statischen Projektstruktur langfristig angelegt sind und in der Regel eine geschlossene Benutzergruppe voraussetzen. Dynamische Mitgliedschaften treten dann auf, wenn eine VO als Folge von Fluktuationen (Ressourcen, Dienste, Mitglieder) oder anderer wechselnder Randbedingungen (zum Beispiel Verlust der Vertrauenswürdigkeit) re-konfiguriert werden muss. Dabei ist jeweils zu klären, welche Managementdienste für die Unterstützung der Dynamik bereitgestellt werden müssen und welche Eigenschaften die neuen Mitglieder besitzen müssen, um in die VO aufgenommen werden zu können. Eine wesentliche Fragestellung betrifft die Überwachung des Status einer VO-Mitgliedschaft (Monitoring). Nach welchen Kriterien kann/soll/muss/darf eine Mitgliedschaft beendet werden? Mit welchen Konsequenzen?

Kooperationsstruktur. Schon heute werden in diversen Grid-Projekten die klassischen Kooperationsstrukturen der Lieferkette, der Sternstruktur und der Peer-to-Peer-Kooperation in unterschiedlichen Ausprägungen verwendet. Obwohl zwar nach wie vor nicht diskutiert wird, welche Struktur unter welchen Umständen die beste ist, muss das VO-Management die Transformation von einer Struktur zu einer anderen im Rahmen von Re-Konfigurierungsmaßnahmen unterstützen.

Administration. VOs werden heute noch weitgehend „auf Zuruf“ administriert. Eine geeignete Toolunterstützung setzt sich nur zögerlich für Einzelaspekte (zum Beispiel im Rahmen des VO Membership Service (VOMS) [Alfieri u. a., 2003]) durch. Obwohl eine vollständige Automatisierung des VO-Managements ausgesprochen visionär ist, ist es dennoch das Ziel, zumindest einige Basisdienste und einen effizienten Aggregationsmechanismus bereitzustellen, der durchaus Aspekte von Context-Awareness [Schiffers, 2004] aufweist. Zu klären ist aber, welche Tools benötigt werden und wie sie sich in eine Gesamtarchitektur zum VO-Management integrieren lassen. Dazu muss spezifiziert werden, welche Schnittstellen zur Verfügung gestellt werden müssen, welche Parameter zu berücksichtigen sind und wie diese über Grid-Managementdienste erhoben und kommuniziert werden können.

Diese Überlegungen führen zusammengefasst zu der Notwendigkeit einer Managementarchitektur, in der die zu managenden Ressourcen Virtuelle Organisationen sind und VO-bezogene Managementinformationen über standardisierte Grid-Servicemechanismen ausgetauscht werden. Der Intention allgemeiner Managementarchitekturen [Hegering u. a., 1999] folgend, wird mit einer solchen VO-Management-Architektur (VOMA) ein Rahmenwerk für das Management Virtueller Organisationen festgelegt. Dieses regelt zwar in erster Linie die Modellierung und Beschreibung der Information, die zu Managementzwecken auszutauschen ist, definiert aber auch die Zuständigkeitsbereiche aller am Managementprozess beteiligten Rollen, die Protokolle für den Zugriff auf VO-Managementinformationen und die möglichen Managementfunktionalitäten. Sie bildet damit die konzeptionelle Basis für den Entwurf VO-spezifischer Managementanwendungen und -plattformen in Grid-Umgebungen. Die folgenden Teilfragestellungen dienen der weiteren Vertiefung der Gesamtproblematik und der Beantwortung der Frage, ob die zur Zeit verfügbaren Grid-Mechanismen für das Management Virtueller Organisationen ausreichen oder ob die derzeitigen Konzepte erweitert werden müssen.

- **Wie können Virtuelle Organisationen *aus Managementsicht* beschrieben werden und wie können diese Beschreibungen im Rahmen Grid-spezifischer Protokolle angesprochen werden?**
  Am Management Virtueller Organisationen sind auf unterschiedlichen Ebenen diverse Akteure beteiligt, die allerdings a priori keinerlei gemeinsames Verständnis über die Information besitzen, die zur Lösung von *VO*-Managementaufgaben ausgetauscht werden muss. Ein nicht unerhebliches Hindernis stellen in diesem Zusammenhang die aus den Grid-Grundprinzipien „Co-Management“ und „Virtualisierung“ resultierenden Effekte dar. Insofern liegt die Hauptproblematik darin, die Informationsmodellierung so festzulegen, dass eine Abbildung des Managementobjektes „Virtuelle Organisation“ mit seinen virtuellen Ressourcen und Diensten auf (lokale) Objekte realer Managementarchitekturen ermöglicht wird. Diese Fragestellung wird im Abschnitt 5.2 behandelt.

- **Welche Rollen sind am Lebenszyklus einer VO wie beteiligt?**

Die am Management des Lebenszyklus einer VO beteiligten Akteure, deren Rollenverteilung und die Grundprinzipien ihrer Kooperation im Rahmen wechselseitiger Interaktionsbeziehungen bedürfen einer allgemeinen Festlegung. Damit rückt die Analyse aufbau- und ablauforganisatorischer Fragen in den Vordergrund. Diesem Fragenkomplex ist Abschnitt 5.3 gewidmet.

- **Welche Kommunikationswege und -vorgänge sind für das Management Virtueller Organisationen notwendig und wie können diese etabliert werden?**
  Für alle im Organisationsmodell identifizierten Rollen muss dargestellt werden, wie die Kommunikationsbeziehungen zwischen ihnen ausgeprägt sind und wie der Austausch VO-spezifischer Managementinformation zu geschehen hat. Die Hauptaufgabe wird also darin bestehen, die erforderlichen Kommunikationsprotokolle zu beschreiben. Dies geschieht im Abschnitt 5.4.

- **Welche Funktionalitäten sind für das Management Virtueller Organisationen erforderlich?**
  Im Rahmen der VOMA-Spezifikation müssen diejenigen Managementfunktionalitäten identifiziert und strukturiert werden, die an einer VO-Managementschnittstelle erbracht werden sollen (und können). Das Ziel ist der Entwurf eines „Baukastens“ generischer VO-Managementdienste, die es erlauben, VO-Managementanwendungen zielorientiert zu konzertieren. VO-Managementfunktionalitäten werden im Abschnitt 5.5 diskutiert.

- **Wie kann eine VO-Managementplattform in Grids realisiert werden?**
  VOMA stellt konzeptionell ein Rahmenwerk zur Verfügung, das noch keine Implementierung für einen Einsatz in Grids impliziert. Eine Umsetzung der VOMA-Konzepte in Grids ist jedoch nicht nur wünschenswert, sondern zur Unterstützung der Nachhaltigkeit von Grid-Infrastrukturen sogar notwendig. Ein entsprechendes Trägersystem für VO-Managementlösungen im Kontext existierender Web Services/Grid-Standards wird deshalb gefordert. Kapitel 6 befasst sich mit diesem Aspekt.

- **Wie können VO-Managementlösungen in die betriebliche Praxis der an VOs beteiligten (realen) Organisationen und des Grid-Managements eingebracht und eingesetzt werden?**
  Das Management Virtueller Organisationen kann nicht nur aus Sicht des Architekten oder des Entwicklers von Diensten und Werkzeugen betrachtet werden. Vielmehr müssen auch die „Betreiber“ Virtueller Organisationen und die beteiligten Ressource- bzw. Dienst-„Provider“ in ein Gesamtkonzept integriert werden, ist doch die nahtlose Einbindung der VO-Managementlösungen in deren betriebliche Abläufe von entscheidender Bedeutung für ein effizientes und effektives Grid-Management. Kapitel 7 adressiert diese Frage näher.

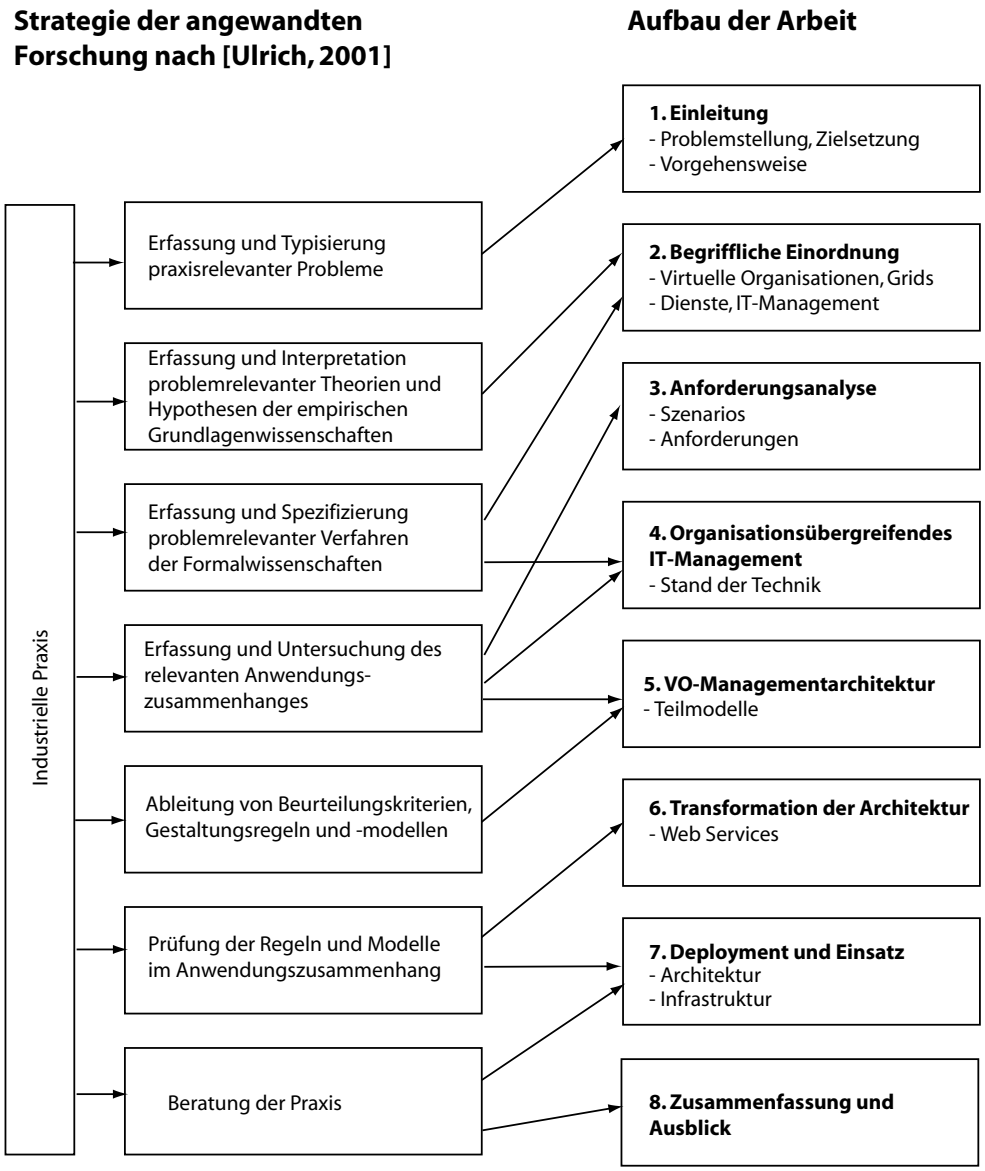


Abbildung 1.4: Vorgehensmodell dieser Arbeit

1.3 Vorgehensmodell

Diese Arbeit ist in erster Linie praxisorientiert und daher eher der angewandten Forschung zuzuordnen. Dies drückt sich insbesondere in der Vorgehensweise aus, die sich methodisch an [Ulrich, 2001] anlehnt. Sie ist in Abbildung 1.4 dargestellt.

Zunächst wird in Kapitel 2 die für diese Arbeit notwendige Terminologie aufgebaut. Dazu werden – ausgehend von einem *allgemeinen* Dienst-, Organisations- und Managementverständnis – der *spezielle* Dienstbegriff, Organisationsbegriff und Managementbegriff in Grids charakterisiert. Dieser Schritt ist notwendig, da sich in Grids bisher nur eine la-

xe Begriffsbildung für den Problembereich „Management dynamischer VOs“ abzeichnet und diese wiederum mit Ansätzen anderer Disziplinen (z.B. Organisationstheorie) nicht kompatibel ist. Insbesondere wird in diesem Kapitel ein Klassifikationsschema für Virtueller Organisationen vorgestellt, das später als *eine* Basis für die Definition von VO-Managementdiensten dienen wird.

Die vorher erarbeitete Taxonomie steuert im Kapitel 3 die Auswahl geeigneter Szenarios zur Anforderungsanalyse. Die gewählten Beispiele orientieren sich dabei an aktuellen und geplanten Grid-Projekten mit dem Ziel, einen umfangreichen Anforderungskatalog für eine VO-Managementarchitektur abzuleiten. Der Anforderungskatalog induziert gleichzeitig ein Einordnungsschema zur Beurteilung der Tauglichkeit existierender Architekturansätze für ein VO-Management in Grids. Die Ableitung der Anforderungen erfolgt über die Analyse von Anwendungsfällen (*use cases*).

Im Kapitel 4 werden relevante Architekturansätze in Grids und des Web Services-Umfeldes auf ihre Eignung für das Management dynamischer Virtueller Organisationen anhand des im Kapitel 3 aufgestellten Anforderungskataloges analysiert. Betrachtet werden dabei neben allgemeinen Ansätzen, wie sie von Standardisierungsgremien vorgeschlagen werden, projektspezifische Vorschläge zum VO-Management. Das Ergebnis dieses Kapitels wird eine Einschätzung sein, welche Ansätze ganz oder in Teilen zur Lösung der Aufgabenstellung dieser Arbeit beitragen können und wo Lücken zu überbrücken sind. Es wird sich dabei zeigen, dass sämtliche aktuellen Managementansätze nur rudimentär den Anforderungskatalog des Kapitels 3 erfüllen, zur Lösung von Teilaspekten aber durchaus herangezogen werden können.

Die Hauptschwierigkeiten bei der Festlegung eines Rahmenwerkes zur Behandlung Virtueller Organisationen in Grids (und damit einer VO-Managementarchitektur) sind in den für Grids typischen dynamischen Organisationskonzepten und den Virtualisierungsansätzen begründet. Im Kapitel 5 wird deshalb mit VOMA eine Architektur vorgeschlagen, die diesen Anforderungen Rechnung trägt und die im Kapitel 3 identifizierten Lücken schließt. Dazu wird im Informationsmodell der Architektur das Managementobjekt „Virtuelle Organisation“ spezifiziert. Zusammen mit einem auf das Management Virtueller Organisationen ausgerichteten Kommunikationsmodell, einem entsprechenden Organisationsmodell und einem dedizierten Funktionsmodell wird mit VOMA ein *Rahmenwerk* für den Entwurf von VO-Managementlösungen vorgegeben. VOMA wird dabei als Plattform-unabhängiges Modell nach dem Prinzip der Model Driven Architecture (MDA) entworfen, um die Wiederverwendbarkeit des Modells zu gewährleisten.

Die in Kapitel 5 vorgeschlagene Plattform-unabhängige Managementarchitektur wird im Kapitel 6 mit Hilfe einer entsprechenden Transformation auf ein Plattform-spezifisches Modell abgebildet. Dies wird am Beispiel des Web Services Distributed Managements (WSDM) dargestellt.

Die in den Kapiteln 5 und 6 vorgestellte Managementarchitektur bildet mit ihren Teilmodellen die *entwicklungs*technische Basis für VO-orientierte Managementanwendungen und -werkzeuge. Das Management Virtueller Organisationen darf jedoch nicht nur aus Sicht

des Architekten oder des Entwicklers betrachtet werden. Vielmehr müssen auch die „Betreiber“ Virtueller Organisationen und die beteiligten Ressourcen- bzw. Dienst-Provider in ein Gesamtkonzept integriert werden. Kapitel 7 adressiert deshalb diese Sicht und untersucht Fragen der Integrierbarkeit in bestehende organisatorische Strukturen und Abläufe. Als Ergebnis liefert das Kapitel eine VO-Management-Infrastruktur und eine darauf ausgerichtete entsprechende Deployment-Methodik.

Kapitel 8 enthält eine Zusammenfassung der erarbeiteten Ergebnisse und streift in einer kurzen Diskussion weiterführende Forschungsfragestellungen, die die Arbeit abrunden.

Die detaillierten Zusammenhänge der einzelnen Kapitel sind in Abbildung 1.5 noch einmal zur Übersicht wiedergegeben.

1.4 Abgrenzung zu verwandten Arbeiten

Facetten des Managements dynamischer Virtueller Organisationen werden in einigen Arbeiten sowohl innerhalb des Munich Network Management (MNM)-Teams als auch im Rahmen anderer akademischer und industrieller Forschungsprojekte behandelt. Im Folgenden wird kurz auf die Gemeinsamkeiten, Unterschiede und Schnittstellen zu denjenigen Arbeiten eingegangen, die in einem deutlich engeren Zusammenhang mit dieser Arbeit stehen als die übrige Literatur, die im Kapitel 4 positioniert wird.

Michael Brenner untersucht in seiner Dissertation [Brenner, 2007] die generische Tool-Unterstützung für IT Infrastructure Library (ITIL)-konforme Prozesse. Die Tool-Unterstützung im Rahmen des Managements einer VO kann als Spezialfall dieser Fragestellung verstanden werden. Insofern haben beide Arbeiten zwar Gemeinsamkeiten, komplementieren sich jedoch.

Wolfgang Hommel adressiert in seiner Dissertation [Hommel, 2007] Identity Management-Konzepte in föderierten Umgebungen (FIM). Die in dieser Arbeit behandelten Fragestellungen profitieren von den Ergebnissen Hommels insofern, als sie die hier betrachteten Management-Dienste um eben diese Komponenten ergänzen. Identity Management ist nicht Gegenstand dieser Arbeit.

Martin Sailer diskutiert in seiner Dissertation [Sailer, 2007] die spezielle Fragestellung der Adäquatheit von Informationsmodellen für das Management von IT-Services. Natürlich ist für das Anliegen der hier vorliegenden Arbeit ein standardisiertes Informationsmodell für VOs notwendig, das auf Konzepten des Sailerschen Modells aufbauen wird.

Vitalian Danciu untersucht in seiner Dissertation [Danciu, 2007] die Abbildung komplexer, auch inter-organisationaler, Prozesse auf Policies. Der Zusammenhang mit dieser Arbeit liegt in der Prozess-Orientierung. Anders als bei Danciu liegt der Fokus dieser

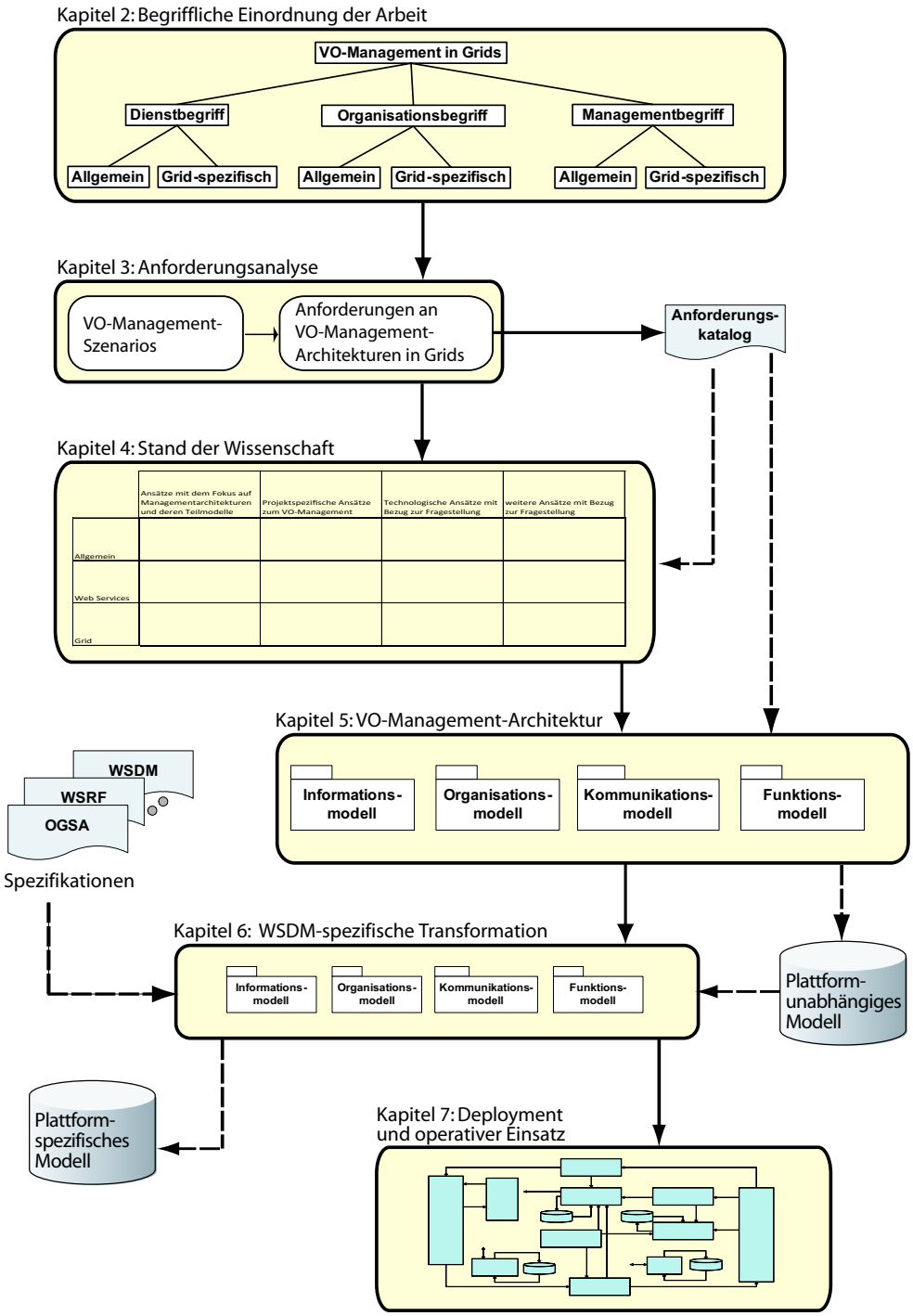


Abbildung 1.5: Zusammenhänge der Kapitel dieser Arbeit

Arbeit jedoch auf der Anwendbarkeit der dort studierten Fragestellung. Insofern kann diese Arbeit in Einzelfragen von den Ergebnissen Dancius profitieren.

Alexander Keller untersucht in [Keller, 1998] CORBA-basiertes Enterprise Management. Keller betrachtet zwar Formen von domänen-übergreifendem IT-Management, diskutiert allerdings weder das Co-Management-Problem noch ein *organisations*übergreifendes IT-*Service*-Management. Dies sind jedoch die Kernanforderungen dieser Arbeit.

Jürgen Hartmut Koch untersucht in [Koch, 2003] die Formation virtueller Communities und schlägt ein System zur Beschreibung solcher Communities vor. Koch betrachtet – im Gegensatz zum Ansatz dieser Arbeit – jedoch ausschließlich Communities als Interessensgruppen soziologischen Gefüges, technische Ressourcen und Dienste sind nicht Bestandteil solcher Communities. Dennoch bildet der von Koch vorgeschlagene Modellierungsansatz einen interessanten Startpunkt zur Beschreibung Virtueller Organisationen.

Michael Langer und Michael Nerb beschreiben in ihren Dissertationen [Langer, 2001] bzw. [Nerb, 2001] einen Architekturansatz für interorganisationales Customer Service Management. Die dort vorgestellten Architekturprinzipien und Vorgehensweisen werden in dieser Arbeit aufgegriffen und an die hier zu diskutierenden Fragestellungen angepasst.

D-Grid adressiert VO-Management vor dem Hintergrund bereits existierender VOs und – zumindest für die erste Projektphase – weitgehend statischer Mitgliedschaften [Gentzsch, 2005]. Das im Rahmen des Projektes zu erstellende VO-Rahmenkonzept kann jedoch durchaus in Teilen für die Betrachtungen dieser Arbeit adaptiert werden.

1.5 Ergebnisse der Arbeit

Vor dem Hintergrund der hier adressierten Fragestellungen können die wichtigsten Ergebnisse dieser Arbeit wie folgt zusammengefasst werden:

1. Die Arbeit liefert eine auf das Management Virtueller Organisationen in Grids ausgerichtete Nomenklatur als begrifflichen Kontext.

2. Die Arbeit liefert einen umfangreichen Anforderungskatalog für eine VO-Managementarchitektur in Grids. Der Katalog wird systematisch aus realen Grid-Szenarios und einem generischen VO-Modell abgeleitet.

3. Die Arbeit liefert eine Analyse des Status Quo zum Management Virtueller Organisationen in Grids.

4. Die Arbeit liefert eine komplette Plattform-unabhängige und wiederverwendbare VO-Managementarchitektur mit allen erforderlichen Teilmodellen.
5. Es wird aufgezeigt, wie diese Architektur auf Plattform-spezifische Umgebungen abgebildet werden kann.
6. Die Arbeit liefert eine auf Konfigurationsmustern basierende Deployment-Strategie zur Nutzung der Architektur.
7. Die Arbeit liefert schließlich eine VO-Management-Infrastrukur, auf der die Architektur im Rahmen von Managementanwendungen einsetzbar ist.

Kapitel 2

Begriffliche Einordnung und Grundlagen

Inhalt des Kapitels

Die in dieser Arbeit erstmals vorgeschlagene Betrachtung dynamischer Virtueller Organisationen als *managed objects* (MO) muss sich damit auseinander setzen, dass bisher keine einheitliche Begrifflichkeit zu dieser Problematik vorliegt. Vielmehr sind die im thematischen Umfeld verwendeten Terminologien häufig inkonsistent und zum Teil sogar widersprüchlich. In diesem Kapitel werden deshalb die für die folgenden Abschnitte relevanten Begriffe eingeführt, mit dem Ziel, neben einer durchgängigen Begriffsbildung auch eine saubere Einordnung des Status Quo in Kapitel 3 zu gewährleisten.

Der begriffliche Kontext der Arbeit ist in Abbildung 2.1 skizziert. Es wird deutlich, dass eine einheitliche Terminologie eine Integration organisationstheoretisch relevanter Definitionen (Bereich ❶) mit der Nomenklatur serviceorientierter Architekturen (Bereich ❷), speziell des Grid-Umfeldes (Bereich ❹), und den im IT-Management verwendeten Konzepten (Bereich ❸) erforderlich macht. Dies ist das Ziel dieses Kapitels.

Im Abschnitt 2.1 werden – ausgehend von einem eher systemtheoretisch geprägten Organisationsverständnis [Girod u. a., 2005] – Virtuelle Organisationen sowohl allgemein als auch Grid-spezifisch diskutiert. Es sei jedoch an dieser Stelle angemerkt, dass dieser Abschnitt weder als Einführung in die Organisationstheorie noch als solche in Verfahren der Organisationsmodellierung zu verstehen ist, dies würde den Rahmen der Arbeit sprengen. Der an weiterführenden Fragestellungen interessierte Leser sei stattdessen auf die einschlägige Literatur verwiesen, zum Beispiel [Vahs, 2005], [Daft, 2006] oder [Rolstadås u. Andersen, 2000]. In diesem Abschnitt geht es vielmehr um eine *Formalisierung* des Organisationsbegriffes aus der Sicht der Informatik, soweit sie für diese Arbeit von Bedeutung ist. Dass dabei auch Erkenntnisse aus anderen Wissenschaftsdisziplinen einfließen werden, liegt in der Natur der Fragestellung begründet.

Virtuelle Organisationen, wie sie hier betrachtet werden, erbringen Dienste über einer dienstorientierten Architektur (den Grids) und sind durch diese wahrnehmbar. Ein sauberes Dienstverständnis ist daher unumgänglich. Dieses wird im Abschnitt 2.2 geliefert. Der Fokus wird dabei auf den an der Diensterbringung beteiligten Rollen und den statischen und dynamischen Aspekten von Diensten liegen. Zur Präzisierung der Terminologie wird mit dem generischen MNM-Dienstmodell [Garschhammer u. a., 2001b] ein Referenzmodell verwendet. Die Abbildbarkeit des generischen Dienstbegriffes in dienstorientierte Architekturen ist ein zusätzliches Anliegen dieses Abschnittes.

Der Fokus dieser Arbeit liegt auf dem *Management* Virtueller Organisationen in Grids. Es ist daher festzulegen, was unter „Management“ zu verstehen ist. Da hier ein integrierter Managementansatz im Sinne [Hegering u. a., 1999] angestrebt wird, ist es nur konsequent, im Abschnitt 2.3 die wesentlichen Grundbegriffe integrierten IT-Managements im

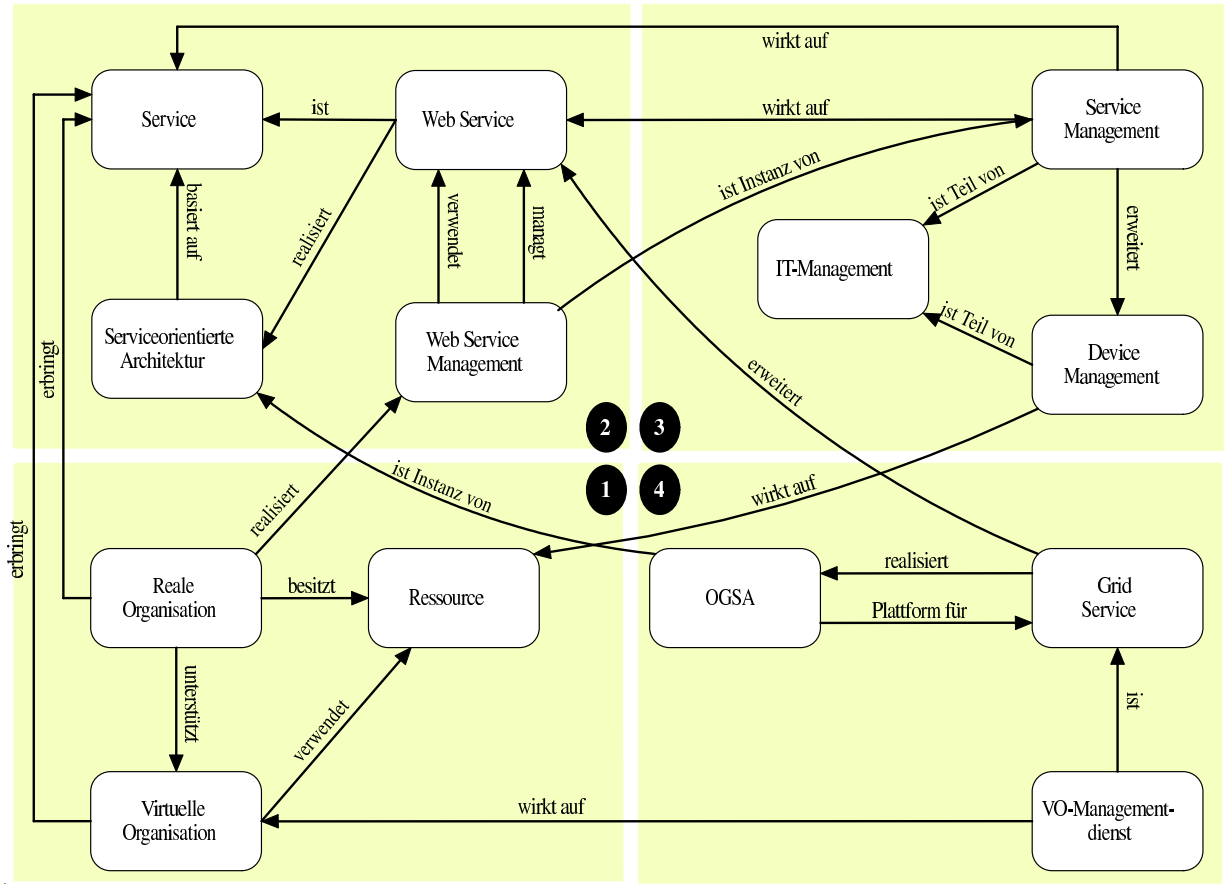


Abbildung 2.1: Begriffliche Zusammenhänge dieser Arbeit

Allgemeinen und des IT-Service Managements nach [Dreo Rodošek, 2002; Lewis, 1999] im Besonderen kurz anzureißen und anschließend mit den Rahmenwerken des Web Services Distributed Management (WSDM)-Standards und des Web Services Management (WSM)-Standards zu vergleichen. Letztere sollen prinzipiell die Basis eines Plattform-spezifischen Realisierungsansatzes bilden.

2.1 Der Aspekt „Virtuelle Organisation“

Virtuelle Organisationen bilden in Grids ein fundamentales Konzept, um organisationsübergreifend komplexe IT-Lösungen, wie sie beispielhaft in [Foster u. Kesselman, 2004b] und [Berman u. a., 2003] beschrieben sind, bereitzustellen. Sie ermöglichen Gruppen von (realen) Organisationen, Teilorganisationen und/oder Individuen die kontrollierte Verwendung von Ressourcen oder Diensten, um ein gemeinsames Ziel auf der „Geschäftsgrundlage“ einer *kooperativen* Zusammenarbeit für die Dauer der Existenz der VO zu erreichen.

Das Konzept „Virtuelle Organisation“ an sich ist nicht neu, wird es doch seit vielen Jahren in der Soziologie, den Wirtschaftswissenschaften, der Organisationstheorie, in den ingenieurwissenschaftlichen Disziplinen, aber auch in der Informatik studiert [Camarinha-Matos u. a., 2005]. Der Blickwinkel der Betrachtung ist jedoch primär disziplinspezifisch. So stellen beispielsweise Soziologie, Psychologie und Politologie eher einen *institutionellen* Organisationsbegriff (etwas *ist* eine Organisation) auch für VOs in den Vordergrund (vergleiche auch die Charakterisierung von virtuellen Communities in [Koch, 2003]). Die Wirtschaftswissenschaften favorisieren dagegen einen *instrumentellen* Organisationsbegriff (etwas *hat* eine Organisation), indem sie den Aspekt des wirtschaftlichen Erfolges, wie er beispielsweise in Lieferanten- und Distributionsnetzen seit Jahren angestrebt wird [Kürümlüoglu u. a., 2005], betonen. Sie sprechen deshalb auch vorzugsweise von virtuellen *Unternehmen* (*virtual enterprise*). Die eher technisch und naturwissenschaftlich orientierten Disziplinen wie das Ingenieurwesen oder die Medizin denken dagegen vornehmlich in Projektgruppen und Konsortien auf der Basis eines hybriden Organisationsverständnisses [Shtub u. a., 2004]. Auch in der Informatik sind Virtuelle Organisationen (im weiteren Sinn) nicht unbekannt, wie Arbeiten zu Prozessgruppen in Verteilten Systemen [Tanenbaum u. van Steen, 2002], zu Lernstrategien von Agenten in der Künstlichen Intelligenz [O'Leary u. a., 1997], zur Organisation von Agentensystemen [Weiss, 2000], im Groupwarebereich [Borghoff u. Schlichter, 2000] oder auch im IT-Management [Hegering u. a., 1999; Keller, 1998; Langer, 2001] beispielhaft zeigen. Die Sichtweise der Informatik ist allerdings eher geprägt von Rollen und deren Beziehungen als formale Relationen.

Trotz unterschiedlicher Facetten im Verständnis Virtueller Organisationen, ist dennoch eine generelle Orientierung an einem *Organisations*gedanken im Sinne des „Organisierens“ und des „Organisiertwerdens“ auszumachen. Damit rücken – wie in realen Organisationen auch – Strukturdiskussionen sowie Prozess- und Abstimmungsfragen in den Fokus, allerdings in einem erheblich dynamischeren Kontext.

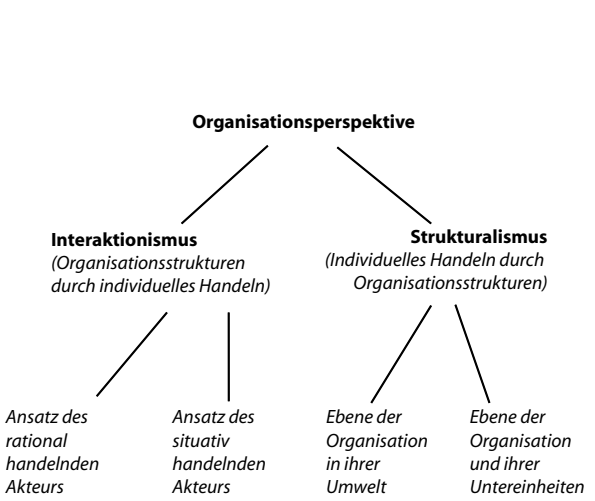
2.1.1 Generelle Anmerkungen zum Organisationsbegriff

Der Begriff „Organisation“ lässt sich vom Wortursprung (lateinisch *organum* bzw. griechisch *organon*) am treffendsten mit „Werkzeug“ oder „Hilfsmittel“ übersetzen und meint damit prinzipiell ein geordnetes Zusammenspiel von Elementen im Rahmen der Planung und Durchführung eines Vorhabens. „Organisation“ wird deshalb auch häufig als Antonym zum unbestimmten Chaos gesehen. Allein aus dieser Etymologie lassen sich schon erste grundsätzliche Organisationsmerkmale wie Mitglieder, Rollen, Lebensdauer, Struktur, Aktivitäten, Prozesse, Zielorientierung oder Koordination kanonisch ableiten.

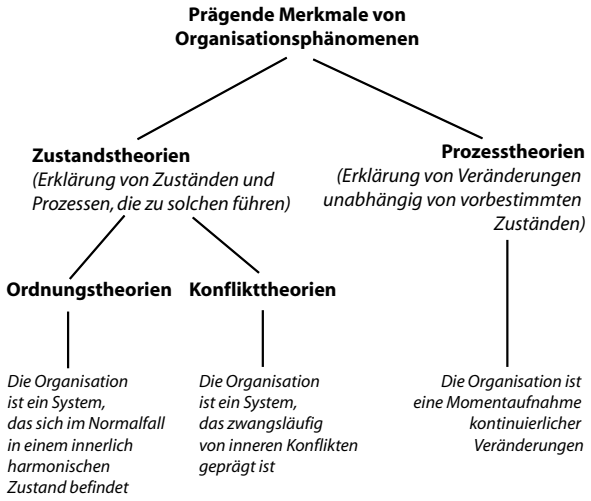
In der Wissenschaft wird der Organisationsbegriff sehr vielschichtig verwendet. Durchgängig anerkannt ist jedoch die Auffassung von Organisationen als „Zielerreichungssysteme“ [Carley, 1995; Thompson u. Zald, 2003], unabhängig davon, ob eine institutionelle oder instrumentelle Organisationsauffassung zu Grunde gelegt wird. Im ersten Fall werden nämlich Organisationen als zielgerichtete (soziotechnische) Systeme verstanden, im zweiten

Fall als Regelsysteme, die die Aufgabenerfüllung zielgerichtet und dauerhaft ordnen.

Jede Beschäftigung mit einem bestimmten Organisationsaspekt erfordert daher zunächst die Klärung dieser Grundperspektive als Ausgangspunkt aller weiteren Betrachtungen. Dazu ist erstens nach dem eigentlichen *Kern* des zu untersuchenden Organisationsphänomens und zweitens nach dessen prägenden *Merkmalen* zu fragen. Die Auseinandersetzung mit dem Kern des Organisationsphänomens führt in der Regel entweder zu einem *interaktionsorientierten* Organisationsbegriff, weil die Organisation als Produkt einzelner Interaktionsbeziehungen zwischen den Akteuren angesehen wird, oder zu einem *strukturorientierten* Organisationsbegriff, weil der Kern des Organisationsphänomens in den von den Akteuren entkoppelten Strukturen selbst zu sehen ist [Gmür, 1993] (siehe Abbildung 2.2(a)). Die Frage nach den prägenden Merkmalen adressiert dagegen die Grunddisposition hinter den beobachteten organisatorischen Einzelphänomenen. Hier sind zu unterscheiden Ansätze, bei denen von einer bestehenden „Konstellation“ der Organisation ausgegangen wird (die Organisation als Zustand), und Ansätze, bei denen die Organisation in erster Linie als Vorgang verstanden wird – unabhängig von Ausgangs- und Zielkonstellationen (siehe Abbildung 2.2(b)). Beide Dimensionen stehen orthogonal zueinander und ermöglichen damit eine grobe Klassifikation organisationstheoretischer Ansätze, die allerdings für die nachfolgenden Ausführungen nicht relevant ist und deshalb auch nicht weiter verfolgt wird. Dem interessierten Leser sei deshalb [Gmür, 1993] empfohlen.



(a) Organisationsperspektiven



(b) Merkmale von Organisationsphänomenen

Abbildung 2.2: Klassifikationsschema für Organisationstheorien nach [Gmür, 1993]

Der Organisationsbegriff dieser Arbeit wird geprägt durch einen ordnungstheoretischen, strukturbezogenen Ansatz, umfasst aber auch *metaorganisatorische* Aspekte – zum Beispiel Mechanismen zur Selbstorganisation, wie sie in Prozesstheorien behandelt werden (Abbildung 2.2).

Damit stehen strukturelle Mechanismen im Vordergrund, die der Organisation ihre Gestalt verleihen, und nicht die Akteure. Der strukturorientierte Ansatz fordert insbesondere, dass Organisationsstrukturen nicht durch Technologien und IT-Architekturen präjudiziert werden, ganz im Einklang mit [Hegering u. a., 1999]. Dass allerdings der Einzug neuer Informationstechnologien in den industriellen Leistungserstellungsprozess durchaus einen Impetus für organisatorischen Wandel darstellen kann, wird jedoch konstatiert. Eine Untersuchung der wechselseitigen Einflüsse von Organisation und Technologie wird von [Schmidt, 2006] durchgeführt.

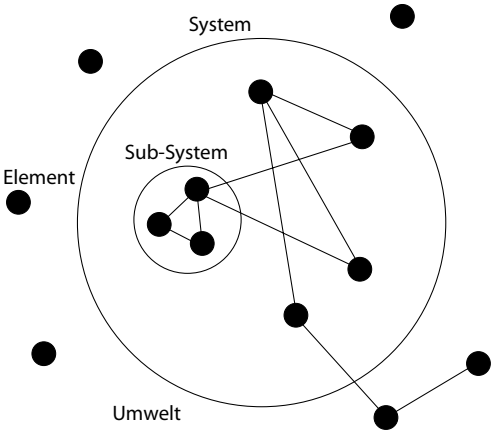


Abbildung 2.3: Allgemeine Darstellung eines Systems nach [Hill u. a., 1994]

Die Position der Informatik orientiert sich in diesem Kontext an einer *Formalisierung* des Organisationsbegriffes. Sie wird deshalb – anders als die Soziologie oder die Organisationstheorie – in erster Linie in Informationsstrukturen, Kommunikationsprotokollen, Koordinationsmechanismen und Metriken zur Bestimmung von Zielerreichungsgraden denken. Im Zentrum eines Informatikansatzes werden folgerichtig Organisations*modelle* stehen, die die geforderte Formalisierung leisten können und eine systematische und möglichst automatisierte Behandlung von organisationsspezifischen Strukturen und Prozessen weitgehend unterstützen. Im Speziellen wird sich der an IT-Managementfragestellungen interessierte Informatiker auf die IT-gestützte Beherrschung und Beherrschbarkeit[1] von Strukturen, Prozessen und kompletten Organisations*lebenszyklen* unter Zuhilfenahme eben solcher Modelle konzentrieren. Für die Informatik ist es dabei prinzipiell unerheblich, ob die zu Grunde liegende Informationstechnologie als Determinante der Organisationsstruktur oder als durch die Organisation zu gestaltendes Objekt gesehen wird. Anhänger des technologischen Imperativs sehen in der verwendeten Informationstechnologie einen gestaltungsbestimmenden Faktor von Organisationsstrukturen und sind daher tendenziell der interaktionalistischen Auffassung in Abbildung 2.2(a) näher. Die andere Seite hingegen fühlt sich der strukturalistischen Perspektive verwandt und fasst Organisationsstrukturen vornehmlich als Ergebnis kontinuierlichen menschlichen Handelns im sozialen Gefüge einer Organisation auf, in denen IT-Strukturen nur eine unterstützende Rolle spielen [Fulk, 1993].

Der Sichtweise der Informatik kommt ein systemtheoretisch motivierter Organisationsbegriff am nächsten (siehe auch [Koch, 2003]), da er Organisationen als „Systeme“ interpretiert, deren Strukturgefüge mit Mitteln der mathematischen Topologie untersucht werden können (Abbildung 2.3). Organisationen kennen damit Begriffe wie „Grenze“ (*boundary*) als Inkarnation der System/Umwelt-Differenz, „Organisationsmitglieder“ als nicht-leere Menge von Systemelementen sowie „Organisationsprozesse“, ausgeprägt im zielgerichteten Zusammenwirken der Mitglieder in einem funktionellen Sinn. Alles außerhalb der Grenze

[1] *Management* und *Manageability* in der englischsprachigen Literatur.

liegende ist nicht Bestandteil der Organisation, sondern deren *Umwelt*. Die Menge der Elemente und der (möglicherweise leeren) Interaktionsbeziehungen repräsentiert die Struktur einer Organisation, auf der mit der Interaktionsrelation ein kanonisches Distanzmaß induziert werden kann, das offene und abgeschlossene Organisationen differenzierbar macht. Eine offene Organisation ist dadurch gekennzeichnet, dass für mindestens ein Mitglied der Organisation eine Wechselwirkung zu Mitgliedern einer anderen Organisation besteht (wie in Abbildung 2.3). Dies kann beispielsweise eine Interaktion zwischen einem Organisationsmitglied und der unmittelbaren Umwelt der Organisation sein, wenn unter „Umwelt“ die Gesamtheit der Faktoren verstanden wird, die auf eine Organisation von außen einwirken und sie beeinflussen. Bei einer in sich geschlossenen Organisation existieren diese Wechselwirkungen nicht. Sub-Organisationen sind nach dieser Definition immer offen, die Umwelt (selbst aufgefasst als System respektive Organisation) ist dagegen immer abgeschlossen. Systemtheoretisch folgt aus den Änderungen des Systemzustands über die Zeit eine Organisationsdynamik, deren Ausmaß wesentlich von den Veränderungen der Umwelt sowie der Offenheit der Organisation gegenüber dieser Umwelt abhängt [Hill u. a., 1994]. Zu beachten ist allerdings, dass die hier angesprochene Dynamik struktur- und lebenszyklusrelevant ist und die „internen“ Abläufe einer Organisation – wenn überhaupt – nur mittelbar betrifft. In erster Näherung kann damit diese Organisationsauffassung gemäß Abbildung 2.4 modelliert werden.

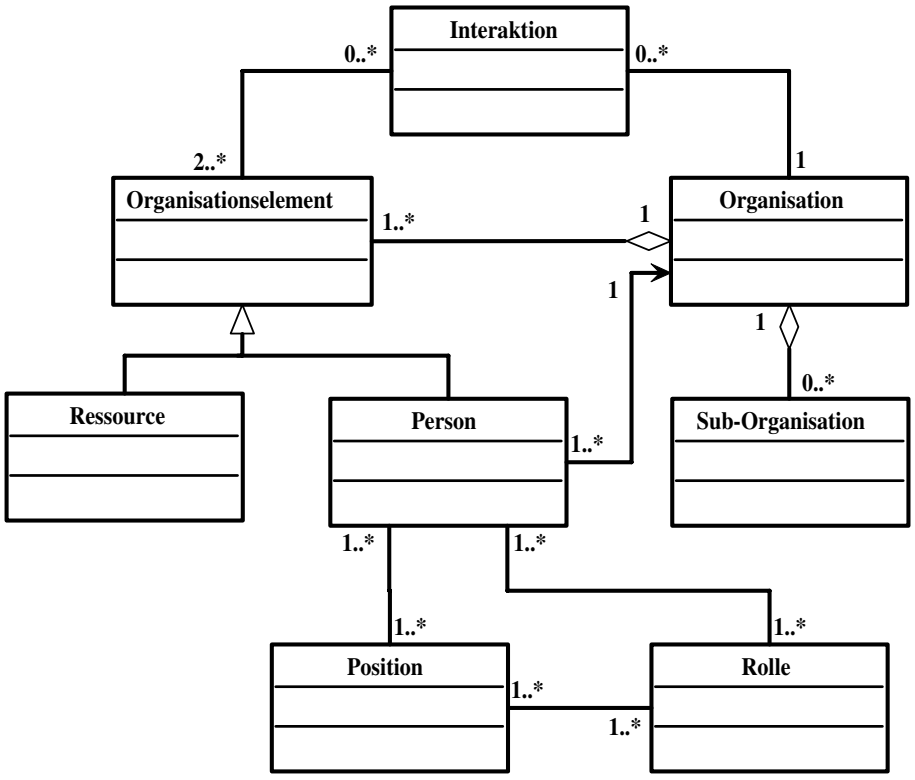


Abbildung 2.4: Vereinfachtes Metamodell einer Organisation

Eine Organisation kennt danach Strukturelemente wie Sub-Organisationen und Organisationselemente, die miteinander in Interaktionsbeziehungen stehen. Organisationselemente können Ressourcen (im weiteren Sinn) oder Personen sein, die Positionen besetzen und Rollen einnehmen. Da Rollen im Gegensatz zu Positionen aufgabenorientiert definiert werden und nicht im Gefüge einer Betriebsablaufsteuerung (siehe auch [Lupu u. Sloman, 1997] und [Schaad, 2003], füllt jede Position mindestens eine Rolle aus und jede Rolle wird von mindestens einer Position wahrgenommen.

2.1.2 Allgemeine Charakterisierung Virtueller Organisationen

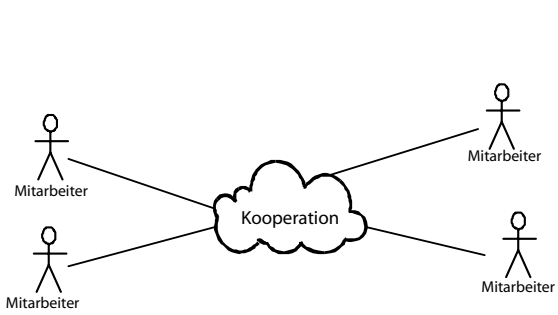
Der Begriff „Virtuelle Organisation“ taucht erstmals 1986 im angloamerikanischen Sprachraum in Analogie zur virtuellen Speicherverwaltung in der Informatik auf [Mowshowitz, 1997]. VOs entstanden zum damaligen Zeitpunkt im Umfeld instrumentell orientierter Organisationen, um den zunehmenden Veränderungen des wirtschaftlichen Wettbewerbsumfeldes (Globalisierung, Verkürzung von Produkt(entwicklungs)zyklen, Wandlung von Anbieter- zu Käufermärkten, Individualisierung von Produkten und Dienstleistungen) Rechnung tragen zu können. Zugleich erlebten Informations- und Kommunikationssysteme deutliche Leistungssteigerungen bei generell sinkenden Anschaffungskosten und wirkten so als „*driving forces*“ einer deutlich erkennbaren Auflösung traditioneller Organisationsstrukturen [Davidow u. Malone, 1993; Garschhammer u. a., 2003] zugunsten *kollaborativer Netzwerke* als neuem Organisationsparadigma. In der Wissenschaft führen analoge Überlegungen zur Zeit zu diversen e-Science-Initiativen, wie sie in [Fox u. Walker, 2003; GridCoord, 2005; Hegering, 2005a] exemplarisch dargestellt werden.

Kooperationen zwischen Organisationen sind kein neues Phänomen. Williamson [Williamson, 1975] argumentierte schon früh für organisationsübergreifende Kooperationen zur Kostensenkung. Auch der Outsourcing-Boom in den 1980er Jahren [Beaumont u. Khan, 2005] induzierte diverse Muster Virtueller Organisationen, zum Teil unterstützt durch Web-/Internet-basierte Werkzeuge, Groupware-Systeme (Computer Supported Cooperative Work) (CSCW) [Borghoff u. Schlichter, 2000; Greif, 1990] und Inter-Enterprise-Applikationen wie Supply Chain Management (SCM) [Zeller, 2003] oder Customer Relationship Management (CRM) [Hippner u. a., 2001]. Neu sind dagegen die Dimensionen, die derartige Kooperationen sowohl quantitativ als auch qualitativ gewonnen haben.

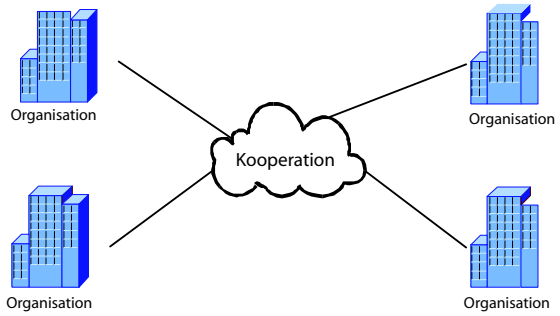
Das Konzept „VO“ wird in einem intra- und einem interorganisatorischen Gestaltungskontext verwendet. Als Prinzip *intra*organisatorischer Gestaltung wird mit dem VO-Konzept das Ziel verfolgt, räumliche und zeitliche Begrenzungen zu überwinden, um die Vorteile des verteilten Operierens, des dezentral verteilten Wissens und der lokalen Präsenz simultan nutzen zu können (Beispiel: Virtuelles Büro). Als Prinzip *inter*organisationaler Gestaltung unterstützt eine VO die institutionelle Organisationsperspektive. Unter dieser Maxime bildet die VO ein kooperatives, flexibles Netzwerk rechtlich selbständiger Organisationen, die Teile ihrer Ressourcen gemeinsam nutzen und ihre jeweiligen Kompetenzen einbringen. So erscheinen die Leistungen einer VO nach außen als eine Einheit, bilden

aber faktisch das Ergebnis eines auf viele Leistungsträger verteilten Leistungserstellungsprozesses. Dies ist auch die Sicht dieser Arbeit. Als „Geschäftsgrundlage“ einer solchen Kooperation dient die Formulierung eines gemeinsamen Ziels. Anders als bei realen Organisationen ist die Lebensdauer einer VO eng an das Erreichen dieses Ziels gekoppelt.

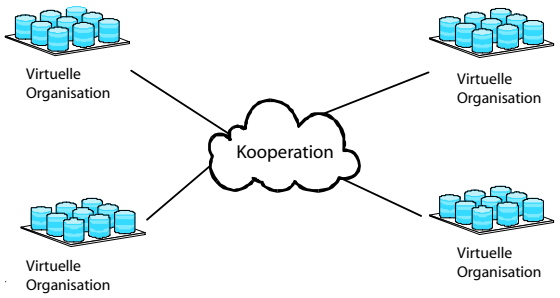
In der Literatur werden zum Teil unterschiedliche Termini in Zusammenhang mit virtuellen Organisationsformen verwendet [Camarinha-Matos u. a., 2005]. Eine Unterscheidung von Virtuellen Organisationen und virtuellen Unternehmen, wie sie in [Scholz, 1994] und [Kürümlüoglu u. a., 2005] vorgenommen wird, wird zwar von den meisten Autoren in dieser Form nicht vollzogen, trägt aber zur begrifflichen Präzisierung und methodisch sauberen Festlegung der Betrachtungsebene bei (siehe Abbildung 2.5). In dieser Arbeit liegt der Fokus vor allem auf der Meso- und der Makroebene von VOs.



(a) Mikroebene: Intraorganisationale Kooperation



(b) Mesoebene: Interorganisationale Kooperation



(c) Makroebene: VO-Kooperation

Abbildung 2.5: Betrachtungsebenen virtueller Organisationsformen nach [Müller, 1997]

Eine eindeutige Bestimmung der zentralen Eigenschaften virtueller Organisationsformen lässt die Literatur zwar nicht zu, trotzdem lassen sich einige Merkmale identifizieren, die eine Integration der verschiedenen Schulen erlauben und sowohl auf der Mikroebene (intra-organisational), der Mesoebene (inter-organisational) als auch der VO-übergreifenden Makroebene vorzufinden sind.

Merkmal der Kooperation. In der Literatur herrscht Konsens darüber, dass die kooperative Zusammenarbeit von Organisationen, im Gegensatz zu kompetitiven Verbünden, ein Charakteristikum Virtueller Organisationen darstellt.

Merkmal der Konzentration auf Kernkompetenzen. In nahezu jeder Publikation zu Virtuellen Organisationen wird die Konzentration der an der VO beteiligten Partner auf ihre jeweiligen Kernkompetenzen herausgestellt. In diesem Zusammenhang betont [Scholz, 1994] die Notwendigkeit der Herausbildung spezifischer komplementärer Kernkompetenzen in den Partnerorganisationen zur Erreichung eines maximalen Wertschöpfungsbeitrages a posteriori.

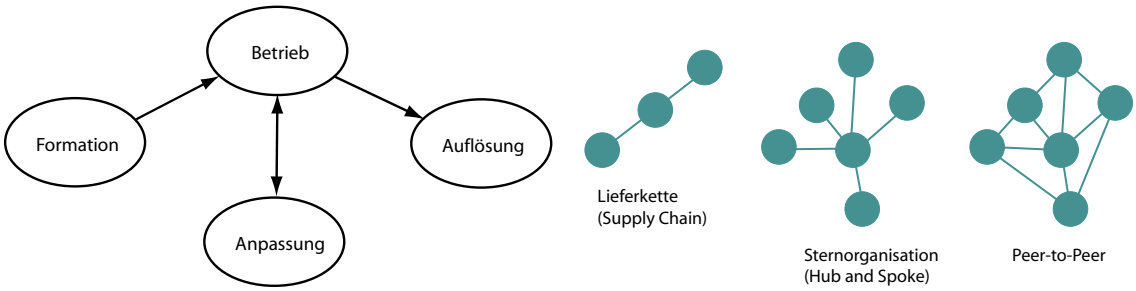
Merkmal der Prozessorientierung. Die Verbindung von Organisationen zu Virtuellen Organisationen und die Konzentration auf Kernkompetenzen im Rahmen der Bildung einer gemeinsamen Wertschöpfungskette verweisen auf die Prozessorientierung Virtueller Organisationen. Betrachtet man VOs nicht nur im institutionellen, sondern auch im instrumentellen Sinn, so lässt sich eine „extreme Dominanz der Ablauf- über die Aufbauorganisation“ [Mertens, 1994] ausmachen. [Scholz, 1994] verweist darauf, dass es sich bei VOs deshalb nicht um statische Gebilde, sondern vielmehr um Organisationen handelt, die für die Dauer ihrer Existenz eine Vielzahl von Prozessphasen, den Lebenszyklus, durchlaufen. Daher lassen sich VOs auch nicht allein in Denkkategorien der klassischen Organisationstheorie verstehen, sondern eher als Geschäftsprozess.

Merkmal der minimalen Institutionalisierung. Ein wesentliches Merkmal Virtueller Organisationen ist der weitgehende Verzicht auf die Institutionalisierung zentraler Managementaufgaben. VOs besitzen keine Linienhierarchien, Organigramme oder Abteilungsstrukturen, sondern werden – wegen des Kooperationsmerkmals – über Kontraktbeziehungen zwischen gleichberechtigten Partnern gesteuert („What replaces hierarchy [in VOs]?“ [Kürümlüoglu u. a., 2005]). Sie verfügen weder über ein gemeinsames juristisches Dach noch über eine gemeinsame Verwaltung. Stattdessen ist die Kooperation in VOs durch ein ausgeprägtes Vertrauensverhältnis zwischen den Akteuren gekennzeichnet.

Merkmal der Lebensdauer und des Lebenszyklus. Virtuelle Organisationen werden allgemein als zeitlich limitierte „Objekte“ gesehen, die für die Dauer ihrer Existenz verschiedene Stufen eines Lebenszyklus durchlaufen. Abbildung 2.6(a) stellt einen solchen Lebenszyklus in seiner generischen Form dar.

Merkmal der Kooperation. Die Kooperation von Organisationen kann nach unterschiedlichen Mustern erfolgen, die häufig zur Typisierung von VOs verwendet werden [Katzy u. a., 2005]. In der Literatur werden die in Abbildung 2.6(b) dargestellten Topologien als Kerntypen akzeptiert.

Aus einer organisatorischen Perspektive beschreiben VO-Typen die primäre Koordinationsstruktur, die die Informations- und Materialflüsse sowie die zu Grunde liegenden Befugnisbeziehungen beeinflusst. So verwenden VOs in der Lieferkettentopo-



(a) VO-Lebenszyklus nach [Camarinha-Matos, 2005]

(b) VO-Typen nach [Katzy u. a., 2005]

Abbildung 2.6: Lebenszyklus und Typisierung Virtueller Organisationen

logie verstärkt Workflowmanagement-Systeme zur Koordination und Prozesskontrolle, während Peer-to-Peer-Topologien auf Selbstorganisationsprinzipien ohne zentrales Management aufgebaut sind. In der Praxis sind allerdings nicht nur diese reinen Formen zu finden, sondern auch kombinierte Ansätze, wenn beispielsweise Vertragsbeziehungen eine sternförmige Topologie aufweisen, Materialflüsse als Lieferketten organisiert sind und Planungsinformationen Peer-to-Peer ausgetauscht werden. Untersuchungen im Rahmen des VOSTER-Projektes[2] zeigen, dass die Peer-to-Peer-Topologie mit Abstand bei den meisten VO-Projekten eingesetzt wird, während die Managementstruktur eher sternförmig ausgelegt ist. Dies erscheint auch sinnvoll, da gemeinsame Ziele nur schwer ohne ein geeignetes zentrales Projektmanagement erreicht werden können [Katzy u. a., 2005]. Es sei darauf hingewiesen, dass dieses Merkmal unabhängig vom Merkmal der minimalen Institutionalisierung (siehe oben) ist.

Merkmal der dynamischen Rekrutierung. Zur Formation Virtueller Organisationen müssen die zur Erledigung der Aufgabe erforderlichen Kernkompetenzen dynamisch aus einem a priori existierenden „Pool von Kompetenzen“ rekrutiert werden (*sourcing*). Dies wird in der Literatur mit dem Herstellen von „*preparedness*“ assoziiert [Tølle u. Bernus, 2003]. Die Begriffsbildung ist allerdings nicht eindeutig: [van den Berg u. a., 2000] verwendet in diesem Zusammenhang die Bezeichnung „*breeding environment*“, [Tølle u. a., 2003] spricht in der Virtual Enterprise Reference Architecture (VERA) vom „*network*“, während [Saabeel u. a., 2002] vom „*universe of modules*“ spricht. Häufig wird dieser Pool auch mit dem Begriff „*community*“ belegt, dem wir uns hier anschließen. Abbildung 2.7 zeigt die Rekrutierung dynamischer VOs konzeptionell.

VOs werden in Communities spontan oder geplant gegründet. Eine grobe Formationstaxonomie ist in Abbildung 2.8 wiedergegeben.

[2] `http://cic.vtt.fi/projects/voster/public.html`

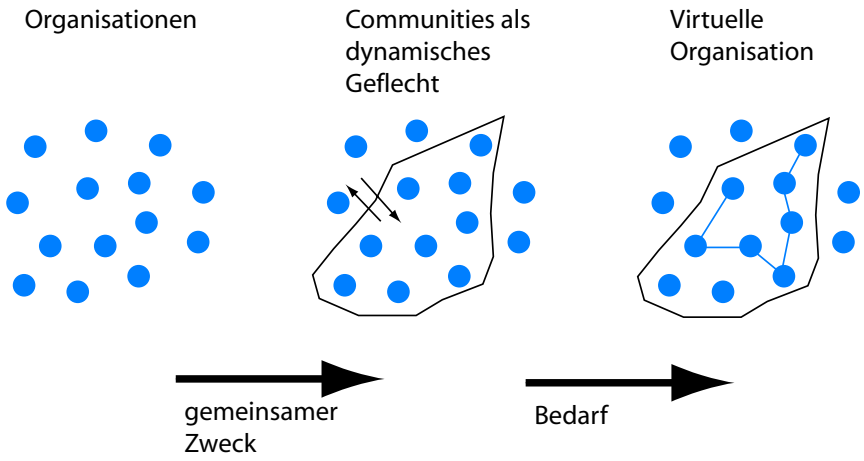


Abbildung 2.7: Formation Virtueller Organisationen nach [Saabeel u. a., 2002]

Merkmal der Informations- und Kommunikationstechnologie. Das wohl prägnanteste Merkmal virtueller Organisationsformen ist der intensive Einsatz von Informations- und Kommunikationssystemen zur räumlichen und zeitlichen Entkopplung arbeitsteiliger Prozesse, wodurch letztlich die virtuelle Präsenz von Organisationen erst ermöglicht wird.

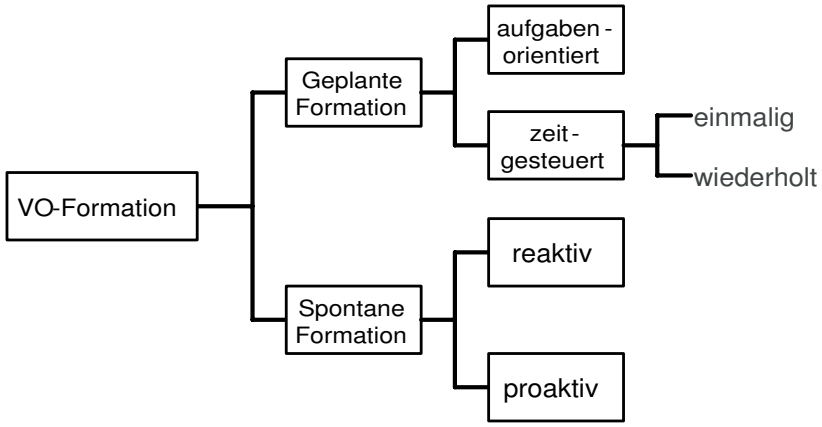


Abbildung 2.8: Taxonomie zur Formation Virtueller Organisationen

Unter Berücksichtigung dieser Merkmale werden – in Analogie zur Abbildung 2.4 – Virtuelle Organisationen gemäß Abbildung 2.9 modelliert. Anders als in Abbildung 2.4 kennen VOs keine Linienorganisation und damit auch keine Positionen. VOs können zwar Sub-VOs enthalten, im Gegensatz zu (realen) Organisationen können diese jedoch durchaus auch Sub-VOs anderer VOs sein. Zusätzlich ist zu beachten, dass VOs und deren Organisationselemente *keine* „*Ownership*-Relation“ bilden, sie werden stattdessen von den (realen) Organisationen für die Dauer deren Existenz an VOs „delegiert“.

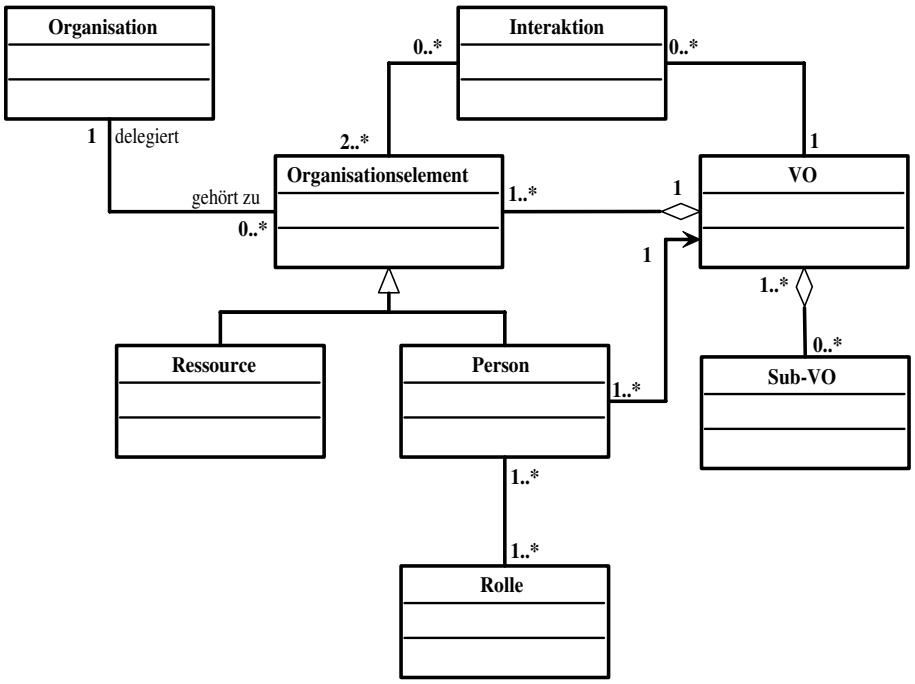


Abbildung 2.9: Vereinfachtes VO-Metamodell

Die vorstehenden Merkmale dienen zwar der Charakterisierung Virtueller Organisationen, eine Klassifikation wie sie für diese Arbeit angestrebt wird, unterstützen sie allerdings nur teilweise. Erforderlich ist vielmehr ein Positionierungsschema, das eine präzisere Attributierung erlaubt. Ein solches wird im Folgenden vorgeschlagen.

Zur Typologisierung von Organisationskooperationen wird in [Bauernhansl, 2003] eine Morphologie angegeben, auf die im Folgenden zur Einordnung Virtueller Organisationen zurückgegriffen wird. Die von Bauernhansl identifizierten Dimensionen sind im oberen Teil der Abbildung 2.10 wiedergegeben. Im Einzelnen sind dies:

Bindungsintensität. Die Bindungsintensität beschreibt den Grad, zu dem sich die beteiligten Parteien einer Organisationskooperation ihre Autonomie, und damit ihre Entscheidungsfreiheit, aufgeben. Sie reicht von der mündlichen Absprache bis hin zur mehrheitlichen Kapitalbeteiligung.

Integrationsgrad. Der Integrationsgrad reicht von Autonomie über die Festlegung gemeinsamer normativer und strategischer Rahmenbedingungen bis hin zur organisatorischen Integration mit einem institutionalisiertem Management, die allerdings für VOs in der Regel nicht zutrifft.

Struktur. Das Strukturmerkmal bezeichnet die Verteilung von Koordinationsbefugnissen. Intraorganisationale Koordination geschieht innerhalb *einer* Organisation, interorganisationale Koordination verteilt sich auf mehrere Organisationen, der Regelfall für VOs.

Entscheidungsreichweite. Die Entscheidungsreichweite bezieht sich auf eine Vereinbarung mit unmittelbarer Auswirkung auf einen oder mehrere Beteiligte eines Netzwerkes oder einer Kooperation, der für VOs typische Fall.

Richtung von Organisationsketten. Unter *horizontalen* Organisationsketten werden Verbünde von Partnern der gleichen Wertschöpfungsstufe bzw. Funktionsschichtung verstanden [Nerb, 2001]. Dagegen bezeichnen *vertikale* Ketten Beziehungen von Partnern auf verschiedenen Ebenen der Wertschöpfungskette bzw. Funktionsschichtung. *Diagonale* Organisationsketten sind Verbindungen mit horizontalen und vertikalen Elementen.

Räumliche Dimension. Organisationskooperationen lassen sich anhand ihrer geographischen Ausdehnung unterscheiden, die von rein lokalen Interaktionen bis hin zu internationalen Konsortien reicht.

Dauer. Die Dauer der Zusammenarbeit in Netzwerken und Kooperationen kann sich ausschließlich auf ein Projekt beziehen oder aber unbefristet ausgelegt sein. Als Zwischenformen sind auch zeitlich bzw. terminlich begrenzte Kooperationen zu finden, die über reine Projektkonstellationen hinausgehen.

Bereich. Die Zusammenarbeit von Organisationen – insbesondere Unternehmen – kann sich auf verschiedenste Bereiche der Wertschöpfungsstufe im Lebenszyklus eines Produktes oder einer Dienstleistung beziehen. Frühe Phasen der Ideenfindung und Detailentwicklung stellen dabei andere Anforderungen an die Partner und deren Koordination als die spätere Leistungserbringung oder gar die kollaborative Kundenbetreuung nach der Leistungserbringung.

Kapazitäten. Die Art der in einem Organisationsnetzwerk zusammengeführten quantitativen Kapazitäten und deren jeweilige Qualitäten können in einer aktivierten Kooperation entweder komplementär oder redundant ausgelegt sein. Die Kombination beider Formen („Komplementärredundanz“) ist von der strategischen Zielsetzung des kooperativen Netzwerks abhängig.

Kompetenzen. Ähnlich wie die Kapazitäten werden auch Kompetenzen abhängig von der Zielsetzung und Aufgabe der Kooperation rekrutiert. Der eher qualitative Charakter als Kombination aus Kapazitäten und Fähigkeiten erschwert zwar eine quantitative Trennbarkeit, Kompetenzen können sich allerdings komplementieren oder sie werden redundant in die Kooperation eingebunden. Komplementärredundanzen sind als Mischformen auch denkbar.

Anzahl der beteiligten Partner. In diesem Merkmal verwischt die Trennung zwischen dem Organisationsnetzwerk selbst und der Kooperation innerhalb des Netzwerkes. In einer rein dyadischen Partnerschaft (häufig auch als *Allianz* bezeichnet) ist eine Unterscheidbarkeit von statischer Vorvernetzung und dynamischer, projektbezogener Interaktion kaum auszumachen. Mit zunehmender Partneranzahl wird eine solche Differenzierbarkeit jedoch evidenter.

Merkmal	Ausprägung					
Bindungsintensität	Absprache		Vertrag		Kapitalbeteiligung	
Integrationsgrad	autonom		koordiniert		integriert	
Struktur	intraorganisational			interorganisational		
Entscheidungs-reichweite	= 1			> 1		
Richtung	horizontal		vertikal		diagonal	
Räumliche Dimension	lokal	regional		national		international
Dauer	projektbezogen		terminlich begrenzt		unbefristet	
Bereich	Forschung & Entwicklung	Beschaffung	Fertigung	Montage	Vertrieb	Service
Kapazitäten	komplementär		redundant		komplementär-redundant	
Kompetenzen	komplementär		redundant		komplementär-redundant	
Anzahl der Partner	dyadisch	triadisch		einfache Netzwerke		komplexe Netzwerke
Koordination	implizit ungeführt			explizit geführt		
Gruppenstruktur	allgemein offen		offen mit Einschränkungen		abgeschlossen	
Gründungsprozess	geplant		spontan ereignisgesteuert		bei Bedarf	
Administration	manuell		Werkzeug-gestützt		automatisch	
Betrachtungsebene	mikro		meso		makro	
Kooperationsstruktur	Lieferkette		Stern		Peer-to-Peer	

Abbildung 2.10: Morphologie von Organisationskooperationen nach [Bauernhansl, 2003] und eigene Erweiterungen gemäß Abbildung 1.2

Koordination. Für die Koordination der Kooperation in Virtuellen Organisationen lassen sich zwei grundsätzlich verschiedene Paradigmen unterscheiden. Die explizitführende Koordination bejaht die Integration einer institutionellen Koordinationsinstanz. Implizit ungeführte Koordinationen hingegen postulieren eine auf rein lokaler Abstimmung basierende Selbstorganisation mit geringer Reichweite der Entscheidungswirkung. In Grid-VOs stellt die erste Ausprägung den Standardfall dar.

Den Diskussionen im Kapitel 1 folgend, zeigt sich, dass für das Ziel dieser Arbeit die angegebenen Merkmale nicht ausreichen. Während die in Abbildung 1.3 dargestellten Dimension „Lebensdauer“, „VO-Lebenszyklus“ und „VO-Mitgliedschaft“ hinreichend direkt oder indirekt abgedeckt sind, gilt dies für andere Dimensionen nicht. Daher wird hier vorgeschlagen, Abbildung 2.10 um die entsprechenden prozessorientierten und Management-affinen Perspektiven zu erweitern. Dies geschieht im unteren Teil der Abbildung. Auch hier seien kurz die Dimensionen vorgestellt:

Gruppenstruktur. Netzwerke von Organisationen bewegen sich – je nach Intention – im Kontinuum zwischen vollständiger Offenheit und strikter Abgeschlossenheit. Je offener ein Netzwerk wird, desto größer wird der Aufwand, das Netzwerk effizient und effektiv zu schützen und geeignete Vertrauensmechanismen zu etablieren.

Gründungsprozess. Kooperative Netzwerke von Organisationen – und damit auch Virtuelle Organisationen – sind einem Ziel verpflichtet und werden daher im Rahmen einer Aufgabe *geplant* gebildet, können sich aber auch – beispielsweise in Notfallsituationen – *spontan* formieren (siehe auch Abbildung 2.8). Je dynamischer der Gründungsprozess ausgelegt wird, desto höher wird in der Regel die Automatisierungsanforderung sein.

Administration. Die Tool-gestützte Administration von Organisationsnetzen wird in der Regel vernachlässigt. Dennoch stellt sie einen der kritischen Effizienzmerkmale dar. Die Administration bewegt sich im Spektrum zwischen rein manueller und vollständig automatisierter Administration. Tool-Unterstützung wird in diesem Zusammenhang stets integral aufgefasst. Eine Korrelation mit dem Merkmal „Gründungsprozess“ ist offensichtlich.

Betrachtungsebene. Abbildung 2.5 stellt die unterschiedlichen Betrachtungsebenen dar, die sich in Abbildung 1.3 in der Dimension „Managementaspekte“ widerspiegeln. Beide Darstellungen machen deutlich, dass das Management von Organisationsnetzwerken anderen Anforderungen genügen muss, je nachdem ob eine Mikro-, Meso- oder Makro-Sicht zu Grunde gelegt wird [Langer, 2001; Nerb, 2001]. Trotz offensichtlicher Ähnlichkeit mit dem Merkmal „Struktur“ sind beide Merkmale sehr wohl unabhängig: Eine VO von VOs (also die „makro“-Ausprägung der „Betrachtungsebene“) kann durchaus intraorganisational koordiniert werden und die Auswirkungen interorganisationaler Koordinationsmaßnahmen können auf der Ebene von „mikro“-Kooperationen beobachtet werden.

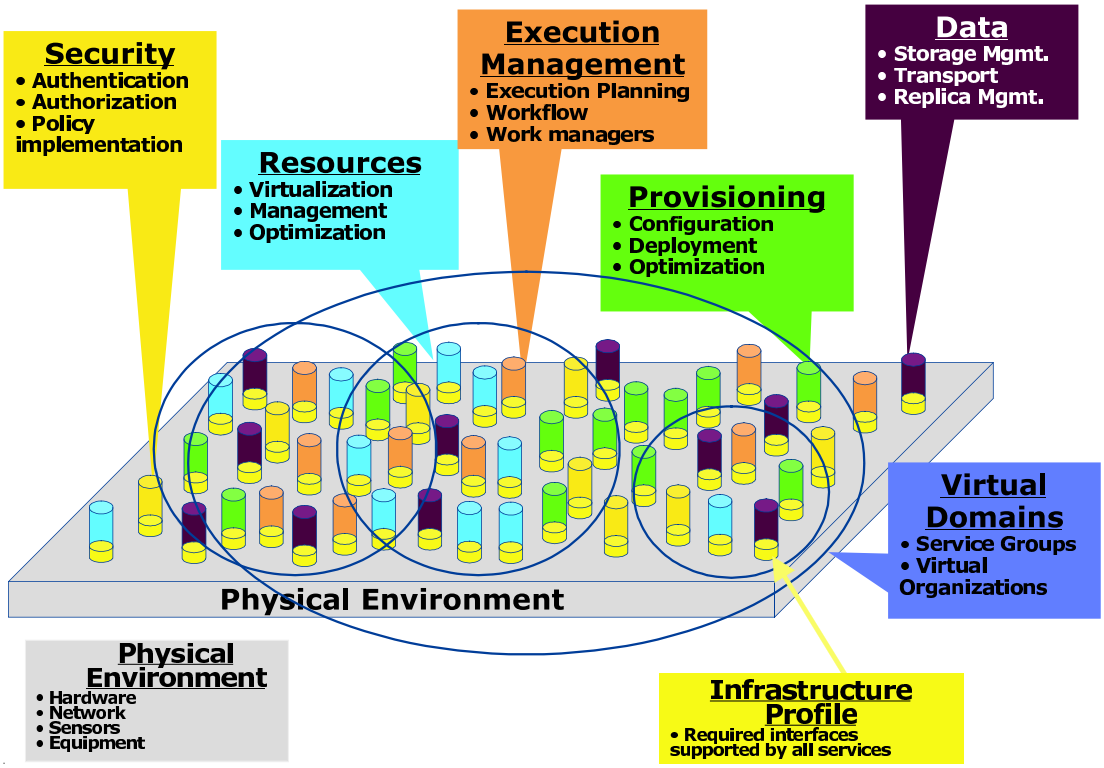


Abbildung 2.11: OGSA-Rahmenwerk nach [Foster u. a., 2006]

Kooperationsstruktur. Die Konzertierung von Organisationsnetzwerken zur Zielerreichung erfordert Kooperationsstrukturen, die diese unterstützen. Diese können als sequentielle Lieferketten (*supply chains*) ausgeprägt sein kann, als sternförmige Master/Slave-Muster (*hub-and-spoke*), als weitgehend strukturlose Peer-to-Peer (P2P)-Systeme oder als Mischformen dieser Grundformen auftreten.

Es sei an dieser Stelle angemerkt, dass in der Praxis die hier vorgestellte glatte Einteilung der Merkmalsausprägungen so nicht immer zu finden ist. Die Klassifikation kann daher nur idealtypischen Charakter haben. Der Übergang von einer Ausprägung zu einer anderen ist keine Seltenheit und schleichende Veränderungen sind durchaus erkennbar, wie [Otto u. a., 2000] an der Entwicklung elektronischer Marktplätze beispielhaft veranschaulicht.

2.1.3 Virtuelle Organisationen in Grids

Nach der allgemeinen Charakterisierung Virtueller Organisationen, werden VOs nun Grid-spezifisch betrachtet. Die Bedeutung des VO-Konzeptes für Grids ist fundamental, wie die Darstellung des Grid-Problems (siehe Seite 1) verdeutlicht. Entsprechend prominent ist der VO-Begriff in der Spezifikation der Open Grid Services Architecture (OGSA) [Foster u. a., 2006] verankert, wie Abbildung 2.11 zeigt.

Dennoch bleibt der VO-Begriff für Grids bisher vage und eine allgemein akzeptierte Definition hat sich noch nicht durchsetzen können. Die heute in der Literatur zu findenden VO-Definitionen für Grids und verwandte Kontexte (siehe die Beispiele in Tabelle 2.1) sind durchweg deskriptiv und auf eine konkrete Sichtweise zugeschnitten. Sie stehen jedoch nicht unbedingt im Einklang mit den organisationstheoretischen Konzepten der Abschnitte 2.1.1 und 2.1.2. Eine Allgemeingültigkeit für den Problemraum dieser Arbeit können sie daher für sich nicht in Anspruch nehmen.

Autor	**Definition**
[Norman, 2006]	*A VO is definable as a list of identified users that represents a real-world group of people that have a clear membership. [...] At its simplest, a VO contains a list of members and their unique identifiers. At its most complex, a VO may contain different status levels of members and many attributes about the members as well as accounting information regarding members' use of grid resources, services or applications.*
[Treadwell, 2006]	*A virtual organization (VO) comprises a set of individuals and/or institutions having direct access to computers, software, data, and other resources for collaborative problem-solving or other purposes.*
[Dreo Rodošek u. a., 2005]	*A VO is defined as a set of virtual resources and virtual services that can be used by individuals to achieve a common goal.*
[von Laszewski u. Wagstrom, 2004]	*An organization that defines rules that guide membership and use of individuals, resources, and institutions within a community production Grid.*
[Kürümlüoglu u. a., 2005]	*Set of cooperating (legally) independent organizations which, to the outside world, provide a set of services and act as if they were one organization*
[Patel u. a., 2005]	*VOs are composed of a number of autonomous entities (representing different individuals, departments and organisations), each of which has a range of problem-solving capabilities and resources at its disposal*

Tabelle 2.1: Definitionen Virtueller Organisationen

Autor	Definition
[VOX Project Team, 2004]	*A Virtual Organization consists of members that may come from many different home institutions, may have in common only a general interest or goal (e.g., CMS physics analysis), and may communicate and coordinate their work solely through information technology (hence the term virtual). In addition, individual members and/or institutions may join and leave the organization over time; sometimes VOs are called dynamic virtual organizations for this reason.*
[Dimitrakos u. a., 2004]	*A virtual organisation is understood as a temporary or permanent coalition of geographically dispersed individuals, groups, organisational units or entire organisations that pool resources, capabilities and information to achieve common objectives*
[Alfieri u. a., 2003]	*abstract entity grouping users, institutions and resources (if any) in a same administrative domain*
[Foster u. a., 2003]	*Dynamic ensembles of resources and services (and people)*
[Foster u. a., 2001]	*Set of individuals and/or institutions defined by resource sharing rules*

Tabelle 2.1: Definitionen Virtueller Organisationen (Fortsetzung)

Viele Definitionen implizieren – im Gegensatz zur Auffassung dieser Arbeit in Abbildung 2.9 – eine Eigentümerschaft (*ownership*), nach der VOs die ihnen zugeordneten Ressourcen auch tatsächlich „besitzen“, oder zumindest den Zugriff darauf *autonom* regeln können. Diese VO-Sicht impliziert eine a priori vorgegebene Statik der Mitgliedschaft. Der dynamische VO-Charakter wird in [Foster u. a., 2003] eingeführt, während [VOX Project Team, 2004] und [Treadwell, 2006] zusätzlich den Aspekt der gemeinsamen Interessenslage und die dafür erforderlichen Koordinationsmechanismen unterstreichen. Eine weitergehende Diskussion der Definitionsproblematik ist in [Norman, 2006] zu finden. In dieser Arbeit wird, den Ausführungen im Abschnitt 2.1.2 folgend, die Prozessorientierung Virtueller Organisationen hervorgehoben. Deshalb wird hier wie folgt definiert:

Definition 1 (Virtuelle Organisation):
Eine Virtuelle Organisation ist eine zeitlich begrenzte koordinierte Kooperation von Elementen in Form von Individuen, Gruppen von Individuen, Organisationseinheiten oder ganzer Organisationen, die Teile ihrer physischen oder logischen Ressourcen oder Dienste auf diesen, ihre Kenntnisse und Fähigkeiten sowie Teile ihrer Informationsbasis in Form virtueller Ressourcen und Dienste über eine Grid-Infrastruktur derart zur Verfügung stellen, dass die gemeinsam vereinbarten Ziele unter Berücksichtigung lokaler und globaler Policies erreicht werden können.

Damit repräsentieren VOs dynamische, organisationsübergreifende Gruppen von „Mitgliedern", die auf (virtuelle) Ressourcen und Dienste unter bestimmten Randbedingungen zugreifen können. VOs kennen folgerichtig eine definierbare Mitgliedschaft (ist Mitglied, ist nicht Mitglied oder ist Mitglied mit eingeschränkten Rechten). Genauer wird die Mitgliedschaft zu VOs geregelt durch:

Definition 2 (Mitgliedschaft zu Virtuellen Organisationen):
Ein Individuum P ist Mitglied einer Virtuellen Organisation V, wenn P autorisiert ist, die von V verwalteten virtuellen Ressourcen und Dienste zu nutzen.

*Eine **Gruppe** G von Individuen ist Mitglied von V, wenn jede Person P in G Mitglied von V ist.*

*Eine **Organisation** O ist Mitglied von V, wenn mindestens eine Person P in O Mitglied von V ist oder wenn O eine Ressource R zur Verwendung durch Mitglieder in V bereitstellt.*

*Die abstrakte Repräsentation eines VO-Mitgliedes wird **Mitgliedsobjekt** genannt.*

2.1.4 Zwischenfazit: VOs als zentrales Grid-Konzept

Die vorangegangenen Überlegungen haben gezeigt, dass die wissenschaftliche Auseinandersetzung mit Virtuellen Organisationen ein weit verzweigtes Gestaltungsfeld darstellt, welches sich durch uneinheitliche Begrifflichkeiten und Konzepte auszeichnet, die in diversen Beschreibungs- und Erklärungsansätzen aufgegriffen werden. Eine gemeinhin anerkannte Strukturierung des Forschungsfeldes liegt derzeit nur rudimentär vor. Es bleibt hervorzuheben, dass der Forschungsschwerpunkt dieser Arbeit – nämlich Virtuelle Organisationen in Grids – nur zögerlich einer systematischen Behandlung zugeführt wird und nach wie vor keiner ganzheitlichen Betrachtung unterliegt. Dies liegt nicht zuletzt am Pragmatismus derzeitig relevanter Anwendungsfälle (siehe auch Abschnitt 3.1.1 und [Milke u. a., 2006a, b]) und der damit verbundenen Intention, der Komplexität des Betrachtungsfeldes auszuweichen. Aus diesem Grunde wird hier vorgeschlagen, generische Merkmale gemäß Abbildung 2.12 als Differenzierungskriterien für VOs in Grids zu betrachten.

Die damit vorgeschlagene Fokussierung impliziert zugleich eine Eingrenzung des Schwerpunktes dieser Arbeit auf

- explizit geführte (Organisations-) Netze, die typischerweise über eine ausgeprägte Koordinationsfunktion verfügen

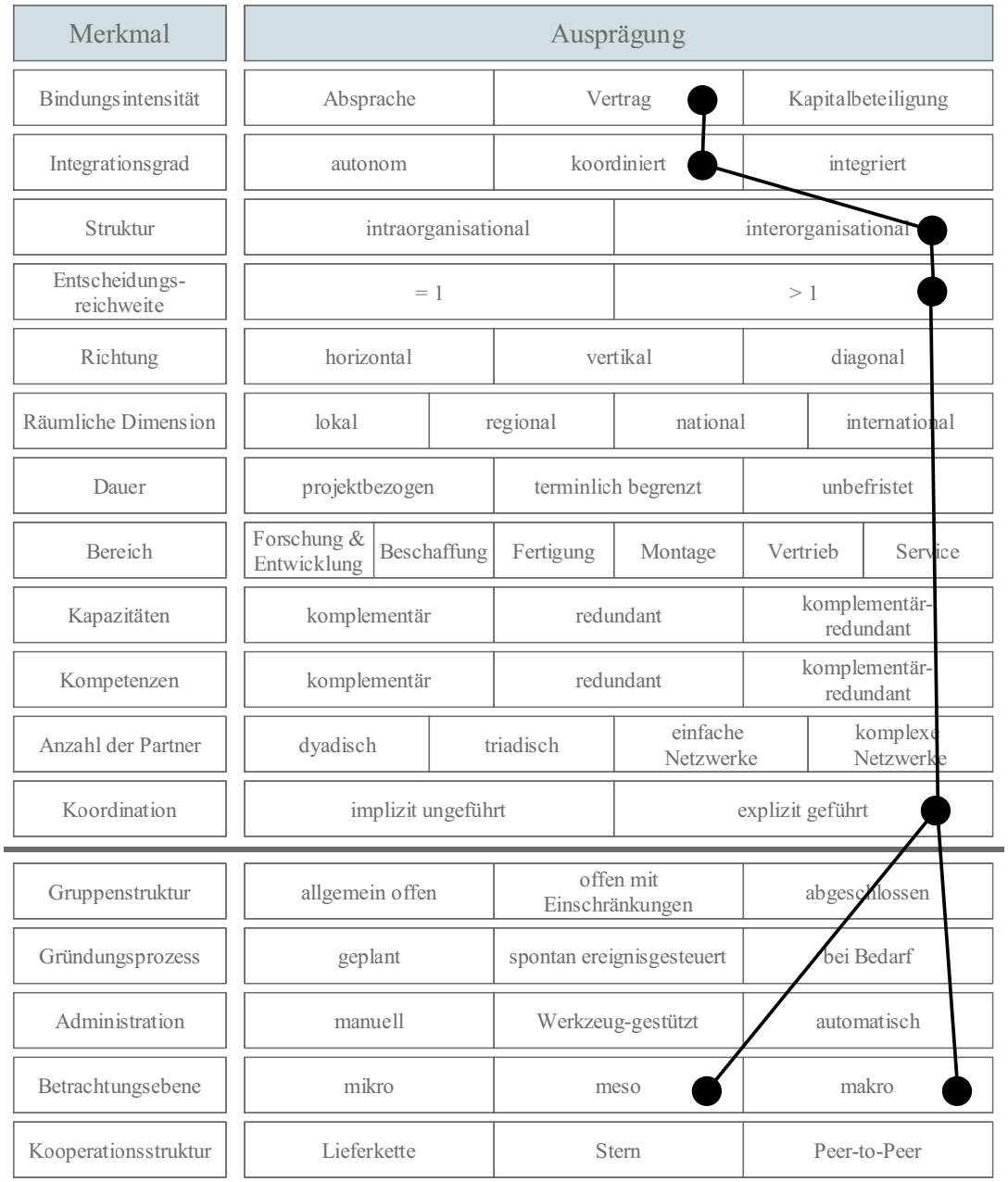


Abbildung 2.12: Standardausprägung Virtueller Organisationen in dieser Arbeit

- polyzentrisch angelegte interorganisationale Strukturen

Damit werden selbstorganisierende Kooperationsnetze ebenso wenig betrachtet wie intraorganisationale Projektmanagement-Strukturen.

2.2 Der Aspekt „Dienstorientierung“

Der Begriff des Dienstes ist für diese Arbeit aus zwei Gründen von Bedeutung (siehe auch Abbildung 2.1): Erstens werden für das angestrebte Management dynamischer Virtueller Organisationen in Grids Managementdienste im Rahmen einer dienstorientierten Architektur bereitzustellen sein, zweitens ist es das Ziel, mit Hilfe solcher Managementdienste auf VOs – und damit indirekt auf die von VOs bereitgestellten Dienste – einzuwirken. Je nach Anwendungsgebiet wird der Dienstbegriff allerdings mit unterschiedlichen Bedeutungen belegt, wie [Dreo Rodošek, 2002] und [Dreo Rodošek u. Hegering, 2004] zeigen, so dass in der Literatur keine allgemeingültig akzeptierte Definition des Dienstbegriffes zu finden ist. Vielmehr prägen die durch die jeweilige Sichtweise implizierten Anforderungen die Semantik des Dienstbegriffes. Aus diesem Grund wird in dieser Arbeit auch nicht detailliert auf die Spezifika einzelner Dienstdefinitionen eingegangen oder gar ein neues formales, für alle Szenarien anwendbares, allgemeines Dienstmodell entwickelt. Dieser Thematik widmet sich [Dreo Rodošek, 2002] ausführlich. Stattdessen wird hier ein Dienstmodell verwendet, das aus einem allgemeinen Referenzmodell instanziiert wird.

Aus den diversen Dienstbegriffen der Literatur lassen sich zwei grundsätzliche Aspekte eines Dienstes isolieren [Dreo Rodošek, 2002; Lewis, 1999; Nerb, 2001]: Der *statische* Aspekt, der sich mit den charakteristischen, weitgehend unveränderlichen Eigenschaften eines Dienstes beschäftigt, und der *dynamische*, in dem Dienste als Funktion der Zeit oder anderen Kausalgefügen betrachtet werden. Beide Aspekte werden im Folgenden kurz eingeführt und anschließend auf den speziellen Kontext Service-orientierter Architekturen und deren Realisierungen im Grid-Umfeld abgebildet.

2.2.1 Allgemeine Charakterisierung des Dienstbegriffes

Im Laufe der Zeit hat es diverse Anstrengungen gegeben, einen Dienstbegriff funktionsorientiert zu definieren („Ein Dienst ist eine Funktionalität, die von einem Objekt an einer Schnittstelle angeboten wird.“). Für eine rein systemtechnische Betrachtung ist diese Sicht eines *computational interface* auch durchaus ausreichend, da sie mit Begriffen wie Dienstprimitive, Parameterübergabe, Access Control operiert [Dreo Rodošek u. Hegering, 2004]. Sie legt ebenso ein rudimentäres Organisationsmodell zu Grunde, das Rollen wie „Provider/Dienstanbieter“ und „Consumer/Dienstnehmer“ postuliert. Sie berücksichtigt allerdings keine managementspezifischen Aspekte. Damit diese jedoch angemessen berücksichtigt werden können – und das müssen sie, wenn Dienste in organisationsweite und organisationsübergreifende, geschäftskritische Prozesse eingebunden werden sollen, müssen Begrifflichkeiten wie „Dienst“ und „Dienstschnittstelle“ hinreichend allgemein gefasst und sauber abgegrenzt werden.

Dieser Problematik widmet sich [Garschhammer u. a., 2001a]. Das dabei entwickelte Dienstmodell des MNM-Teams (das MNM-Dienstmodell) und seine Erweiterung in [Garschhammer u. a., 2001b] verfolgt zwei Ziele: Einerseits sollen die Begriffe, die im Umfeld ei-

nes dienstorientierten Managements verwendet werden, möglichst klar aus dem Modell ableitbar sein, andererseits sollen konkrete Szenarios sauber modellierbar sein. Die Festlegung von Entitäten, Rollen, Interaktionen und Dienstlebenszyklen sowie deren Beziehungen dient genau diesen Zwecken.

Das MNM-Dienstmodell kennt mehrere Sichten auf einen Dienst. Im Basismodell (siehe Abbildung 2.13) werden die drei fundamentalen Rollen „Nutzer“ (*User*), „Kunde“ (*Customer*) und „Dienstleister“ (*Provider*) definiert und mit dem Dienstbegriff assoziiert. Die Rollen werden dabei entweder der Dienstnehmerseite oder der Dienstleisterseite zugeordnet. Eine Domäne wird verwendet, um Zuständigkeitsbereiche festzulegen und zu beschreiben.

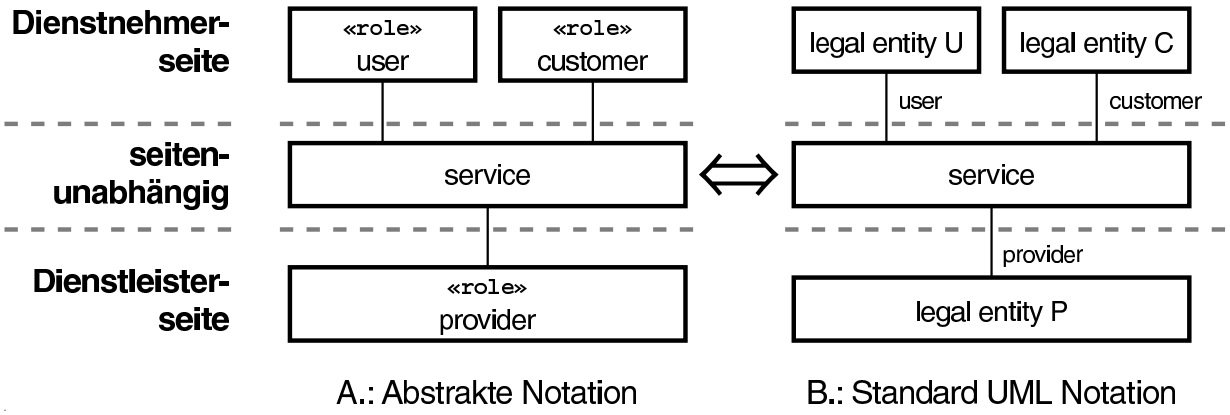


Abbildung 2.13: Das MNM-Basismodell nach [Garschhammer u. a., 2001b]

Auf der Dienstnehmerseite werden explizit die Rollen „Nutzer“ und „Kunde“ unterschieden. Die Kundenrolle besitzt zwei Ausprägungen. Zum einen ist es der Kunde, der eine vertragliche Beziehung über den Dienst mit dem Dienstleister eingeht[3]. Zum anderen ist der Kunde aber auch in Managementinteraktionen involviert, die er über das Customer Service Management (CSM) [Langer, 2001] zur Verfügung gestellt bekommt. Im Gegensatz zum Kunden greift der Dienstnutzer auf die Nutzfunktionalität eines Dienstes über den Dienstzugangspunkt zu. Selbstverständlich können die Rollen „Nutzer“ und „Kunde“ auch von derselben Entität im Rahmen von Dienstketten (vertikaler und horizontaler Art [Nerb, 2001]) eingenommen werden.

Auf der Dienstleisterseite übernimmt die Provider-Rolle den Gegenpart zum Kunden im Rahmen vertraglicher Verhandlungen und Vereinbarungen. Der Diensterbringer ist für die Bereitstellung des Dienstes verantwortlich. Dazu betreibt er eine Dienstimplementierung und ein Dienstmanagement. Im MNM-Dienstmodell werden Dienstfunktionalität und Dienstrealisierung unabhängig voneinander betrachtet. Entsprechend ist der Dienst selbst auch keiner der beiden Seiten zugeordnet.

[3]Kunde, Nutzer und Dienstleister müssen nicht notwendigerweise natürliche oder juristische Personen sein.

Neben dem Basismodell kennt das MNM-Dienstmodell deshalb noch die Dienst- und die Implementierungssicht. Um ein kongruentes Dienstverständnis zwischen Dienstleister und -nehmer zu erzielen, wird der Dienst über die Dienstsicht (siehe Abbildung 2.14) spezifiziert.

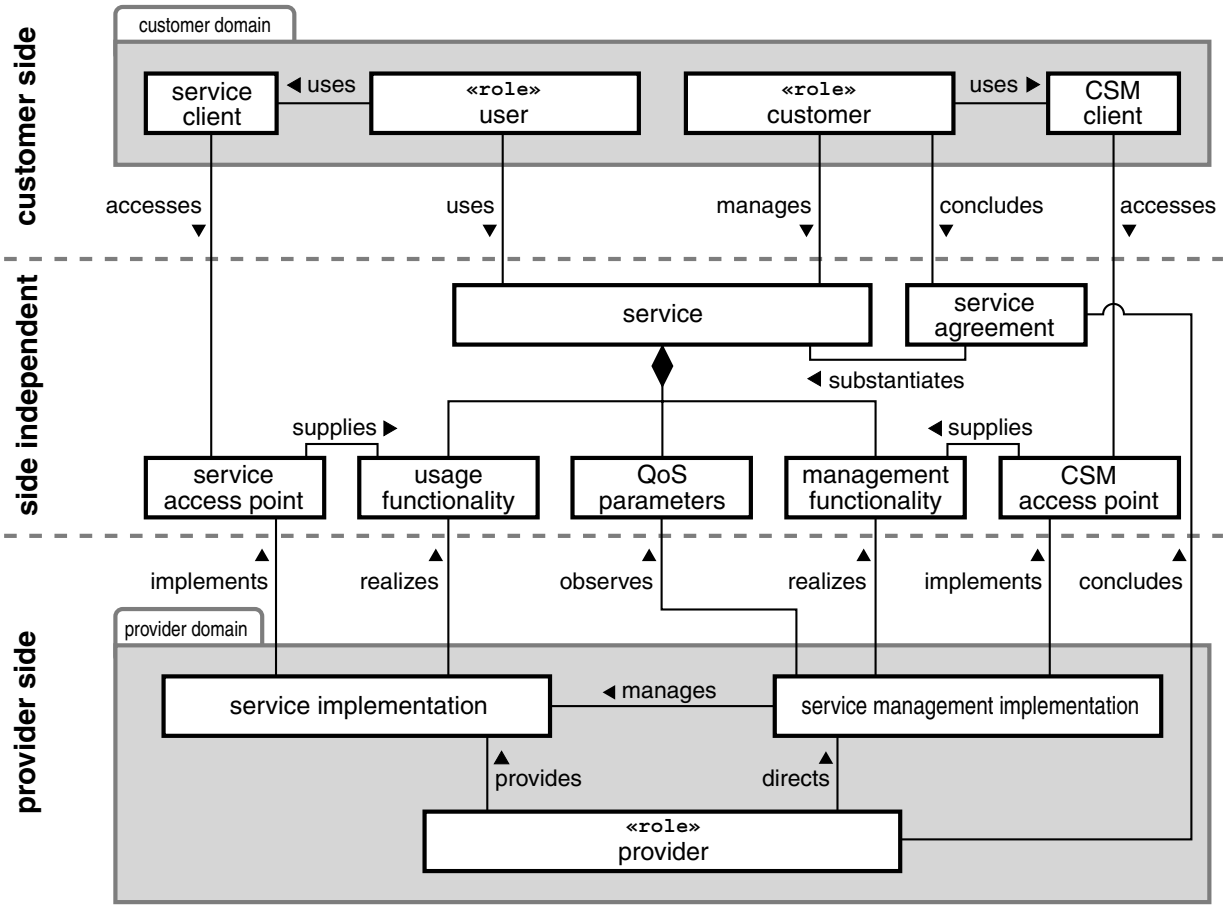


Abbildung 2.14: Die Dienstsicht des MNM-Dienstmodells nach [Garschhammer u. a., 2001b]

Der Dienstbegriff des MNM-Modelles (siehe Abbildungen 2.13 und 2.14 und [Dreo Rodošek, 2002; Garschhammer u. a., 2001b]) kennzeichnet ganz allgemein eine Funktionalität, die einem Dienstnehmer an einer Schnittstelle mit gewissen Dienstgütemerkmalen von einem Dienstleister zur Verfügung gestellt wird. Die angebotene Funktionalität umfasst dabei sowohl Nutzungs- als auch Managementfunktionalitäten. Als Funktionalität wird dabei eine Menge von Interaktionen verstanden, die zwischen der Dienstleister– und Dienstnehmerseite stattfinden. Die Interaktionen können auf Anwendungstransaktionen, Protokolltransaktionen und Workflows abgebildet werden. Während die Nutzungsfunktionalität alle Interaktionen zwischen der Dienstleister– und Dienstnehmerseite umfasst, die dem eigentlichen Zweck des Dienstes dienen, werden die übrigen zwischen Dienstnehmer– und Dienstleisterseite stattfindenden Interaktionen, die für den Betrieb eines Dienstes notwendig sind, aber nicht dem eigentlichen funktionalen Zweck des Dienstes zugeordnet werden können, zur Managementfunktionalität zusammengefasst. Typische Beispiele sind User

Support, der Betrieb einer Hotline, Rechnungstellung und –zahlung, aber auch das Starten/Stoppen von Ressourcen oder das Anstoßen von Dienstanpassungen.

Die Güte, mit der die Dienstfunktionalität erbracht werden soll, wird mit Hilfe von *Dienstgüteparametern* beschrieben, die für beide Arten der Funktionalität getrennt definiert werden können. Über entsprechende Vereinbarungen werden in der Regel mit Hilfe der Dienstgüteparameter Schranken für die noch tolerierbare Dienstgüte durch Festlegung konkreter Werte spezifiziert. Erfolgt keine Wertefestlegung bezüglich der Dienstgüteparameter, so wird der Dienst nach dem Best–Effort–Prinzip betrieben.

Diejenigen Entitäten, die notwendig sind, um dienstnehmerseitig die Funktionalität des vereinbarten Dienstes nutzen zu können, werden in der Dienstschnittstelle zusammengefasst. Analog zur Aufteilung der Dienstfunktionalität wird die Dienstschnittstelle in den Dienstzugangspunkt und den Management-Zugangspunkt (CSM-Schnittstelle in [Nerb, 2001]) aufgeteilt. Während am Dienstzugangspunkt dem Dienstnutzer nur die vereinbarte Nutzungsfunktionalität zur Verfügung steht, erlaubt der Management-Zugangspunkt dem Dienstkunden den vereinbarten Zugriff auf Managementfunktionalitäten.

Die Dienstvereinbarung enthält eine Beschreibung der Dienstfunktionalität sowie die Festlegung der dazugehörigen Dienstgüteparameter. Zusätzlich werden auch die Dienstschnittstellen spezifiziert, an denen von Dienstnehmerseite aus die beschriebene Funktionalität in Anspruch genommen werden kann. Die Dienstvereinbarung wird zwischen Diensterbringer und Kunde im Rahmen eines Vertrages geschlossen. In [Schmidt, 2001] wird die Struktur, der Inhalt sowie die Vorgehensweise zur Spezifikation von Dienstvereinbarungen detailliert behandelt.

Die Dienstimplementierung, die vom Diensterbringer betrieben wird, realisiert primär die Nutzungsfunktionalität des vereinbarten Dienstes. Zusätzlich wird auch die Nutzungsschnittstelle implementiert, um den Zugriff auf die Funktionalität zu ermöglichen. Eine Dienstimplementierung ist jedoch nicht nur rein technisch zu sehen. Hierbei handelt es sich oft um die Kombination des gesamtenorganisatorischen Wissens, des Personals sowie der Hard– und Software, die für die Dienstrealisierung erforderlich ist. Die Dienstimplementierung wird durch die Implementierungssicht des MNM-Dienstmodelles (dargestellt in Abbildung 2.15) unterstützt.

Im Zusammenhang mit der Dienstimplementierung umfasst die Dienstmanagementimplementierung alle erforderlichen Maßnahmen und Ressourcen, die sicherstellen, dass der Dienst geplant, installiert und betrieben werden kann und zu keinem Zeitpunkt gegen Dienstvereinbarungen verstoßen wird.

Dienste können horizontal und vertikal (hierarchisch) in Dienstketten angeordnet werden [Nerb, 2001]. In [Garschhammer u. a., 2001a] wird deshalb die Beziehung zwischen Diensterbringer und Sub–Diensterbringern untersucht, die an der Bereitstellung eines Dienstes beteiligt sind. In diesem Fall werden Teile der Dienstimplementierung respektive der Dienstmanagementimplementierung des Diensterbringers an weitere (Sub–)Diensterbringer delegiert. Da die Beziehungen zwischen Diensterbringer und Sub–Diensterbringern rekursiv auf Customer–Provider–Beziehungen des MNM-Dienstmodells zurückgeführt wer-

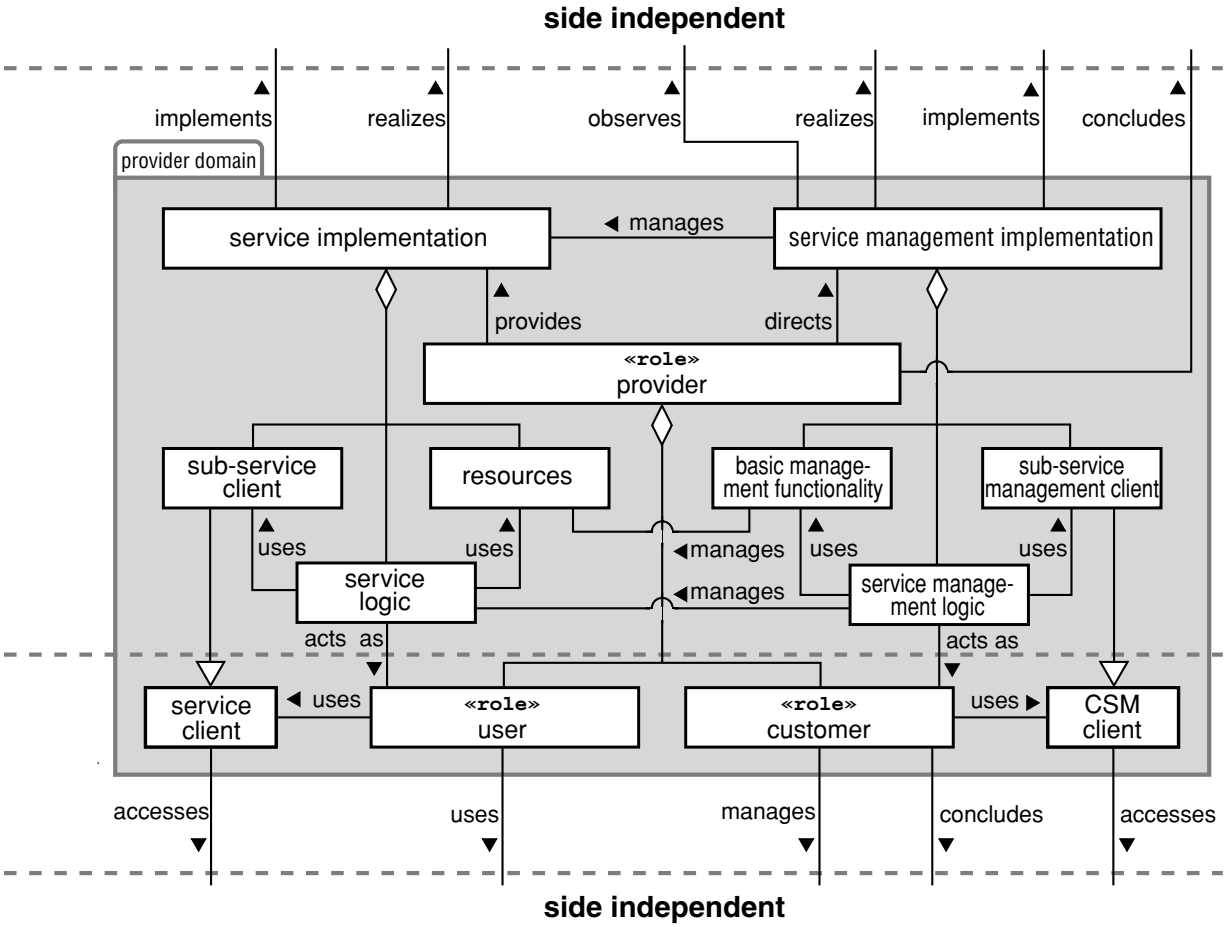


Abbildung 2.15: Die Implementierungssicht des MNM-Dienstmodells nach [Garschhammer u. a., 2001b]

den können, ist eine Einführung neuer Rollen, um derartige Beziehungsgeflechte auszudrücken, nicht notwendig. Auf eine tiefergehende Betrachtung dieses Aspektes wird in diesem Abschnitt verzichtet und auf [Garschhammer u. a., 2001a] sowie [Garschhammer u. a., 2001b] verwiesen.

Das MNM-Dienstmodell ermöglicht die Beschreibung von beliebigen Diensten. Die Frage, wie das Dienstmodell auf einen gegebenen Fall anwendet werden kann, wird in [Garschhammer u. a., 2002] im Rahmen einer allgemeinen Anwendungsmethodik behandelt. Eine Anwendung des Modells auf Web Hosting Services ist in [Hanemann u. a., 2004] zu finden, [List, 2004] verwendet es für die Modellierung von Vertrauensdiensten in elektronischen Marktplätzen.

Neben dem statischen Aspekt besitzen Dienste eine Dynamik, die im Lebenszyklus von Diensten beobachtet werden kann. Dienstlebenszyklen umfassen die grundsätzlichen Phasen eines Dienstes und deren Übergänge vom ersten Entwurf während der Planung bis zur vollständigen Auflösung (siehe Abbildung 2.16). Die dabei auftretenden Lebenszyklus-

phasen stellen in der Regel eine Verfeinerung der in [Hegering u. a., 1999] vorgeschlagenen Phasen dar und finden sich in ähnlicher Weise in [Dreo Rodošek, 2002], dem EGA Referenzmodell [Enterprise Grid Alliance, 2006], aber auch in der Dienstarchitektur des TINA Consortiums [Abarca u. a., 1997] und im Service Lifecycle Management der Configuration Description, Deployment, and Lifecycle Management (CDDLM)-Working Group des OGF [Bell u. a., 2005; Dantas u. a., 2006], wieder.

Basierend auf [Hegering u. a., 1999] unterscheidet [Dreo Rodošek, 2002] sechs Lebenszyklusphasen (siehe auch Abbildung 2.16). In der Planungsphase unterbreitet der Dienstanbieter einem Dienstinteressenten ein Angebot über die Diensterbringung und das von ihm bereitstellbare Leistungsspektrum. In der Regel prüft der Interessent mehrere Angebote von verschiedenen Dienstanbietern. Aus der Managementsicht erfordert die Planungsphase eine Reihe von Analysen [Hegering u. a., 1999]. Beispiele sind Anwendungsanalysen (Wie ist der Dienst zu erbringen? Was ist seine Funktionalität?), Bedarfsschwerpunktanalysen (räumliche Verteilung der potenziellen Nutzer und Ressourcen), Bedarfsgrößenanalysen (Feststellung der zeit- und mengenmäßigen Verteilung von Transaktionen und Daten) oder Komponentenanalysen (Feststellung von Typ und Anzahl der zur Diensterbringung benötigten Ressourcen). Damit werden in der Planungsphase die vorher skizzierten statischen Aspekte des Dienstes analysiert und festgelegt. Dies betrifft vornehmlich die Identifizierung von Dienstzugangspunkten, Management-Zugangspunkten, Dienstfunktionalitäten, aber auch die Spezifikation von Dienstgüteparametern sowie der Dienstabhängigkeiten und der grundsätzlichen systemtechnischen Dienstrealisierung.

In der Verhandlungsphase wird ein auf der vorherigen Planung ausgelegter Vertrag zwischen dem Interessenten und dem Anbieter ausgehandelt. In der Regel endet diese Phase mit dem Abschluss eines Vertrages, wodurch der Interessent zum Kunden wird. Ein solcher Vertrag enthält typischerweise eine detaillierte Beschreibung des Dienstes und der möglicherweise anzuwendenden Tarife respektive Vertragsstrafen.

In der Bereitstellungsphase wird die vorher vertraglich vereinbarte Dienstfunktionalität durch den Anbieter realisiert. In dieser Phase werden die Ressourcen (z.B. Geräte, Endsysteme, IP-Adressen, Applikationen) in der Art und Weise bereitgestellt, installiert, initial konfiguriert und getestet, dass der Dienst nach den vereinbarten Grundsätzen betrieben werden kann. Diese Phase endet mit der Akzeptanz des konkreten Dienstes durch den Kunden.

Nachdem der Kunde den Dienst abgenommen hat, wird der Dienst in der Betriebsphase in Betrieb genommen. Neben dem reinen Dienstbetrieb werden vom Diensterbringer auch Managementaufgaben wie der Betrieb einer Hotline oder eines User Help Desks übernommen. Nach [Hegering u. a., 1999] wird die Betriebsphase aus Managementsicht in den Routinebetrieb, den Störungsbearbeitungsbetrieb und den Änderungsbetrieb unterschieden. Alle drei Betriebsmodelle reflektieren unterschiedliche betriebliche Abläufe, die im Zusammenhang mit der Bereitstellung der Dienstleistung durch den Dienstanbieter durchgeführt werden müssen. Im Routinebetrieb wird der laufende Betrieb des Dienstes überwacht, Datensicherungs-, Pflege- und Instandhaltungsarbeiten durchgeführt und Be-

triebsstatistiken mitgeführt. Der Störungsbetrieb muss dafür Sorge tragen, dass auftretende Störungen im Sinne [Avižienis u. a., 2001] frühzeitig erkannt und behandelt werden.

Der Änderungsbetrieb beinhaltet die geplante und abgestimmte, aus dem laufenden Betrieb resultierende Durchführung von Änderungen am Dienst, wie die Einführung neuer (Anwendungs-) Software oder die Einführung neuer Hardwaresysteme.

In der Anpassungsphase werden alle Aktivitäten zusammengefasst, die mit Änderungen der Dienstfunktionalität, der Dienstimplementierung und des Dienstmanagements einhergehen. Die Anpassungsphase ist nicht zu verwechseln mit dem Änderungsbetrieb der Betriebsphase. In der Anpassungsphase werden grundsätzliche planerische Eingriffe an einem Dienst vorgenommen. Ein typischer Anwendungsfall ist die Einstellung der Dienstleistung durch einen Dienstanbieter (d.h. die Dienstleistung wird nicht mehr durch den Dienstanbieter angeboten). Die Anpassungsphase spielt eng mit der eingangs genannten Planungsphase zusammen und verdeutlicht, dass spätestens an dieser Stelle eine Rückkopplung zwischen den identifizierten Phasen stattfinden muss.

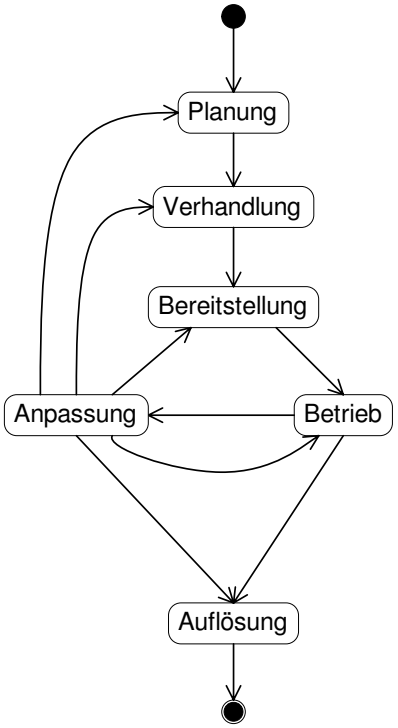


Abbildung 2.16: Typischer Dienstlebenszyklus nach [Dreo Rodošek, 2002]

Der Dienstlebenszyklus endet schließlich mit der Auflösungsphase, in der die durch die Implementierung belegten Ressourcen wieder freigegeben werden und die für das Management des Dienstes erforderlichen Konfigurationen in einen definierten Endzustand versetzt werden.

2.2.2 Service-orientierte Architekturen

Das vorher beschriebene allgemeine Dienstmodell findet in abgewandelter Form Anwendung in Service-orientierten Architekturen (SOA) [Erl, 2005]. SOAs stellen eine Weiterentwicklung der seit den 1990er Jahren populären Component Based Architectures (CBA) [Crnkovic u. Larsson, 2002] dar. Klassische CBA-Beispiele sind Microsoft's Distributed Component Object Model (DCOM) [Stal, 2006], die Common Object Request Broker Architecture (CORBA) [Siegel, 1996], die Java 2 Platform Enterprise Edition (J2EE) [Farley u. a., 2005], aber auch das Internet an sich. Obwohl in der Öffentlichkeit SOA und Web Services-Technologien [Newcomer, 2002] häufig synonym verwendet werden, sind beide Konzepte zu unterscheiden: Web Services bilden nur *eine* Möglichkeit, Dienst-orientierte Architekturen zu implementieren. CORBA, Microsoft's .NET oder die 1999 vom OASIS-Konsortium und der United Nations Economic Commission for Europe (UN/ECE)-Tochterorganisation Centre for Trade Facilitation and Electronic Business (CEFACT)[4] gestartete Electronic Busi-

[4] `http://www.unece.org/cefact/`

ness using eXtensible Markup Language (ebXML)-Initiative [OASIS, 2006] stellen alternative Implementierungsmodelle dar, die allesamt auf dem vom SOA Reference Model Technical Committee des OASIS-Konsortiums definierten SOA-Referenzmodell aufbauen, dieses allerdings den eigenen Anforderungen entsprechend erweitern [OASIS, 2007]. Trotz zum Teil erheblicher Unterschiede basieren alle implementierten SOA-Ansätze auf den in Abbildung 2.17 dargestellten Konzepten.

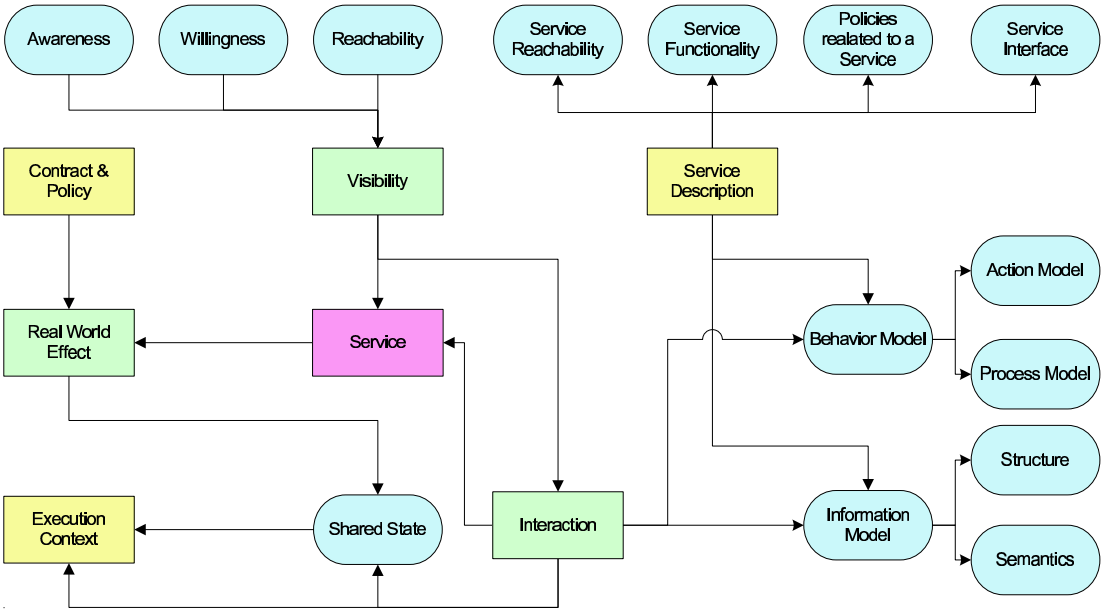


Abbildung 2.17: Basiskonzepte des SOA Referenzmodells nach [OASIS, 2007]

Das SOA-Referenzmodell

Das SOA-Referenzmodell betrachtet *Services* als Kern einer jeden SOA. Sie stellen den Mechanismus dar, der für den Zugang zu *Capabilities* über definierte Schnittstellen unter Berücksichtigung von Policies oder anderen in der Dienstbeschreibung spezifizierten Einschränkungen nötig ist. Ein Service wird durch einen *Service Provider* (SP) einem oder mehreren *Service Consumern* (SC) zur Nutzung zur Verfügung gestellt. Anders als im klassischen Fall sind jedoch weder die Consumer a priori bekannt noch das mögliche Nutzungsmuster *usage pattern*. Auf einen Dienst wird stets über das *Service Interface* zugegriffen. Dienste in SOA sind insofern *opaque* als ihre Implementierungen vor dem Consumer durch Kapselung verschattet werden. Ausnahmen bilden lediglich die Informations- und Verhaltensmodelle, die über die Dienstschnittstelle offen gelegt werden, und die Information, die Consumer benötigen, um die Adäquatheit eines Dienstes einschätzen zu können. Das Ergebnis der Ausführung eines Dienstes ist die Realisierung eines oder mehrerer *real world effects*, die sich in der Übergabe einer verlangten Information oder einem beobachtbaren Zustandswechsel oder einer Kombination beider Effekte ausdrücken. Wie im allgemeinen

Fall auch, unterscheidet das SOA-Servicekonzept strikt zwischen der Funktionalität eines Dienstes – repräsentiert durch *Capabilities* – und dem Zugangspunkt, an dem die *Capabilities* bereitgestellt werden.

Die Dynamik von Diensten wird in *Interaktionen* reflektiert. Dabei kommen folgende Konzepte zum Tragen:

Visibilität zwischen SP und SC. SP und SC sind gegenseitig *visibel* wenn sie miteinander interagieren. Dazu muss der Initiator einer Service-Interaktion die anderen Teilnehmer an der Interaktion kennen (*awareness*), die Parteien müssen interagieren wollen (*willingness*) und die Parteien müssen interagieren können (*reachability*).

Interaktionen zwischen SP und SC. Service-Interaktionen sind primär Aktionen gegen Dienste. In den meisten Fällen bedeutet dies den expliziten Austausch von Nachrichten, andere Modelle sind aber auch denkbar (wie beispielsweise Zustandswechsel gemeinsam genutzter Ressourcen). Die Grundlage für Service-Interaktionen bildet die Dienstbeschreibung, die auf ein Informations- und ein Verhaltensmodell referenziert. Im *Informationsmodell* wird dargelegt welche Information mit dem Dienst ausgetauscht werden kann. Das *Verhaltensmodell* beschreibt die möglichen Aktionen und deren zeitliche Abhängigkeiten.

Beobachtbare Effekte. Das Ziel von SCs ist es, mittels der von SPs zur Verfügung gestellten Dienste, eine Anwendung zu realisieren („Trying to get the service to do something“ [OASIS, 2007]). Dieser beobachtbare Effekt wird im SOA-Referenzmodell *real world effect* genannt.

Das SOA-Referenzmodell verwendet daneben noch ein *Contract & Policy*-Konzept und einen *Execution Context*. *Contracts* und *Policies* spielen insbesondere in organisationsübergreifenden Interaktionen eine nicht unerhebliche Rolle, da sie die Bedingungen und Voraussetzungen für die Nutzung, den Einsatz und die Beschreibung eines Dienstes regeln. Der Ausführungskontext einer Service-Interaktion ermöglicht die Unterscheidung zwischen verschiedenen Diensten und verschiedenen Instanzen eines Dienstes.

Mit diesen Konzepten stellt das SOA-Referenzmodell eine Spezialisierung des allgemeinen Dienstmodelles des Abschnitts 2.2 dar. Für eine weiterführende Diskussion des SOA-Referenzmodelles wird auf [OASIS, 2007], [Krauter u. a., 2002], [Keen u. a., 2004] oder [McGovern u. a., 2003] verwiesen.

SOA und Web Services

SOAs können zwar prinzipiell über jeder dienstzentrischen Technologie implementiert werden, die Umsetzung von SOAs mittels Web Services [Zimmermann u. a., 2005] spielt für den Kontext dieser Arbeit allerdings eine wichtige Rolle. Deshalb soll an dieser Stelle kurz auf Web Services-Technologien eingegangen werden. Während der vorangehende Abschnitt 2.2.2 eine Instanziierung der Dienstsicht des allgemeinen Dienstmodells (Abbildung 2.14)

darstellte, instanziieren die folgenden Ausführungen die Implementierungssicht dieses Modells (Abbildung 2.15).

Die Realisierung einer Service-orientierten Architektur im Rahmen einer Web Services-spezifischen Plattform setzt – gemäß SOA Referenzmodell – im wesentlichen auf der Beschreibung der Dienste und Abläufe durch Web Services Description Language (WSDL)-Dokumente und Business Process Execution Language (BPEL)-Dokumente auf. Danach implementiert ein *Web Service* eine Schnittstelle, die eine Menge von Operationen exponiert, auf die über XML-Nachrichten zugegriffen werden kann [Zimmermann u. a., 2005].

Folgende Rollen können im Rahmen des Web Services-Ansatzes identifiziert werden (siehe auch [Booth u. a., 2004]):

- Service Requestor/Consumer, der auf Web Services eines Service Providers zugreift.
- Service Provider, der Web Services über ein Netzwerk einem Service Requestor zur Verfügung stellt.
- Discovery Agent, der einen Service Requestor mit einem Service Provider verbindet.

In Abbildung 2.18 sind die Interaktionen zwischen Service Providern und Service Requestor dargestellt. Der Service Provider implementiert Dienste in Form von Web Services. Mit Hilfe eines WSDL-Dokumentes wird für den jeweiligen Web Service dessen Schnittstelle spezifiziert, ohne allerdings festzulegen, *wie* der Web Service „hinter“ der Schnittstelle den Dienst realisiert. Innerhalb eines WSDL-Dokumentes werden die folgende Aspekte beschrieben:

- Wo ist der Service zu finden?
- Wie kann der Service angesprochen werden?
- Welche Funktionen können aufgerufen werden?
- Welche Datenformate werden unterstützt?

Grundsätzlich reicht für die Interaktionsfähigkeit zwischen Service Provider und Service Requestor diese Schnittstellenbeschreibung aus. Allerdings ist für eine effektive Interaktion auch die Semantik der aufgerufenen Dienste von Bedeutung, die in der *Binding and Implementation*-Beschreibung festgelegt wird.

Die WSDL-Beschreibung sowie Informationen über den Service Provider und die Web Services werden in der Regel in einem Service Register gespeichert. Häufig wird zur Implementierung des Discovery Agents das *UDDI (Universal Description, Discovery and Integration)*-Repository eingesetzt, das aber – unter anderem aus Skalierbarkeitsgründen – in Grid-Umgebungen nicht zum Einsatz kommt. Möchte ein Service Requestor einen Web Service in Anspruch nehmen, so ist dafür die Kenntnis der Schnittstellenbeschreibung erforderlich. Besitzt der Requestor keine Information über die Schnittstelle, den Web Service bzw. den Service Provider, so konsultiert er ein UDDI-Register. Hat er die WSDL-Beschreibung vom UDDI-Register erfragt, so sendet er einen Service Request an den Service Provider. Je

nach Art des angesprochenen Dienstes und dessen Spezifikation sendet der Service Provider eine Service Response zurück, deren Format im WSDL-File spezifiziert ist.

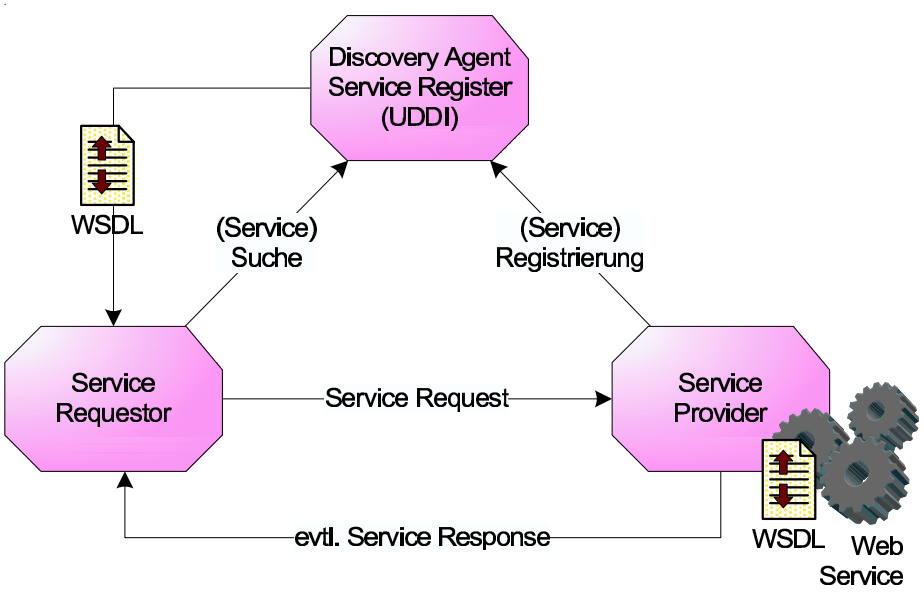


Abbildung 2.18: Web Service Architektur

Abbildung 2.19 zeigt einen häufig verwendeten Protokollstack für Web Services. Neben den bereits betrachteten Begriffen UDDI und WSDL sind in dieser Abbildung noch die Protokolle für den Datentransfer und Transport aufgezeigt. Für den Datentransfer innerhalb der in Abbildung 2.18 beschriebenen Umgebung wird *SOAP*[5] verwendet. SOAP ist ein XML-basiertes Protokoll zum Austausch strukturierter Daten zwischen Applikationen, wobei es selber weder die Semantik der Applikationen noch ein Programmiermodell definiert oder fordert (siehe [Zimmermann u. a., 2005]). Da es der Kommunikation auch keine bestimmten Muster aufzwingt, können SOAP-Messages in diverse Transportprotokolle, wie z.B. HyperText Transfer Protocol (HTTP), eingebunden werden.

Trotz der Beschreibung des Nachrichtenaustauschs durch WSDL sind die modellierbaren Beziehungen nicht ausreichend, um komplexe Kompositionen von Web Services auszudrücken. Zur Konzertierung und Koordination von Web Services in einer Service-orientierten Architektur sind weitergehende Workflowmechanismen notwendig. Hier wird in der Regel mit BPEL eine Sprache verwendet, die die Spezifikation verteilter Service-orientierter Anwendungen auf hohem Abstraktionsniveau unterstützt [Bolie u. a., 2006]. BPEL unterstützt die Koordination der Aktivitäten miteinander interagierender Web Services, indem Mechanismen zur Verfügung gestellt, die die Reaktion auf eingehende Nachrichten erlauben, Ergebnisse auswerten helfen, Datentransformationen und Formatübersetzungen zum Zwecke der Interaktion mit anderen Diensten ermöglichen, Nachrichten als Trigger versenden können und Prozessflüsse koordinieren

[5] Vor der Version 1.2 wurde SOAP als Akronym für *Simple Object Access Protocol* verwendet.

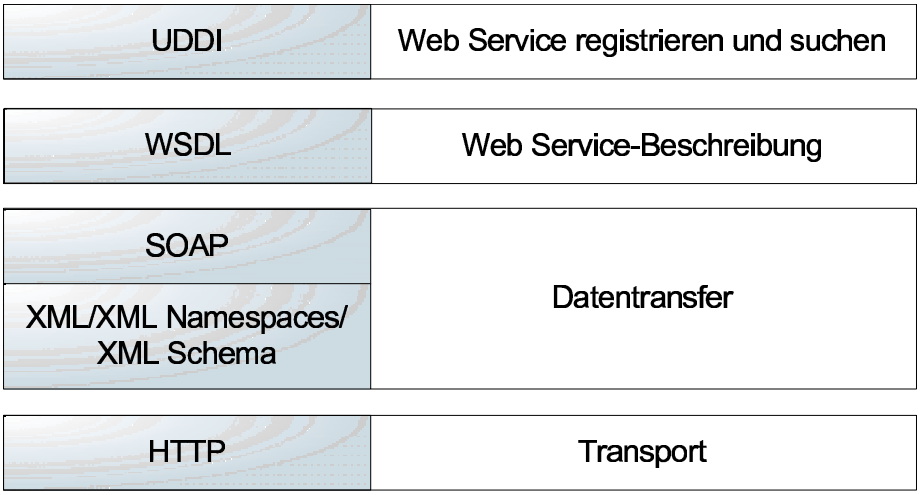


Abbildung 2.19: Protokollstack für Web Services nach [Zimmermann u. a., 2005]

bzw. die Ausführungslogik von Komponenten definieren helfen. Für weitergehende Informationen zu BPEL sei auf die einschlägige Literatur verwiesen (zum Beispiel [Bolie u. a., 2006] oder [Juric, 2006]). Eine konkrete Anwendung von BPEL zur automatischen Verwaltung von Mitgliedschaften in Grid-VOs wird in [Amikem, 2007] demonstriert. Beispiele für kommerzielle SOA-Konzepte liefern die Oracle Fusion SOA Suite[6], die SAP NetWeaver Plattform[7] und IBM's Service Bus-Konzept[8].

2.2.3 Der Dienstbegriff in Grids

Wie das SOA-Referenzmodell zeigt, werden in einer Service-orientierten Architektur die Komponenten des verteilten Systems durch ihre Schnittstellen und ihr Verhalten definiert. Die Implementierung spielt dabei nur eine untergeordnete Rolle. Die Interaktion zwischen den Diensten regeln definierte Protokolle. Als Instanziierung des SOA-Referenzmodelles stellt die Open Grid Services Architecture (OGSA) [Foster u. a., 2006] eine erweiterbare Achitektur für das Erbringen, das Zusammenspiel und die Nutzung von Diensten in Grids dar. OGSA wurde von der gleichnamigen Working Group des Global Grid Forums (GGF) bereits im Jahr 2002 als Standardisierungsvorschlag vorgelegt und befindet sich – aufgrund seiner Komplexität und der weitreichenden Konsequenzen – immer noch in der Diskussionphase, liegt aber inzwischen in der Version 1.5 vor.

Open Grid Services Architecture

OGSA definiert ein Komponentenmodell, das es Anwendungen ermöglicht, auf einfache Weise im Grid angebotene Dienste zu nutzen und miteinander zu koppeln. OGSA geht

[6] `http://www.oracle.com/technologies/soa/soa-suite.html`
[7] `http://www.sap.com/germany/plattform/enterprisesoa/index.epx`
[8] `http://www-306.ibm.com/software/integration/wsesb/`

besonders auf die grundlegenden Probleme in Grid-Umgebungen ein, die bisher zum Teil für jedes Projekt individuell gelöst werden mussten:

- Unterstützung heterogener Ressourcen (Auffinden, Anfrage und Lebenszyklus-Management von verteilten Diensten und Ressourcen),
- Handhabung der Verfahrensregeln lokaler Verwaltungseinheiten (Abbildung der Verfahrensregeln, Transparenz, Sicherheit, Abrechnung),
- Nutzung der Ressourcen (Reservierung, Überwachung, Kontrolle, Nutzungsprofile, Prognosen),
- Job-Ausführung und Qualitätsüberwachung (Co-Scheduling, Job- Überwachung, Workflow-Management, Dienste-Komposition, Dienstgütevereinbarungen),
- Management verteilter Daten (Datenzugriff, Datenintegration, Metadatenverwaltung, Replikation, Caching, Staging, Platzierung),
- Sicherheit (Authentifizierung, Autorisierung, Isolation, Delegation, Umgang mit Firewalls),
- Skalierbarkeit (Vermeidung zentraler Komponenten, dynamisches Hinzufügen und Entfernen von Ressourcen),
- Verfügbarkeit (Umgang mit unzuverlässigen Komponenten, transparente Ausfallbehandlung).

Prinzipiell können Dienste im Grid mit Hilfe der Standard-Web Services-Technologien realisiert werden. Dabei werden Grid-Dienste – wie im klassischen Fall – über WSDL-Dokumente beschrieben. Grid-Dienste können sich über das SOAP-Protokoll untereinander verständigen und den Web Services Security Layer [Kaye, 2003] nutzen. Allerdings gibt es einen wesentlichen Unterschied zwischen den Diensten, die von Web Services einerseits bzw. Grid-Services andererseits bereitgestellt werden: Web Services sind zustandslos, während Grid-Dienste in der Regel ein Zustandskonzept erwarten [Sotomayor u. Childers, 2006]. Diese Diskrepanz wird mit dem WSRF-Ansatz überbrückt.

Web Services Resource Framework

Das Web Services Resource Framework[9] [Czajkowski u. a., 2005; OASIS, 2006j] umfasst eine Sammlung modularer Einzelspezifikationen und beschreibt, wie mit Hilfe von Web Services auf *zustandsbehaftete*, durch Web Servces repräsentierte, Ressourcen zugegriffen werden kann.

Das Kern-Konzept des WSRF-Rahmenwerkes bilden WS-Ressourcen [OASIS, 2006i]. Eine WS-Ressource besteht aus zwei Komponenten, einem *Web Service* und einer *zustandsbehafteten Ressource*, die kein Web Service zu sein braucht (z.B. eine Datenbank oder ein

[9] `http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf`

persistentes VO-Log). Auf die Ressource greift ausschließlich der Web Service zu. Wie die Schnittstelle zur Ressource aussieht, wird in [OASIS, 2006i] spezifiziert und in Abbildung 2.20 dargestellt.

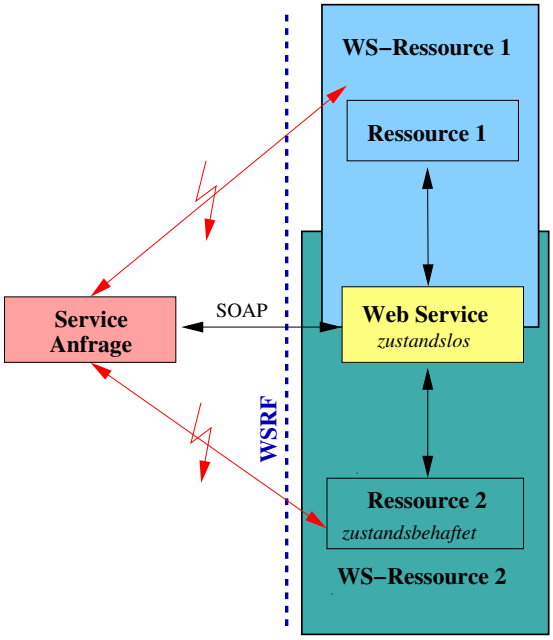


Abbildung 2.20: Aufbau einer WS-Resource nach [OASIS, 2006i] und [Sotomayor u. Childers, 2006]

Damit jede WS-Resource eindeutig angesprochen werden kann, kennt WSRF für jede WS-Resource eine Endpoint Reference (EPR), mit der die Ressource über Mechanismen des WS-Addressing [Bosworth u. a., 2004] referenziert werden kann, und ein `ResourcePropertyDocument`. Ein Beispiel eines `ResourcePropertyDocuments` wird im Listing 2.1 mit der Resource Property eines `GenericDiskDrives` dargestellt.

Listing 2.1: Resource Property eines `GenericDiskDrives` nach [OASIS, 2006i]

```
<wsdl:definitions ... xmlns:tns="http://example.com/diskDrive" ...>
 ...
<wsdl:types>
<xsd:schema targetNamespace="http://example.com/diskDrive" ... >

<!-- Resource property element declarations -->
<xsd:element name="NumberOfBlocks" type="xsd:integer"/>
<xsd:element name="BlockSize" type="xsd:integer" />
<xsd:element name="Manufacturer" type="xsd:string" />
<xsd:element name="StorageCapability" type="xsd:string" />

<!-- Resource properties document declaration -->
<xsd:element name="GenericDiskDriveProperties">
<xsd:complexType>
<xsd:sequence>
<xsd:element ref="tns:NumberOfBlocks"/>
<xsd:element ref="tns:BlockSize" />
```

```
    <xsd:element ref="tns:Manufacturer" />
19  <xsd:any minOccurs="0" maxOccurs="unbounded" />
    <xsd:element ref="tns:StorageCapability"
21  minOccurs="0" maxOccurs="unbounded" />
    </xsd:sequence>
23  </xsd:complexType>
    </xsd:element>
25  ...
    </xsd:schema>
27  </wsdl:types>
    ...
29  <!-- Association of resource properties document to a portType -->
    <wsdl:portType name="GenericDiskDrive"
31  wsrf-rp:ResourceProperties="tns:GenericDiskDriveProperties" >

33  <operation name="start" .../>
    <operation name="stop" .../>
35  ...
    </wsdl:portType>
37  ...
</wsdl:definitions>
```

WSRF basiert neben WS-Addressing auf den folgenden Einzelspezifikationen:

WS-ResourceLifetime. Um die Lebensdauer einer WS-Ressource zu spezifizieren oder zu überwachen, werden Konstrukte dieser WSRF-Teilspezifikation verwendet. WS-ResourceLifetime [OASIS, 2006k] unterscheidet zwischen *zeitgesteuerten* und *sofortigen* Auflösungen von WS-Ressourcen. Der Zeitpunkt der Auflösung darf dabei beliebig weit in der Zukunft liegen. Um die Lebensdauer zu verlängern oder zu verkürzen, wird die WSRF-Methode `SetTerminationTime` verwendet, die Referenzzeit des Servers kann mit der Funktion `CurrentTime` abgerufen werden. Die sofortige Zerstörung einer WS-Ressource wird über die `destroy`-Methode erreicht. Mit der Ausführung jeder Aktion, ob erfolgreich oder fehlerhaft, ist immer eine entsprechende Notifikation verbunden.

WS-ResourceProperties. Die Spezifikation der WS-ResourceProperties [OASIS, 2006l] beschreibt den Zugriff auf *Resource Properties* über Operationen wie `GetResourceProperty`, `SetResourceProperty` und `DeleteResourceProperty`.

WS-ServiceGroup. In der WS-ServiceGroup-Spezifikation [OASIS, 2006m] wird beschrieben, wie Web Services fallspezifisch zu `ServiceGroups` gruppiert werden. Die Gruppierungskriterien werden dabei über die WS-ResourceProperty `MembershipContentRule` festgelegt. Zur Verwaltung und Registrierung steht die `ServiceGroupRegistration`-Schnittstelle zur Verfügung. Ein Web Service kann mehreren `ServiceGroups` angehören. `ServiceGroups` können die Mitgliedschaft einschränken (wenn z.B. in einer Gruppe nur die Ressourcen aufgenommen werden sollen, die von einem bestimmten Web Service bedient werden). Abbildung 2.21 zeigt das `ServiceGroup`-Konzept im Überblick

WS-BaseFaults. WS-BaseFaults [OASIS, 2006a] beschreibt die Struktur der Fehlermeldungen im WSRF-Rahmenwerk. Alle Fehlermeldungen besitzen die im XML-Schema

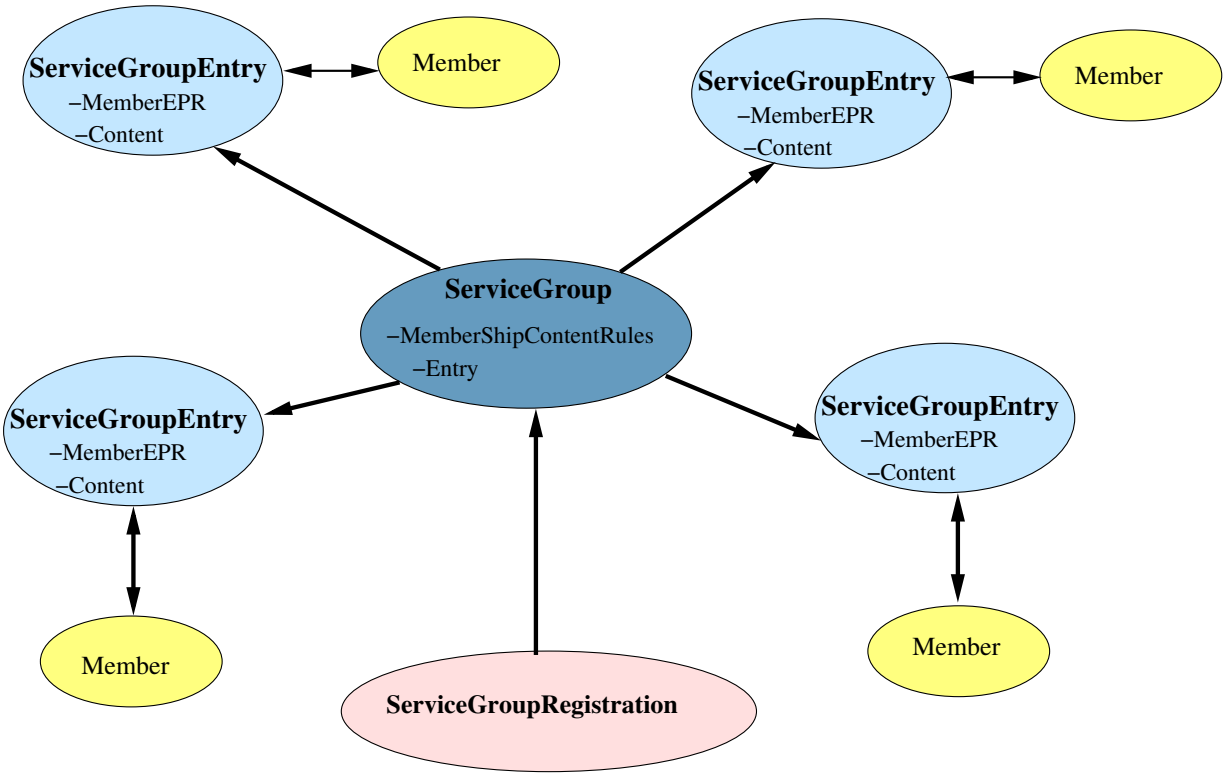


Abbildung 2.21: WSRF Service Groups nach [Lindner u. Schier, 2004]

des Listings 2.2 dargestellte Struktur mit einer Fehler-Beschreibung, einem Zeitstempel, einem Fehlercode und der Quelle des Fehlers.

Listing 2.2**:** XML-Schema des BaseFaults nach [OASIS, 2006a]

```
<BaseFault>
        {any}*
        <Timestamp>xsd:dateTime</Timestamp>
        <OriginatorReference>
        wsa:EndpointReferenceType
        </OriginatorReference> ?
                <ErrorCode dialect="anyURI">xsd:anyType</ErrorCode> ?
        <Description>xsd:string</Description> *
        <FaultCause>{any}</FaultCause> ?
</BaseFault>
```

WS-Notification. WSRF sieht einen Ereignis-gesteuerten Notifikationsmechanismus vor, ähnlich dem Publish/Subscribe-Paradigma Message-orientierter Middleware-Ansätze [Tanenbaum u. van Steen, 2002] oder wie sie im Systemmanagement zu finden sind [Hegering u. a., 1999]. WS-Notification bezeichnet eine Familie verwandter Spezifikationen für Web Services auf der Basis *Topic*-bezogener Publish/Subscribe-Muster und beinhaltet neben der WS-BaseNotification-Spezifikation [OASIS, 2006b] auch die Broker-gestützte Notifikation (WS-BrokeredNotification) [OASIS, 2006c] und einen auf Topics aufsetzenden Subscribe-Mechanismus (WS-Topics) [OASIS, 2006n].

Figur 2.22 fasst die im WSRF-Rahmenwerk vorhandenen und die damit eng zusammenhängenden Spezifikationen zusammen.

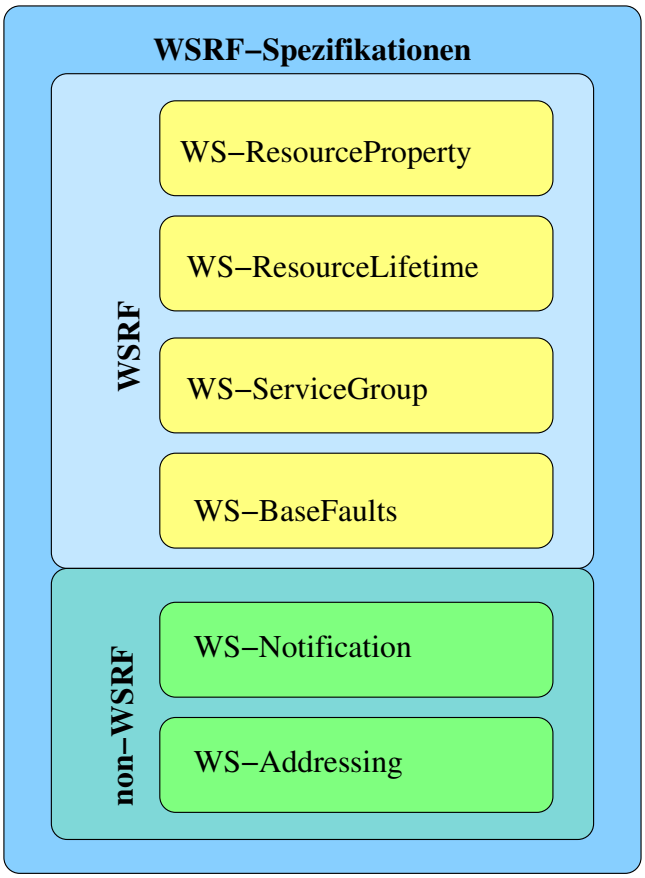


Abbildung 2.22: WSRF-Spezifikationen im Überblick

2.2.4 Zwischenfazit: Web Services-Technologien als Basis für Grids

Die Diskussionen der letzten Abschnitte haben die Bedeutung Service-orientierter Architekturen für die Bereitstellung organisationsübergreifender Dienste gezeigt. Es ist aber auch deutlich geworden, dass „Dienstorientierung“ einen konsistenten begrifflichen Kontext erfordert, um allgemeine Rahmenwerke spezifizieren zu können. Dass dieses bisher nicht (oder nur unzureichend) geschehen ist, liegt zum großen Teil daran, dass sich der SOA-Konzeptrahmen noch weitgehend in den Kinderschuhen befindet.

Mit dem generischen Dienstmodell ist eine präzise Begriffswelt für Dienste und dienstorientierte Architekturen vorgestellt worden, auf die die Nomenklatur des SOA-Referenzmodells und die des Web Services-Umfeldes abbildbar sind. Trotz der weitgehenden Kompatibilität der Begriffswelten sind Einschränkungen in den Implementierungs-

sichten nicht zu übersehen. Insbesondere die Zustandslosigkeit von Web Services macht sie für den Einsatz im Grid-Umfeld nur bedingt einsetzbar, da Grid-Dienste persistente Zustände erfordern. Diese Lücke wird durch das WS Resource Framework geschlossen, das folgerichtig auch allen modernen Grid-Middlewareansätzen zu Grunde liegt und sowohl die Spezifikation der Open Grid Services Architecture als auch die des Web Services Distributed Managements (WSDM) (siehe Abschnitt 2.3.4 und Kapitel 6) wesentlich beeinflusst.

2.3 Der Aspekt „Management“

Mit der zunehmenden Verbreitung offener Systeme und leistungsfähiger Kommunikationsnetze nimmt die Kooperation verteilter und heterogener Hard- und Softwarekomponenten eine immer größere Rolle ein. Dies gilt insbesondere seit dem Deployment großflächiger und organisationsübergreifender Web Services- und Grid-Architekturen. Der Preis dafür ist in der Regel ein sehr viel komplexeres technisches Management solcher Systeme, das nur auf der Basis standardisierter Architekturen und Frameworks effizient zu bewältigen ist. In Grids wird man sich nicht mehr wie bisher streng auf die Administration des reinen Kommunikationsnetzes (Netzmanagement), der Endsysteme (Systemmanagement) und kritischer Anwendungen (Anwendungsmanagement) beschränken können. Vielmehr müssen *alle* Komponenten dieser Umgebungen, auch die logischen wie Virtuelle Organisationen, einem integrierten Managementansatz der Gesamt-IT unterworfen werden [Hegering u. a., 1999].

Verteilte Systemumgebungen, so wie sie in dieser Arbeit betrachtet werden, unterscheiden sich erheblich bezüglich ihrer Architekturen, ihrer Größe, ihrer Komponenten und ihrer Ausrichtung. Es kann folglich auch nicht *eine* Managementlösung für alle verteilten Systeme geben. Anzustreben ist vielmehr ein „Baukasten“prinzip, in dem die Teile, die einzelne Problembereiche adressieren, flexibel orchestriert werden können, um für jede Umgebung optimale Managementlösungen bereitstellen zu können. Dazu bedarf es standardisierter Managementarchitekturen [Hegering u. a., 1999], die sich an den Dimensionen des IT-Managements orientieren (siehe Abbildung 2.23) und eine solche systemübergreifende Kombination von Managementmodulen erst ermöglichen.

2.3.1 Managementarchitekturen

Generell befasst sich das technische Management eines IT-Systems mit dessen Lebenszyklus, den Phasen und deren Übergänge, mit der Behandlung der verschiedenen Informationstypen des Systems, der verwendeten Infrastrukturkomponenten, den Funktionsbereichen und den Managementdisziplinen. Ein *Rahmenwerk* für managementrelevante Standards wird *Managementarchitektur* genannt [Hegering u. a., 1999]. Im *Informationsmodell* einer solchen Architektur werden die relevanten Managementobjekte beschrieben und es

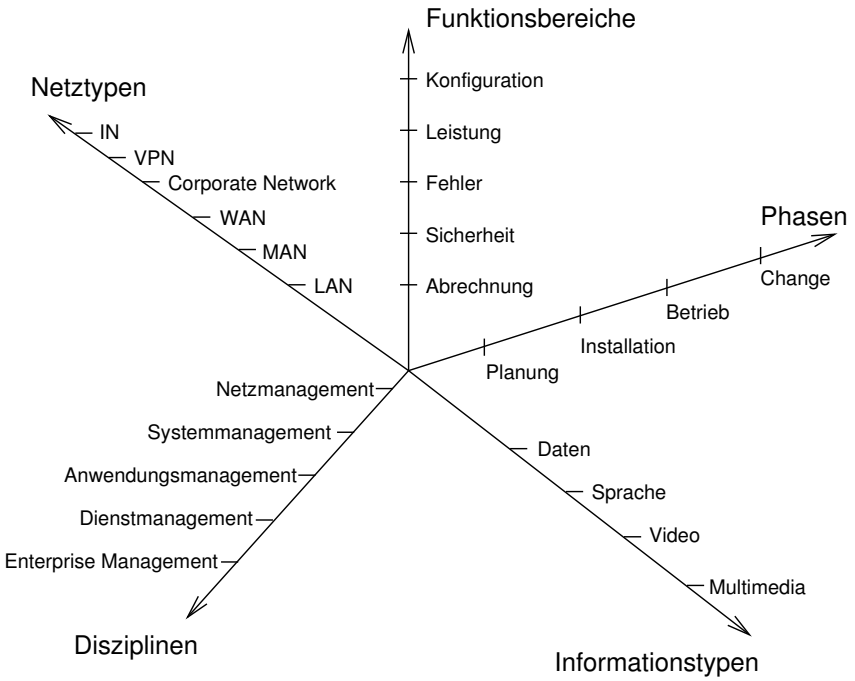


Abbildung 2.23: Dimensionen des technischen Managements nach [Hegering u. a., 1999]

erfolgt eine Festlegung der syntaktischen und semantischen Möglichkeiten zur Modellierung und Beschreibung von Ressourcen und Informationen. Im *Kommunikationsmodell* stehen die Zugriffsmechanismen auf die Managementobjekte im Vordergrund. Das *Funktionsmodell* gliedert schließlich den Komplex „Management“ in handhabbare Einheiten und definiert generische Managementfunktionen, während das *Organisationsmodell* die am Managementprozess beteiligten Rollen, Kooperationsformen und Domänen festlegt. Diese Modelle werden in den folgenden Abschnitten kurz beschrieben und in ihren Teilaufgaben dargestellt (siehe dazu auch Abbildung 2.24).

Das Informationsmodell

Mit dem Informationsmodell einer Managementarchitektur wird ein Beschreibungsrahmen für Managementobjekte bereitgestellt, indem ein *einheitliches* Format für Managementinformationen, also die Informationen, die zu Managementzwecken auszutauschen sind, festgelegt wird. Managementobjekte sind dabei als managementrelevante Abstraktion realer Ressourcen anzusehen. In einer Managementinformationsbasis (MIB), deren Struktur durch das Informationsmodell determiniert wird, werden alle von einer Rolle des Organisationsmodells verwalteten Managementobjekte zusammengefasst. Damit umfasst eine MIB die Gesamtheit der Managementschnittstellen, die ein Agent einem Manager zur Verfügung stellt. Dies beinhaltet nicht nur alle Operationen, Nachrichten und Attribute, sondern in der Regel auch Angaben über die Struktur des vom Agenten verwalteten Systems, wie z.B.

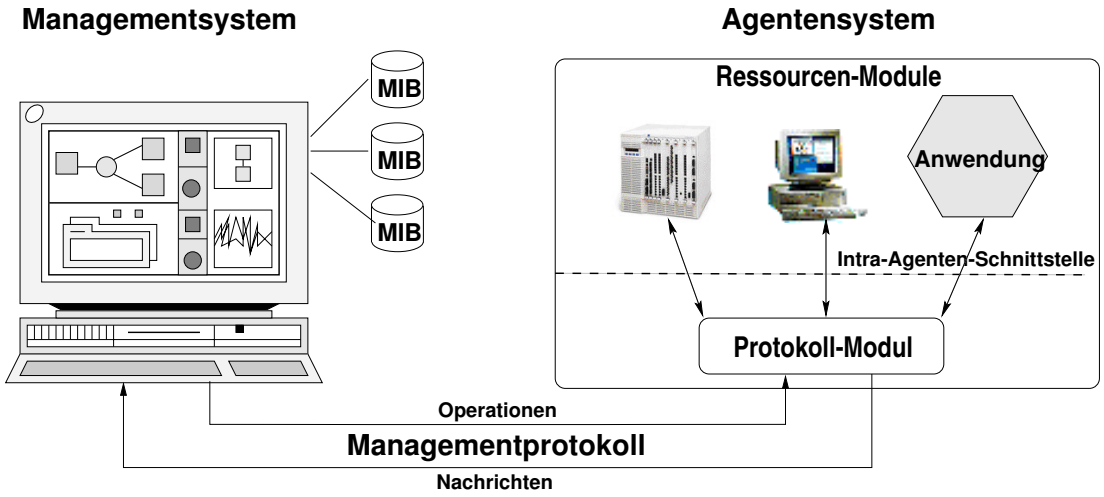


Abbildung 2.24: Management-Gesamtarchitektur nach [Keller, 1998]

Abhängigkeiten zwischen einzelnen Objektklassen. Da die Systemarchitekturen der eingesetzen Manager- und Agentensysteme völlig verschieden sein können, sind die Datenstrukturen einer herkömmlichen Programmiersprache kein geeignetes Format zur Beschreibung von Managementinformation. Es werden daher oft eigenständige, managementspezifische Notationen zur Beschreibung von Managementinformationen verwendet. Beispiele von Informationsmodellen sind in [Hegering u. a., 1999] und [DMTF, 2006a] zu finden.

Das Organisationsmodell

Im Organisationsmodell werden die Rollen der am Managementprozess beteiligten Systeme und deren jeweilige Zuständigkeitsbereiche (Domäne) definiert. Eng mit dem Domänenbegriff ist der Begriff der Zielvorgabe oder Policy verknüpft. Policies leiten aus übergeordneten Zielen bzw. Prozessen Vorgaben für das technische Management und sind dementsprechend auf verschiedenen Ebenen des IT-Managements zu finden. In der Regel werden Systeme, die aktive Rollen übernehmen, also steuernd auf andere Systeme einwirken, als Manager bzw. Managementsystem bezeichnet. Andererseits können passive Komponenten identifiziert werden, die ihrerseits von anderen Systemen administriert werden und daher im Sinne von Interaktionssystemen träge sind [Wedig, 2004]. Solche Systeme werden auch als Agenten bzw. Agentensysteme bezeichnet. Natürlich kann ein System auch beide Rollen einnehmen und dabei sowohl als Manager, als auch als Agent agieren. Solche Systeme werden dann als verteilte kooperative Managementsysteme bezeichnet [Keller, 1998].

Derzeit findet man im Umfeld des IT-Managements primär zwei Klassen von Kooperationsformen. Entweder sind sie in einer hierarchischen bzw. asymmetrischen Manager-Agenten-Konstellation angeordnet sind (wie beispielsweise in der OSI-Managementarchitektur und der Internet-Managementarchitektur) oder sie liegen in ei-

ner gleichberechtigten bzw. symmetrischen Peer-to-Peer-Beziehung vor (wie bei der Open Management Architecture).

Das Kommunikationsmodell

Im Kommunikationsmodell werden Prinzipien und Konzepte zum Austausch von Managementinformationen zwischen den im Organisationsmodell definierten Rollen festgelegt. Dazu werden die erforderlichen Kommunikationskanäle, die durch Managementprotokolle realisiert werden, definiert. Diese setzen funktionsfähige End-to-End-Verbindungen zwischen Managern und Agenten voraus und erlauben die Ausführung von Operationen durch Managementsysteme auf Agentensystemen bzw. die Zustellung von Nachrichten durch Agenten an Manager. Im Kommunikationsmodell ist daher festzulegen,

- welche Partner zur Kommunikation miteinander berechtigt sind,
- welcher Kommunikationsmechanismus verwendet wird, d.h. Protokoll- und Dienstspezifikation zum Informationsaustausch unter Berücksichtigung der zu Grunde liegenden Kommunikationsarchitektur,
- wie die Syntax und Semantik der Protokoll-Datenstrukturen festgelegt ist.

Der Austausch von Managementdaten kann in Managementarchitekturen entweder im Pull-Modus, im Push-Modus oder in einer hybriden Form erfolgen. Im Pull-Modus geht die Initiative vom Managementsystem aus, das bei den Agenten die geforderte Managementinformation abfragt. Erfolgen diese Anfragen in regulären zeitlichen Intervallen, wird auch von Polling-Verfahren gesprochen. Im Push-Modus hingegen senden die Agenten ohne vorherige Aufforderung durch das Managementsystem die notwendigen Managementinformationen selbständig. Dies ist der Fall bei asynchronen Ereignismeldungen, in denen ein Agent einen (oder mehrere) Manager über aufgetretene Zustandsänderungen informiert.

Das Funktionsmodell

Das Funktionsmodell einer Managementarchitektur gliedert die Gesamtaufgaben des Managements in adäquate Funktionsbereiche und versucht, als Basis eines „Baukastensystems“ allgemeine Managementfunktionen festzulegen. Im Funktionsmodell sind für die einzelnen Funktionsbereiche die erwartete Funktionalität und die Dienste sowie die Managementobjekte zur Erbringung der Funktionalität zu definieren [Hegering u. a., 1999]. Die im Funktionsmodell identifizierten Funktionen werden in der Regel auf Managementplattformen und Agentensystemen implementiert und verschiedensten Anwendungen über geeignete Programmschnittstellen (API) bereitgestellt. Typische Funktionalbereiche, für die Dienste im Funktionsmodell festgelegt werden, sind die so genannten FCAPS-Bereiche:

Fehlermanagement. Fehler sind Soll-Ist-Abweichungen im Verhalten von Ressourcen. Das Fehlermanagement umfasst sowohl reaktive als auch proaktive Maßnahmen. Die Hauptaufgabe des Fehlermanagements liegt in der Aufrechterhaltung einer hohen

Verfügbarkeit durch schnelle Identifizierung und Beseitigung von Fehlern im Sinne von [Avižienis u. a., 2001]. Als wesentliche Teilaufgaben des Fehlermanagements können identifiziert werden: die Überwachung der Netz- und Systemzustände, die Entgegennahme und Verarbeitung von Alarmen, die Diagnose von Fehlerursachen, die Einleitung und Überprüfung von Maßnahmen zur Fehlerbehebung, das Einrichten von Help-Desks, aber auch der Betrieb eines Trouble-Ticket-Systems (TTS).

Konfigurationsmanagement. Der Begriff „Konfiguration“ besitzt mehrere Bedeutungen. Im Bereich des Netz- und Systemmanagements bezeichnet er einerseits die Beschreibung des vernetzten Systems mit seinen Komponenten, den physischen Verbindungen und den logischen Beziehungen. Andererseits bezeichnet er auch den Vorgang der Konfiguration (Konfigurierung) als Aktivität im Zusammenhang mit der „Manipulation“ der Struktur eines verteilten Systems bzw. das Ergebnis eines solchen Vorgangs. Üblicherweise ergibt sich aus dem Systemkontext, welche Bedeutung der Begriff „Konfiguration“ jeweils hat. Das Konfigurationsmanagement umschließt alle drei Semantiken und umfasst damit das Setzen von Parametern, das Festlegen von Schwellwerten und Filtern, die Dokumentation des Gesamtsystems und seiner Evolution sowie das aktive Ändern der Konfiguration.

Abrechnungsmanagement. Die Bereitstellung von Kommunikations- und sonstigen Diensten führt zu Kosten, die typischerweise der Verursacher zu tragen hat. Wie diese Aufteilung erfolgt, ist Gegenstand von Abrechnungs-Policies. Eine wichtige Aufgabe des Abrechnungsmanagements besteht damit darin, diese Aufteilung gemäß den Vorgaben der Policies durchzuführen. Teilaufgaben des Abrechnungsmanagements betreffen die Festlegung von Abrechnungsdaten, das Führung von Abrechnungskonten, die Zuordnung von Kosten und Konten, die Kontingentverwaltung und -überwachung, das Führen von Verbrauchsstatistiken sowie die Festlegung von Policies und die Aushandlung von Tarifen.

Leistungsmanagement. Das Leistungsmanagement kann von seiner Zielsetzung her als eine Weiterführung des Fehlermanagements verstanden werden, da die Lauffähigkeit des Systems innerhalb bestimmter Dienstgüteparameter im Fokus liegt. Diese Parameter werden typischerweise in SLAs zwischen Dienstanbietern und Dienstnehmern festgelegt. Wichtige Teilaufgaben des Leistungsmanagements bestehen in der Bestimmung von Dienstgüteparametern und Metriken zur Überwachung der Dienstgüte, in der Ressourcenüberwachung hinsichtlich Engpässen, in der Durchführung von Messungen, in der Aufzeichnung von Systemprotokollen, der Aufbereitung und Aggregation von Messdaten sowie der Durchführung von Leistungs- und Kapazitätsplanungen und -anpassungen.

Sicherheitsmanagement. Sicherheitsmanagement bezeichnet schließlich alle Maßnahmen zur Gewährleistung eines sicheren und geschützten verteilten Systems, in dem die schützenswerten Ressourcen einer Organisation (z.B. Informationen, Infrastrukturen, Dienstleistungen) vor unautorisierten Zugriffen abgeschirmt werden. Dem Ver-

lust dieser Werte muss durch Sicherheitsmaßnahmen vorbeugt werden, die abhängig von einer Bedrohungsanalyse sind. Typische Bedrohungsszenarios sind passive und aktive Angriffe, aber auch Fehlfunktion von Ressourcen und Fehlbedienungen. Ausgezeichnete Überblicke über mögliche Bedrohungslagen und Abwehrmechanismen sind in [Eckert, 2006] und [Oberhaitzinger u. a., 2004] zu finden. Basierend auf den Analysen der Bedrohungslage und den zu schützenden Werten ergeben sich Sicherheitsziele bzw. -anforderungen, auf deren Grundlage Sicherheits-Policies definiert werden müssen. Wichtige Teilaufgaben des Sicherheitsmanagements bestehen somit in der Durchführung von Bedrohungsanalysen, der Identitätsfeststellung, der Sicherstellung von Vertrauen und Vertraulichkeit und Datenintegrität sowie der ständigen Berichterstattung.

2.3.2 Managementplattformen

Managementarchitekturen implizieren noch keine einheitliche Implementierung, da sie nur Rahmenwerke darstellen. Implementierungen von Managementarchitekturen unter Verwendung standardisierter Programmier- und Dienstschnittstellen werden Managementplattformen genannt. Sie stellen eine gemeinsame Infrastruktur und Ablaufumgebung für Managementapplikationen verschiedener Hersteller bereit und unterstützen diese durch eine einheitliche Darstellung, Speicherung und Verwaltung sämtlicher Managementobjekte. Typischerweise verfügen Managementplattformen über mehrere Kommunikationsmodule, die den Austausch von Managementinformation über standardisierte Managementprotokolle gestatten. Managementapplikationen basieren deshalb immer auf *konkreten* Plattformimplementierungen, was nicht nur den Verlust von Portabilität mit sich bringt, sondern auch dem Prinzip offenen Managements widerspricht. Beispiele von Managementplattformen sind Hewlett-Packard's OpenView[10], IBM's Tivoli[11], Computer Associates' Unicenter[12] oder – als jüngste Entwicklung – Oracle's SOA-Suite[13]. Der grundsätzliche Aufbau einer Managementplattform ist in Abbildung 2.25 dargestellt.

2.3.3 Spezialitäten des Managements von Diensten

Ganz allgemein erfordern die Bereitstellung und der Betrieb verteilter IT-Dienste, um diese geht es letztlich in dieser Arbeit, die Entwicklung dienstadäquater Managementkonzepte. Diese zielen einerseits auf Dienste als *managed objects*, andererseits müssen sie aber auch geeignete (Management-) Funktionalitäten vorsehen. Die hohe Komplexität des resultierenden Dienstmanagements folgt hierbei zum einem aus den potenziellen Abhängigkeiten zwischen Diensten (vertikal und horizontal [Nerb, 2001]) und zum anderen aus der verteilten Diensterbringung durch mehrere Komponenten [Dreo Rodošek, 2002]. Dass es sich bei

[10] http://openview.hp.com
[11] http://www-306.ibm.com/software/tivoli/
[12] http://www.ca.com
[13] http://www.oracle.com/technologies/soa/soa-suite.html

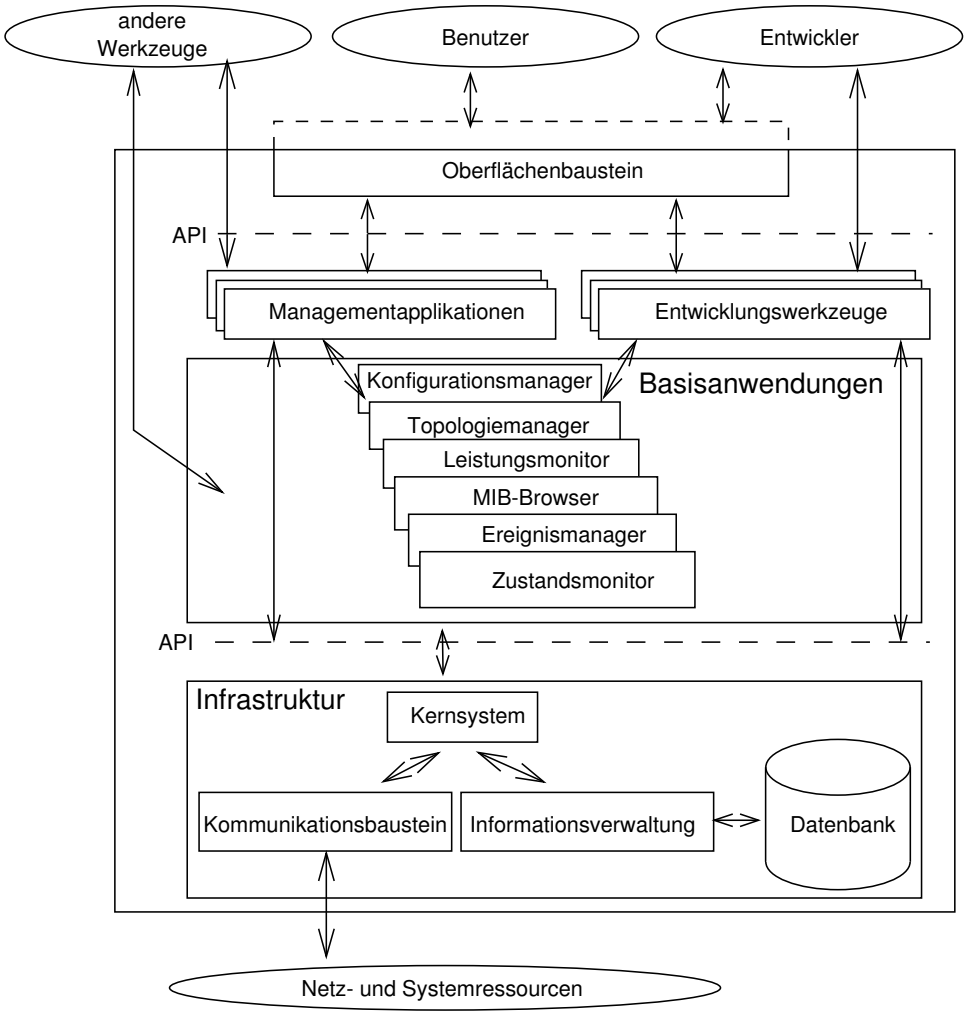


Abbildung 2.25: Aufbau von Managementplattformen nach [Hegering u. a., 1999]

diesen Komponenten um solche einer Grid-Infrastruktur handelt, erhöht die Komplexität eher als dass sie verringert wird.

Mit „Dienstmanagement“ wird der Teil der Managementpyramide bezeichnet, der zwischen dem rein technischen Management (die unteren drei Ebenen in Abbildung 2.26) und dem eher Geschäftsprozess-orientierten IT-Management (die oberen beiden Ebenen in Abbildung 2.26) angesiedelt ist.

Das Zielobjekt des Dienstmanagements ist nach [Hegering u. a., 1999] und [Dreo Rodošek, 2002] in erster Linie die managementrelevante Abstraktion der Ressource „Dienst“, der Dienst als *managed object*. Damit umfasst das Dienstmanagement in seiner allgemeinen Definition sämtliche Mechanismen, die für das Management der statischen und dynamischen Aspekte eines Dienstes erforderlich sind, also insbeondere die Einrichtung, Überwachung und Aufrechterhaltung des SAPs, des Management-SAPs, der Dienstbeschreibung, der Dienstgüteparameter, der Dienstvereinbarungen, der Dienstabhängigkeiten und der eigent-

lichen Dienstrealisierung über alle Phasen des Dienstlebenszyklus (siehe Abbildungen 2.16 und 2.14).

Abbildung 2.26 zeigt außerdem, dass Dienstmanagement stets im größeren Kontext unternehmensweiten IT-Managements gesehen werden muss. Die Schnittstelle zum Enterprise Management [Keller, 1998] impliziert eine stärkere Einbindung und Verknüpfung des Dienstmanagements mit der Aufbau- und Ablaufstruktur einer Organisation. Diese Aspekte sind in der Literatur bisher nur in Teilen behandelt und befinden sich im Fokus aktueller Forschungsarbeiten. Ein besonderes Interesse gilt dabei der Modellierung von Managementprozessen und deren Einbindung in Management-Workflows im Rahmen von Policy Based Management-Ansätzen wie sie beispielhaft in [Radisic, 2003] diskutiert werden. Andererseits erfährt das Dienstmanagement aber auch aufgrund der Abhängigkeitsbeziehungen zwischen Diensten und den Ressourcen, die diese Dienste realisieren (d.h. Netzkomponenten, Systeme, Anwendungen), eine Integrationsverpflichtung mit dem Netz-, System- und Anwendungsmanagement.

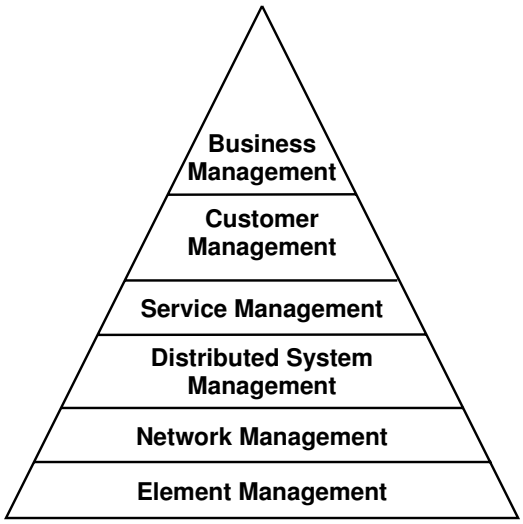


Abbildung 2.26: Managementpyramide nach [Hegering u. a., 1999]

In organisationsübergreifenden Szenarios wird deutlich, dass das Dienstmanagement nicht nur Aufgaben umfasst, die innerhalb einer Organisation wahrgenommen werden müssen. Diesem Fragenbereich widmet sich das interorganisationale Dienstmanagement [Langer, 2001]. Für weiterführende Fragestellungen sei deshalb auf diese Arbeit verwiesen.

2.3.4 Management von und mit Hilfe von Web Services

Nicht zuletzt mit dem verstärkten Aufkommen von Web Services- und Grid-Technologien hat sich in den letzten Jahren eine heterogene Landschaft von Systemmanagement-Technologien und -lösungen entwickelt. Insbesondere vor dem Hintergrund, Ressourcen und Dienste diverser Anbieter zu einem Geschäftsprozesse unterstützenden Ganzen „zu verschmelzen“, sind adäquate Integrations- und Managementmechanismen erforderlich, die nicht nur in der Lage sind, Web Services-basierte Managementlösungen anzubieten, sondern auch Web Services selbst zu managen. Dies ist der Hintergrund der Web Services Distributed Management (WSDM)-Spezifikation [OASIS, 2006g, h], der korrespondierenden WSM-Spezifikation [Arora u. a., 2005] und dem „vereinheitlichten Management“ [IBM, 2007]. Das Ziel der Spezifikationen ist eine Hersteller-neutrale und Technologie-unabhängige Infrastruktur für das Netz- und Systemmanagement.

Da auf WSDM und verwandte Konzepte in den Kapiteln 4 und 6 noch weiter eingegangen wird, wird auf eine detailliertere Darstellung an dieser Stelle verzichtet. Hier sollen lediglich

die Kernkonzepte der Spezifikationen (nämlich *manageable resource* und *manageability consumers*) am Beispiel WSDM eingeführt werden (siehe auch Abbildung 2.27).

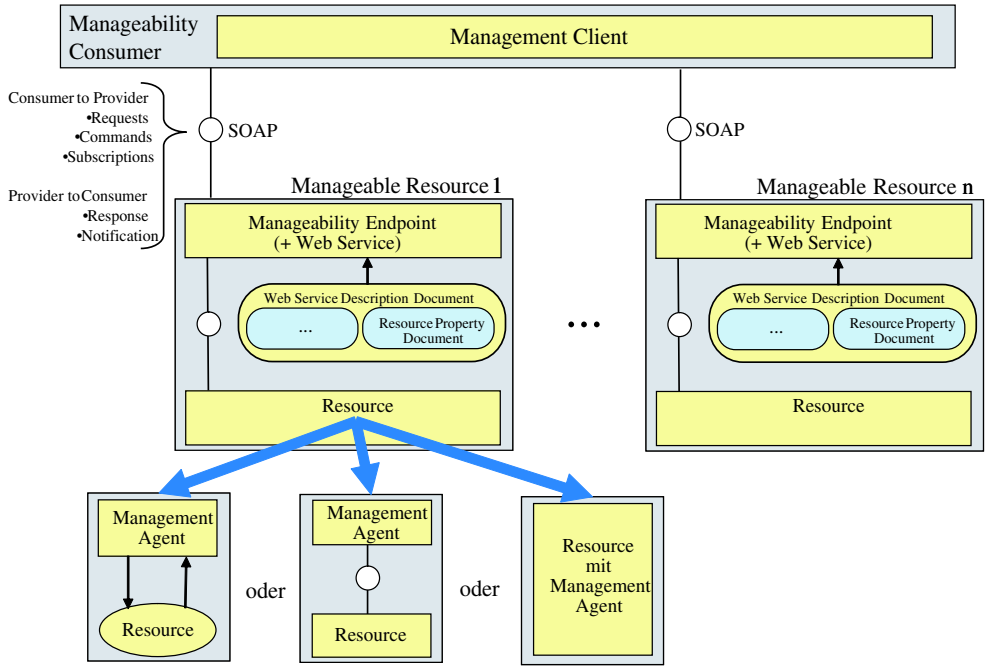


Abbildung 2.27: WSDM-Architektur (MUWS) nach [OASIS, 2006d, e, f]

Der Fokus der WSDM-Architektur liegt auf manageable resources im Sinne der WSRF-Spezifikation (Abschnitt 2.2.3). Danach muss eine manageable resource als Web Service repräsentiert werden. Folgerichtig wird auf Ressource-spezifische Managementinformationen über Web Service-Endpoints und Referenzen darauf (Endpoint Reference (EPR) im Sinne der WS-Addressing-Spezifikation [Bosworth u. a., 2004]) zugegriffen. Endpoints, die einen Zugriff auf *manageable resources* erlauben, werden manageability endpoints genannt. Ein EPR definiert das Ziel, an das ein manageability consumer Nachrichten senden kann. Ganz im Sinne der WSRF-Spezifikation kann eine *manageable resource* Notifikationen an einen *manageability consumer* schicken, vorausgesetzt, dieser hat auch die entsprechenden Nachrichten abonniert (WS-Topics [OASIS, 2006n]). Abbildung 2.28 zeigt den konzeptionellen Aufbau einer *manageable resource* nach [OASIS, 2006d, e, f].

2.3.5 Zwischenfazit: VO-Management ist Web Services Management

Die Diskussion des Managementaspektes hat verdeutlicht, dass das Management von Diensten, insbesondere im Grid- und Web Services-Umfeld, in den allgemeinen Rahmen einer generischen Managementbegrifflichkeit eingebettet werden muss (und auch kann).

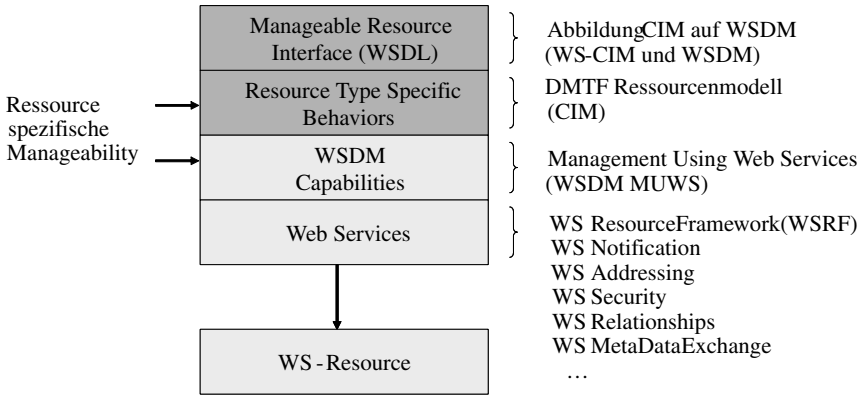


Abbildung 2.28: Konzeptioneller Aufbau einer Manageable Resource nach [OASIS, 2006d, e, f]

Es hat sich außerdem gezeigt, dass das Management Virtueller Organisationen vom klassischen Resource- und Job-Management nicht abgedeckt wird und deshalb auch keinen integralen Bestandteil aktueller Grid-Architekturen bildet. Damit kann dieser Managementaspekt auch keiner allgemein anerkannten begrifflichen Einordnung zugeführt werden. Dennoch erscheint der (Plattform-spezifische) WSDM-Ansatz nicht nur begriffstechnisch vielversprechend, wenn VOs als WS-Ressourcen im Sinne des WSRF (siehe Abschnitt 2.2.3) aufgefasst werden können, ein Aspekt, der in [Cojocaru, 2007] untersucht wird.

2.4 Zusammenfassendes Fazit

Der begriffliche Rahmen für das hier adressierte Themenfeld ist in Abbildung 2.1 angedeutet. Wegen der bisher uneinheitlichen, oft fehlenden und manchmal konfliktären, Begrifflichkeiten zum Management Virtueller Organisationen in Grids, wurden in diesem Kapitel die Aspekte „Management“, „Dienst“ und „Virtuelle Organisation“ näher betrachtet. Die Ergebnisse des Kapitels können wie folgt zusammengefasst werden:

- Virtuelle Organisationen wurden in einen organisationstheoretischen und Informatik-Kontext gesetzt und sowohl allgemein als auch Grid-spezifisch definiert und charakterisiert.
- Mit dem MNM-Dienstmodell wurde ein allgemeines Dienstmanagement-Referenzmodell rekapituliert und mit Service-orientierten Architekturen (SOA) assoziiert. Das Web Services Resource Framework (WSRF) als SOA-Instanziierung wurde entsprechend positioniert.
- Basierend auf [Hegering u. a., 1999] wurden Managementarchitekturen betrachtet und in den Web Services/Grid-Zusammenhang gebracht.

Der nächste Schritt wird nun darin bestehen, dem begrifflichen Rahmenwerk folgend, typische Grid-Szenarios zu betrachten, um Anforderungen an die geplante Architektur für das Management Virtueller Organisationen in Grids zu bestimmen. Dies geeschieht im Kapitel 3.

Kapitel 3

Spezifikation der Anforderungen

Inhalt des Kapitels

Im Kapitel 2 wurden die für eine VO-Managementarchitektur in Grids notwendigen begrifflichen Grundlagen gelegt. Dabei hat sich gezeigt, dass VO-spezifische Aspekte wie Lebenszyklen, Kooperationsformen oder Rollen einer begrifflichen Konsolidierung bedurften, um ein adäquates Rahmenwerk für VO-Managementfragen, wie es hier angestrebt wird, überhaupt entwickeln zu können. Gleichzeitig ist aber auch deutlich geworden, dass der konzeptionelle Ansatz zur Behandlung Virtueller Organisationen als *managed objects* einer grundsätzlichen Festlegung bedarf. Der nächste logische Schritt ist deshalb eine systematische Herleitung eines Anforderungskataloges für eine solche Managementarchitektur. Dies ist das Hauptanliegen dieses Kapitels. Die dabei angewandte Methodik folgt der Vorgehensweise der objektorientierten Analyse von Anwendungsfällen (*use cases*) gemäß [Bruegge u. Dutoit, 2003; Hennicker, 2006; Jacobson u. a., 1993; KBSt, 2006; Rumbaugh u. a., 1993; Rupp u. a., 2005].

Die Ausgangssituation der Diskussion manifestiert sich in den generellen Randbedingungen, die einerseits aus der Genuität Virtueller Organisationen stammen, und andererseits in den Praktiken bestehender Grid-Projekte zu beobachten sind. Im Abschnitt 2.1 wurde der erste Aspekt behandelt, Abschnitt 3.1 adressiert nachfolgend den zweiten Aspekt.

Die dabei untersuchten Szenarios sind zwar so gewählt, dass die in den Abbildungen 1.2, 2.10 und 2.12 aufgeführten Dimensionen bzw. Merkmale weitgehend abgedeckt werden, in Ermangelung eines allgemeinen Rahmenwerkes repräsentieren sie jedoch nur *current practices* Grid-spezifischer Managementansätze. Anforderungen und Randbedingungen an die Realisierung und den Betrieb von VO-Managementarchitekturen sollen jedoch nicht nur die vorgestellten Szenarios abdecken, sondern auf beliebige Grid-Umgebungen und VO-Strukturen anwendbar sein. Daher wird im Abschnitt 3.1.2 ein verallgemeinertes VO-Managementszenario skizziert, das die eigentliche Zielsituation, wie sie in Abbildung 1.3 dargestellt wird, unterstützt.

Nach der Beschreibung der Ausgangssituation mit den Anforderungen an ein VO-Management im Abschnitt 3.1 schließt sich im Abschnitt 3.2 die Spezifikation der Anforderungen an eine VO-Managementarchitektur an. Dazu werden zunächst die funktionalen Anforderungen festgelegt. Dies beinhaltet die Identifizierung der am Lebenszyklus Virtueller Organisationen beteiligten Aktoren und deren Anwendungsfälle, um daraus die erforderlichen VO-Managementoperationen und -prozesse zu gewinnnen. Im Rahmen der dort verwendeten Anwendungsfälle werden auch die fallspezifischen nicht-funktionalen Anforderungen definiert, die anschließend im Abschnitt 3.2.3 durch einige allgemeine, Anwendungsfall-übergreifende, nicht-funktionale Anforderungen ergänzt werden. Der Prozess der Anforderungsspezifikation endet im Abschnitt 3.3 mit einem ersten Grobentwurf der VO-Managementarchitektur. Abschnitt 3.4 fasst schließlich dieses Kapitel kurz zusammen.

3.1 Darstellung der Ausgangssituation

Die Ausgangssituation für die nachfolgende Anforderungsspezifikation wird einerseits durch die bestehenden VO-Managementansätze derzeitiger Grid-Projekte beschrieben und andererseits durch die generischen Charakteristika Virtueller Organisationen, wie sie aus der Einordnung im Kapitel 2 folgen. Beide Kategorien fußen stillschweigend auf einer Reihe von Grundannahmen, die mit dem speziellen Kontext der Arbeit zusammenhängen:

Grundannahme 1: Verteiltheit als Basisanforderung. VO-Management in Grids findet in *verteilten* Umgebungen statt. Daraus entstehen grundsätzliche Anforderungen, die in dieser Verteiltheit begründet sind. Sie betreffen beispielsweise die Speicherung und Übermittlung von VO-relevanten Managementinformationen oder die Protokolle, die nebenläufige Zugriffe darauf regeln müssen. Diese sind jedoch nicht VO-Management-spezifisch, sondern gelten vielmehr für alle verteilten Anwendungen. Anforderungen dieser Kategorie werden im Folgenden nicht weiter spezifiziert, da in dieser Arbeit eine reife Grid- bzw. Web Services-Infrastruktur vorausgesetzt wird, in der die Beherrschung von Verteiltheit *per constructionem* als erfüllt postuliert wird.

Grundannahme 2: Verlässlichkeit *lokaler* Managementsysteme. Das durch VOs induzierte (Co-)Managementproblem in Grids (siehe auch Kapitel 1) erfordert nicht nur die Bereitstellung von Managementinformationen durch die an den VOs beteiligten Organisationen (genauer: durch deren Managementsysteme), sondern insbesondere einen Grad von Verlässlichkeit dieser Information. Die Spezifikation verlässlicher *lokaler* Managementsysteme und -architekturen ist jedoch nicht Gegenstand dieser Arbeit, verlässliche lokale Plattformen werden vielmehr postuliert. Allenfalls werden umgekehrt *zusätzliche* Verlässlichkeitsanforderungen an derartige Systeme formuliert.

Grundannahme 3: Entwicklungswerkzeuge. Die Verfügbarkeit leistungsfähiger Werkzeuge für die Entwicklung sowohl von Managementplattformen (siehe Abschnitt 2.3.2) als auch Managementdiensten und -anwendungen stellt eine der Grundanforderungen dar, obwohl es sich hierbei um kein architekturbezogenes Kriterium im eigentlichen Sinn handelt. Derartige Werkzeuge werden in dieser Arbeit allerdings nicht bereitgestellt, stattdessen wird deren Verfügbarkeit vorausgesetzt. Im direkten Zusammenhang damit steht die Frage nach der Unterstützung von Implementierungsmodellen. Im Rahmen dieser Arbeit werden keine generischen Implementierungsmodelle betrachtet, sondern nur die in Grids auftretenden, Web Services-basierten Modelle, und dies auch nur exemplarisch.

Grundannahme 4: Leistungsindikatoren. Einen wichtigen Aspekt jeglicher Organisationskonstruktion stellen Rückkopplungsschleifen auf verschiedenen Ebenen der Wertschöpfungskette zur Erbringung von „Organisationsleistungen“ dar, um die Adaptivität von Organisationen sicherstellen und den Grad der Zielerreichung überwachen zu können. Die Behandlung derartiger Rückkopplungen erfordert neben adäquaten Leistungsindikatoren entsprechende Bewertungskriterien und darauf ausgerichtete Messverfahren. In dieser Arbeit wird auf diese Aspekte nicht näher eingegangen, geeignete Verfahren zur Instrumentierung und Messung werden stattdessen vorausgesetzt (siehe auch [Hauck, 2001]).

Grundannahme 5: Projektspezifische Fragestellungen. VOs werden für einen bestimmten Zweck gegründet und betrieben. Eine solche Zweckorientierung erfordert eine breite Palette von zweckspezifischen Funktionalitäten und Diensten. Beispiele sind projektspezifische Applikationen, allgemeine Dienste zur Tele-Überwachung und -Intervention bei der Durchführung von Experimenten und des Managements von Instrumenten und Sensoren, projektspezifische Kollaborationsdienste zur kooperativen Planung, Ausführung und Analyse von Experimenten, Management projektspezifischer Workflows, Web-Portale zur Verwendung von Softwarepaketen und VO-weiten Datenbanken sowie flexible Nutzerschnittstellen im Wissensmanagement. Obwohl im weitesten Sinn dem VO-Management zuzuordnen, werden solche projektspezifischen Fragenkomplexe hier nicht oder nur am Rande erörtert.

3.1.1 Szenarios zum VO-Management in Grids

Die vorliegende Arbeit fokussiert auf das Management dynamischer Virtueller Organisationen in Grids. Auch wenn die in den nachstehenden Abschnitten zu entwickelnde Managementarchitektur prinzipiell in allgemeinen Föderationen ebenso einsetzbar ist, werden hier – der Aufgabenstellung entsprechend – vornehmlich Grid-spezifische Fragenkomplexe zum VO-Management angesprochen. Dies drückt sich auch in den ausgewählten Szenarios aus.

Der Taxonomie in Abbildung 2.8 folgend, wird mit

- dem Distributed European Infrastructure for Supercomputing Applications (DEISA)-Projekt zunächst der Lebenszyklus langlebiger, geplanter VOs betrachtet.
- Der Betriebsaspekt isolierter (d.h. nicht überlappender) VOs steht anschließend im D-Grid-Szenario im Vordergrund,
- während Managementaspekte spontaner, ereignisgesteuerter, kurzlebiger und möglicherweise überlappender VOs im Emergency Grid (EmerGrid)-Szenario untersucht werden. EmerGrid ist zwar noch fiktiv, wird aber in Teilfragestellungen zur Zeit diskutiert [Berry u. a., 2005].
- Im abschließenden Szenario des Internationalen Polarjahres (IPY) wird schließlich noch der Aspekt von VO-Ketten vertikaler und horizontaler Art (die Makroebene in Abbildung 2.5) aufgegriffen[1].

Für jedes Szenario wird im Rahmen einer kurzen Einführung die Positionierung im Raster der Abbildung 2.10[2] dargestellt. Anschließend wird der spezielle Aspekt des VO-Managements näher betrachtet.

Szenario I: Einfache Virtuelle Organisation in DEISA

Der Fokus dieses Szenarios liegt auf dem Lebenszyklus langfristig angelegter und geplant gegründeter VOs, in denen Benutzergruppen mit unterschiedlichen Rechten existieren. Das Szenario entspricht damit der klassischen Situation vieler heutiger Grid-Projekte [Johnston, 2003]. Das hier besprochene DEISA-Szenario kann wie in Abbildung 3.1 positioniert werden.

Kurzbeschreibung des Szenarios
Die Distributed European Infrastructure for Supercomputing Applications (DEISA) [DEISA, 2006] ist ein europäisches Konsortium führender nationaler Supercomputing-Zentren.

[1] IPY ist jedoch nicht Grid-basiert.
[2] Einige Merkmale werden nur unvollständig berücksichtigt, da sie entweder zu anwendungsspezifisch ausgeprägt oder Grid-untypisch sind.

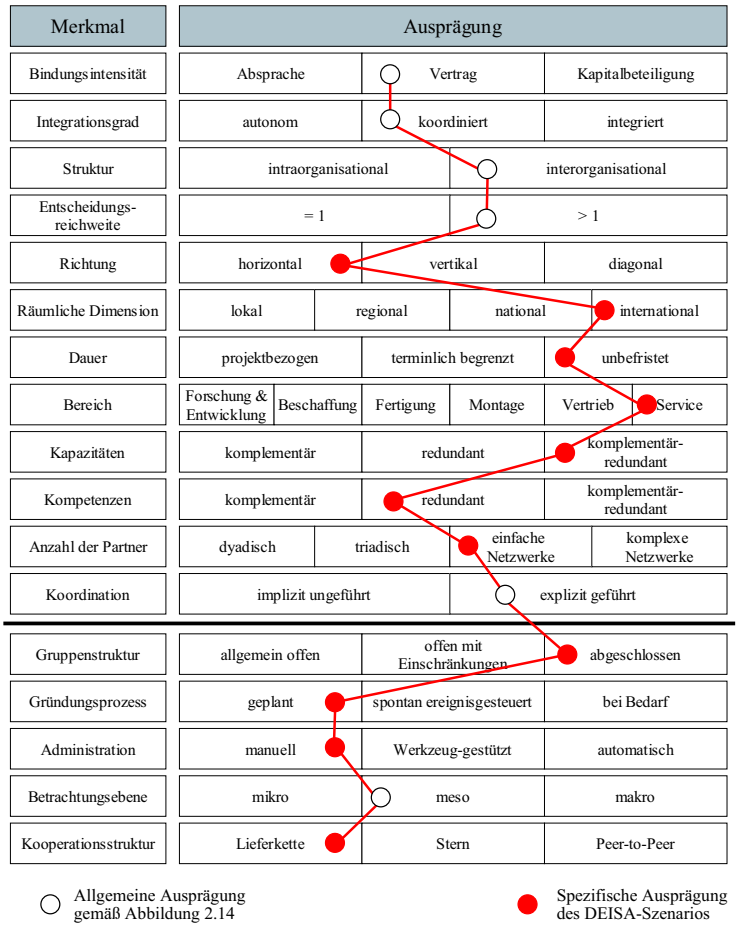


Abbildung 3.1: Positionierung des DEISA-Szenarios im Raster von Abbildung 2.10

Der Anspruch des Konsortiums ist die Bündelung von Aktivitäten im High Performance Computing (HPC) mit dem Ziel, gemeinsam eine verteilte Supercomputing-Ressource – das DEISA Virtual Supercomputing Center – im Peta-Maßstab über ultraschnelle Netze bis 10Gbps (siehe Abbildung 3.2) auf der Basis moderner Grid-Technologien zu schaffen und zu betreiben. Die festen Mitglieder des Konsortiums (die so genannten *DEISA-Sites*) sind das Barcelona Supercomputing Centre (BSC), das italienische Consortio Interuniversitario per il Calcolo Automatico (CINECA), das Finnish Information Technology Centre for Science (CSC), die britischen Zentren der University of Edinburgh and CCLRC (EPCC/HPCx) und das European Centre for Medium-Range Weather Forecast (ECMWF). Aus Deutschland nehmen das Forschungszentrum Jülich (FZJ), das High Performance Computing Centre Stuttgart (HLRS), das Leibniz-Rechenzentrum der Bayerischen Akademie der Wissenschaften (LRZ) und das Rechenzentrum der Max-Planck Gesellschaft Garching (RZG) teil, aus Frankreich das Institut du Développement et des Ressources en Informatique Scientifique (IDRIS) sowie aus den Niederlanden das Dutch National High Performance Computing and Networking Centre (SARA).

Das DEISA-Projekt fokussiert auf zwei Ebenen: Im Infrastrukturbereich steht im *Kernprojekt* reines HPC über die bereitgestellten Bandbreiten im Rahmen vereinbarter Kontingente im Vordergrund. Zum Einsatz kommen dabei solche Grid- und Multi-Cluster-Technologien, die für eine feste Kopplung des pan-europäischen DEISA-Superclusters notwendig sind (Single System Image, Global File System). Im Erweiterungsprojekt eDEISA sollen klassische Grid-Dienste in Kooperation mit anderen europäischen HPC-Standorten und Forschungsinfrastrukturen betrieben werden, die den DEISA-Supercluster als reine Grid-Ressource sehen. eDEISA geht dabei von einem technologieübergreifenden Betriebsmodell aus, das folgerichtig Transparenz auf der Anwenderebene (Anwender sollen nicht

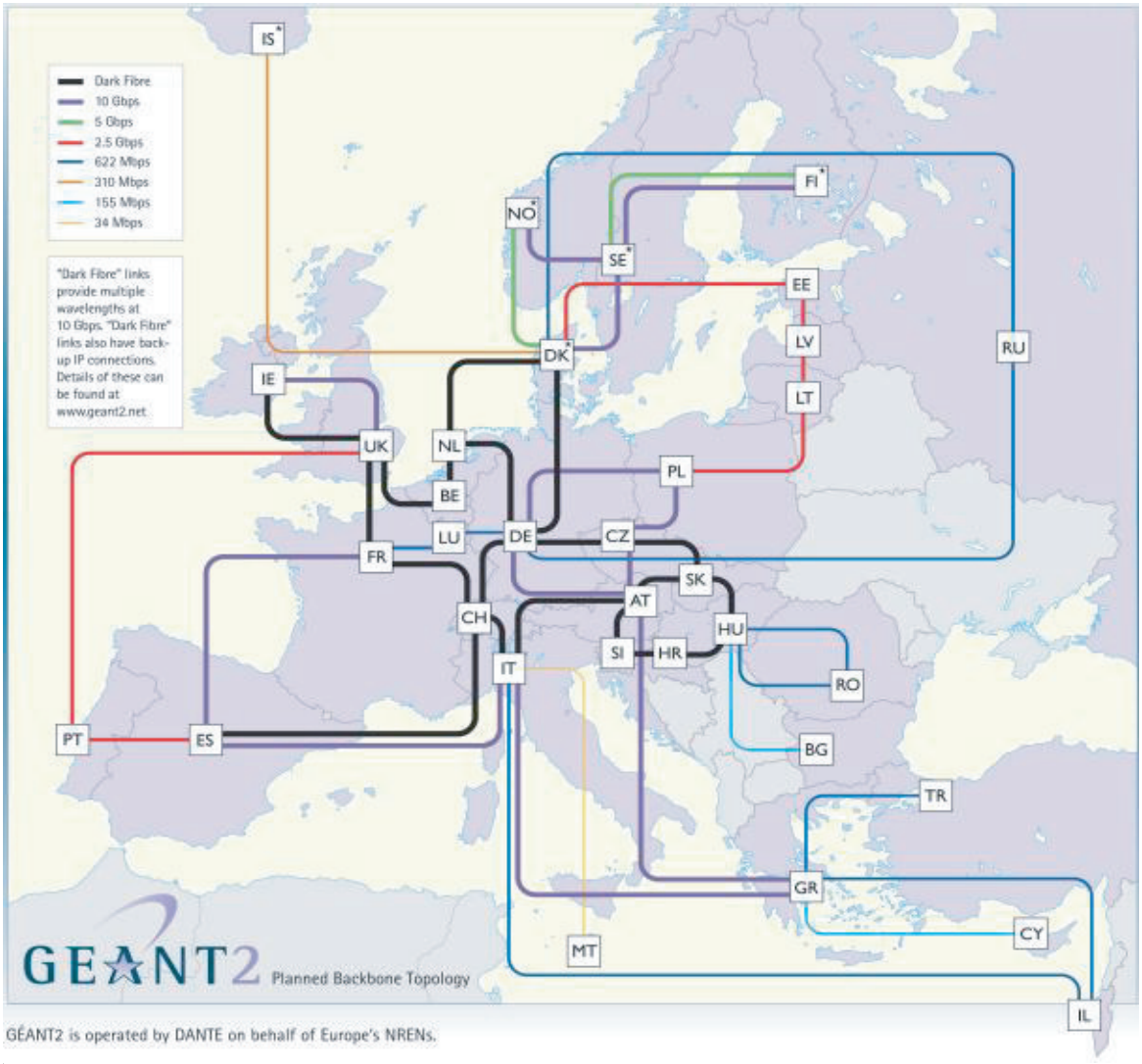


Abbildung 3.2: Géant2-Infrastruktur für DEISA nach `http://www.geant2.net`

mit der komplexen Grid-Technologie konfrontiert werden) und der Anwendungsebene (minimaler Eingriff in Anwendungen bei Technologiewechseln) fordert. Zu beachten ist jedoch, dass im DEISA-Kontext der Konsortiumsbegriff keinesfalls komplette Organisationen umfasst, sondern nur *Teile* von Organisationen in Form von Personen und technischen Ressourcen bzw. Kontingenten von Ressourcenkapazitäten.

Die *Anwendungsebene* bildet die zweite Projektebene. Hier stehen die wissenschaftlichen Anwendungsprojekte im Fokus. Diese werden primär in der DEISA Extreme Computing Initiative (DECI) jährlich ausgeschrieben und durch die so genannte Applications Task Force (ATaskF) des Konsortiums unterstützt.

VO-Management in DEISA

DEISA kennt vor dem Hintergrund seiner UNICORE-Historie [Unicore Forum, 2006] zwar

keinen differenzierten VO-Begriff, dennoch kann für den Zweck dieser Arbeit das DEISA-Konsortium als solches als VO aufgefasst werden. Im Sinne der Definition 1 (Seite 40) stellt das Konsortium nämlich eine langlebige, trotzdem zeitlich begrenzte, Kooperation von HPC-Zentren dar, die Kontingente ihrer Rechenkapazitäten aggregieren und als virtuelle Ressource derart zur Verfügung stellen, dass die Mitglieder des Konsortiums – unter Berücksichtigung lokaler und globaler Policies – die von ihnen angestrebten Ziele erreichen können. Das Konsortium ist zudem insofern stabil, als sich einerseits die partizipierenden Partner langfristig gebunden haben und andererseits neue Partner dem Konsortium ausgesprochen selten beitreten[3].

Unabhängig von dieser Betrachtungsweise rücken mit der geplanten Erweiterung im Rahmen des eDEISA-Projektes [DEISA Konsortium, 2005] aber auch in DEISA das Globus Toolkit und die Middleware der japanischen National Research Grid Initiative (NAREGI) [Miura, 2006] zunehmend in den Mittelpunkt – und damit auch VO-spezifische Aspekte, die sich u.a. in der simultanen Existenz sich überlappender VOs ausdrücken können.

Um die weiteren Überlegungen zu vereinfachen, wird vor diesem Hintergrund für den Rest dieses Abschnitts das DEISA-Konsortium als „DEISA-VO“, oder kurz „VO“, bezeichnet – wenn keine Missverständnisse zu befürchten sind.

Organisatorischer Rahmen des VO-Managements

Die DEISA-VO bildet den organisatorischen Rahmen für die Bereitstellung des virtuellen Superclusters, zur Administration von Benutzern und zur Formulierung von Policies zur Regelung der Verwendung des Clusters. Sie wird von dem DEISA Executive Committee (DEC) als oberster Instanz gemanagt. Abbildung 3.3 zeigt dies im Überblick.

VO-Management vollzieht sich in DEISA konzeptionell auf zwei Ebenen: Während auf einer Metaorganisationsebene (siehe Abschnitt 2.1.1) der Fokus auf dem *Lebenszyklus* der (in diesem Fall einzigen) VO und dessen Management liegt, konzentriert sich die VO-Ebene auf das Management von Mitgliedschaften und Ressourcenzuteilungen zur und innerhalb der VO.

Für die Metaorganisation sind VO-spezifische Gremien verantwortlich. So institutionalisiert jedes an der DEISA-VO teilnehmende nationale Rechenzentrum (z.B. das LRZ) einen Lenkungsausschuss (*National Scientific Evaluation Board*), der die eigenen Ressourcen zur Verwendung im Rahmen eingereichter Projekte freigibt. Jedes Rechenzentrum fungiert damit auch stets als Resource Provider (RP). Die Entscheidung, wann und wie diese Ressourcen genutzt werden dürfen, obliegt jedoch der VO, die auch mögliche Konfliktfälle autark entscheidet. Damit liegt die „Ressourcenhoheit“ nach wie vor bei den RPs, den eigentlichen Ressourcen„besitzern“, da die DEISA-VO für die Durchführung eines Projektes das Plazet mindestens eines Lenkungsausschusses benötigt. *Verwaltet* wird die Ressource allerdings von der VO.

Das VO-Management muss sich in DEISA mit verschiedenen Zugangsklassen auseinandersetzen:

[3] zuletzt das LRZ, Garching, und das HLRS, Stuttgart, im Mai 2005

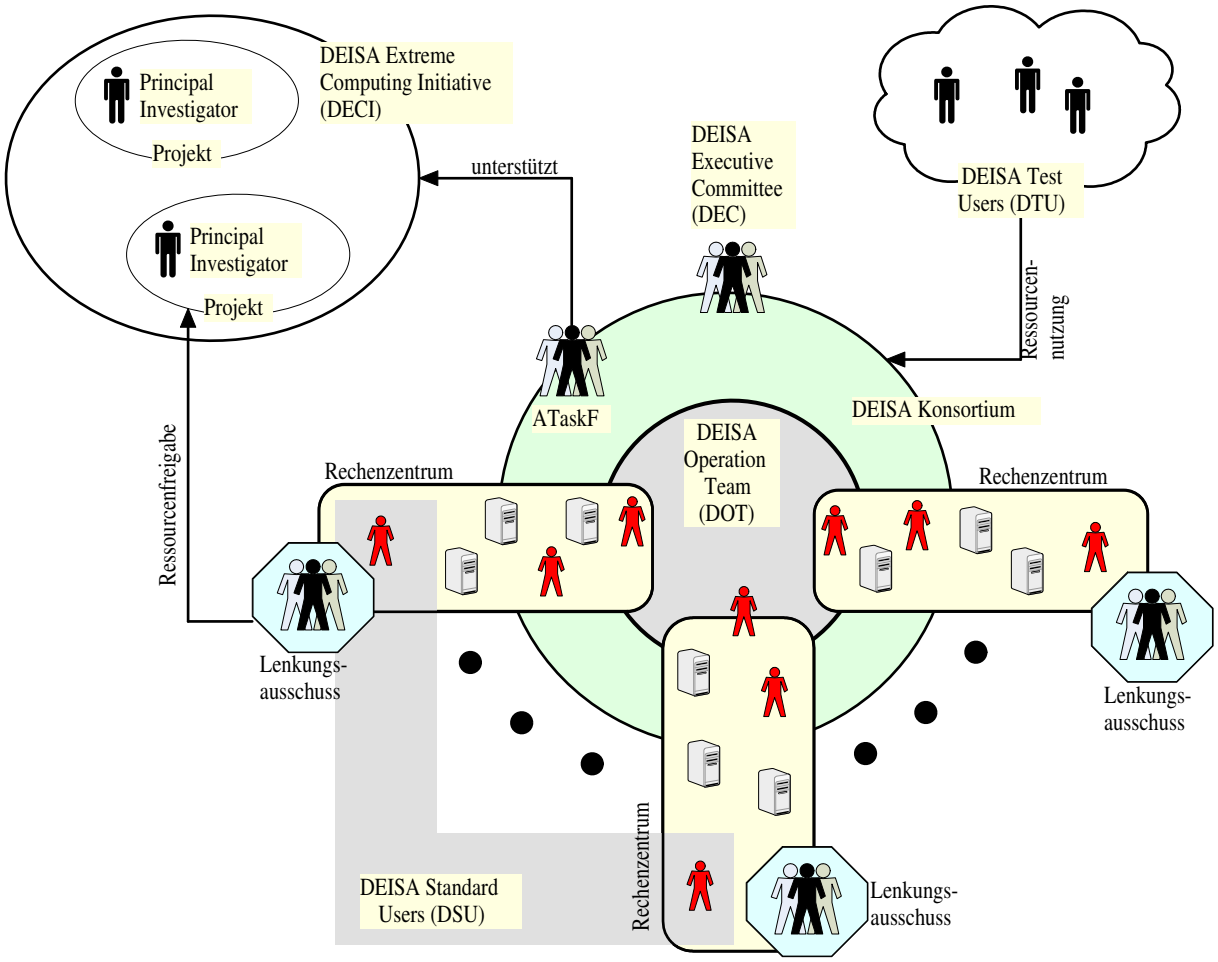


Abbildung 3.3: Organisatorischer Rahmen des VO-Managements in DEISA

- Die Klasse der DEISA Test User (DTU) ist *personen*spezifisch orientiert. Mitglieder dieser Klasse können für die Dauer von drei Monaten individuell auf VO-Ressourcen zugreifen („Schnupperzugang“). Die Mitgliedschaft in einer realen Organisation „Rechenzentrum“ wird dabei nicht vorausgesetzt.

- Die Klasse der DEISA Standard User (DSU) ist *organisations*spezifisch orientiert. Mitgliedern dieser Klasse ist zunächst nur der Zugriff auf reale Rechenzentrumsressourcen im Rahmen zuvor durch den Lenkungsausschuss bewilligter Projekte gestattet. Die Ausweitung des Zugriffs auf die virtuelle DEISA-Ressource ist fallweise erlaubt. Ein typisches Beispiel bildet die Klasse der Simulationsanwender am Höchstleistungsrechner in Bayern (HLRB) II des LRZ.

- Die Klasse der DECI User ist *projekt*spezifisch orientiert. Mitglieder dieser Klasse beantragen den exklusiven Zugang zur virtuellen DEISA-Ressource im Rahmen von

Projekten beim lokalen Lenkungsauschuss. Die Projektvergabe erfolgt jährlich neu[4]. Jedes *Projekt* wird durch einen Principal Investigator (PI) repräsentiert und mit Unterstützung der ATaskF der VO „DEISA-fiziert“. In diese Klasse fallen auch die Joint Research Activities (JRA) des initialen DEISA-Programms, die hier aber nicht weiter verfolgt werden und in Abbildung 3.3 auch nicht separat dargestellt sind.

- Die Klasse der Administratoren im DEISA Operation Team (DOT) ist *VO*-spezifisch orientiert. Mitglieder dieser Klasse verwalten im Rahmen des „täglichen Betriebes“ den Zugang zur virtuellen DEISA-Ressource. Jedes Rechenzentrum stellt mindestens einen Administrator bereit.

VO-Management in der Formationsphase

Die Gründung der DEISA-VO ist das Ergebnis eines Konsortialvertrages, initiiert im Mai 2004 durch das sechste Rahmenprogramm der Europäischen Kommission (FP6). Einen wesentlichen Bestandteil des VO-Formationsprozesses bildet die Institutionalisierung einer Rechenzentren-übergreifenden VO-Managementstruktur (siehe Abbildung 3.3), bei der der lokalen Ressourcenhoheit (über die Lenkungsausschüsse und das Operations Team) ebenso Rechnung getragen wird wie den VO-weiten Belangen (mit dem Executive Committee). Während der Formationsphase wird neben der Einrichtung der Managementstrukturen auch die Grundlage für die Erbringung der Organisationsleistung (im Sinne des Abschnitts 2.1) durch die Initialisierung der VO-Struktur und -Prozesse gelegt. Dies beinhaltet neben der Bereitstellung der virtuellen Ressource „Supercluster“ inklusive der dafür erforderlichen Zugangs- und Anwendungsdienste auch die Akkreditierung der Nutzer in den verschiedenen Zugangsklassen. Im letzten Schritt des Initialisierungsprozesses wird die sich anschließende Betriebsphase vorbereitet, indem die erforderlichen Managementdienste und -informationssysteme bereitgestellt und initialisiert werden. Abbildung 3.4 zeigt die wesentlichen am Formationspozess beteiligten Rollen und Anwendungsfälle.

VO-Management in der Betriebs- und Anpassungsphase

Während der Betriebsphase der VO (Abbildung 3.5) können sich Benutzer für den Zugang zum DEISA-Supercluster bei ihren „Heimat“-Einrichtungen in derselben Art und Weise wie für den Zugang zu lokalen HPC-Systemen registrieren. Erst nach einer erfolgreichen Authentifizierung autorisiert die DEISA-VO den Zugriff auf die von ihr verwalteten Ressourcen. Diese Akkreditierungsprozedur variiert mit den Zugangsklassen (Der Zugang zu VO-Ressourcen erfolgt beispielsweise bei DECI-Usern projektspezifisch.).

Die DEISA-VO agiert im Rahmen der Authentifizierungs- und Autorisierungsprozedur als Registrierungsstelle (RA) und stellt sicher, dass für die Anwender gültige Zertifikate ausgestellt werden und der Zugang zur DEISA-Infrastruktur den Rechten entsprechend gewährt wird.

[4]Beispiele für operative Projekte der Jahre 2005 bis 2007 sind auf den Web-Seiten `https://www.deisa.org/applications/projects2005-2006/index.php` und `https://www.deisa.org/applications/projects2006-2007/index.php` zu finden

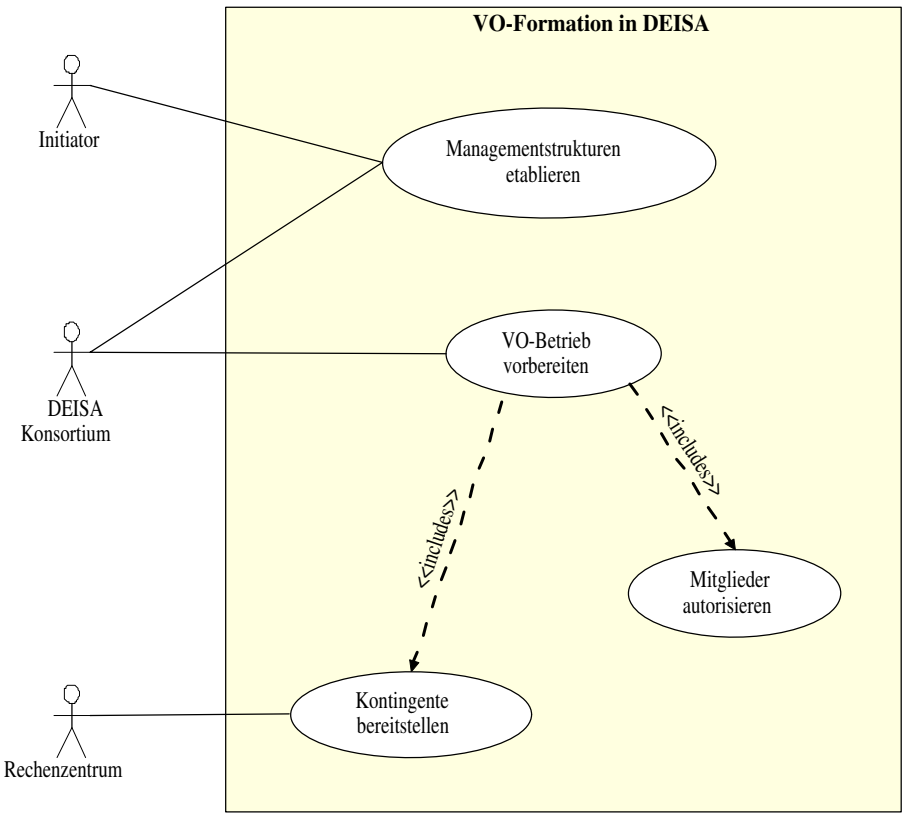


Abbildung 3.4: VO-Management in DEISA: Formationsphase

In der DEISA-VO akzeptieren alle Partner die von den nationalen Zertifizierungsinstanzen (CA) signierten Zertifikate. Eine CA kann dabei entweder durch ausgezeichnete Standorte wahrgenommen werden oder durch eine externe Institution. Die Abbildung von X.509-Zertifikaten auf lokale Benutzerkennungen liegt im Verantwortungsbereich der Administration eines jeden DEISA-Partners. In DEISA werden EUGridPMA-konforme (European Policy Management Authority for Grid Authentication in e-Science) (EUGridPMA) Zertifikate verwendet. EUGridPMA[5] ist eine internationale Organisation zur europaweiten Koordination von Grid-Authentifizierungen.

DEISA erlaubt den Zugriff auf VO-Ressourcen auch Anwendern, die nicht Mitglied der VO sind. Diese Zugänge werden von der VO fallweise entschieden und sind prinzipiell kostenpflichtig.

Aus der Kontingentierung der Ressourcen ergibt sich für das VO-Management während der Betriebsphase die Notwendigkeit eines Kontingentmanagements für beantragte Projekte. Welche Maßnahmen müssen getroffen werden, wenn Kontingente *vorzeitig verbraucht* worden sind? Welche Maßnahmen müssen getroffen werden, wenn Kontingente nach Ab-

[5]Zur Zeit existieren in Deutschland zwei EUGridPMA-kompatible CAs, der DFN-Verein (`http://www.pca.dfn.de/dfn-pki/certification`) und das Forschungszentrum Karlsruhe (`http://www.gridka.de/cgi-bin/frame.pl?seite=/ca/d_inhalt-en.html`).

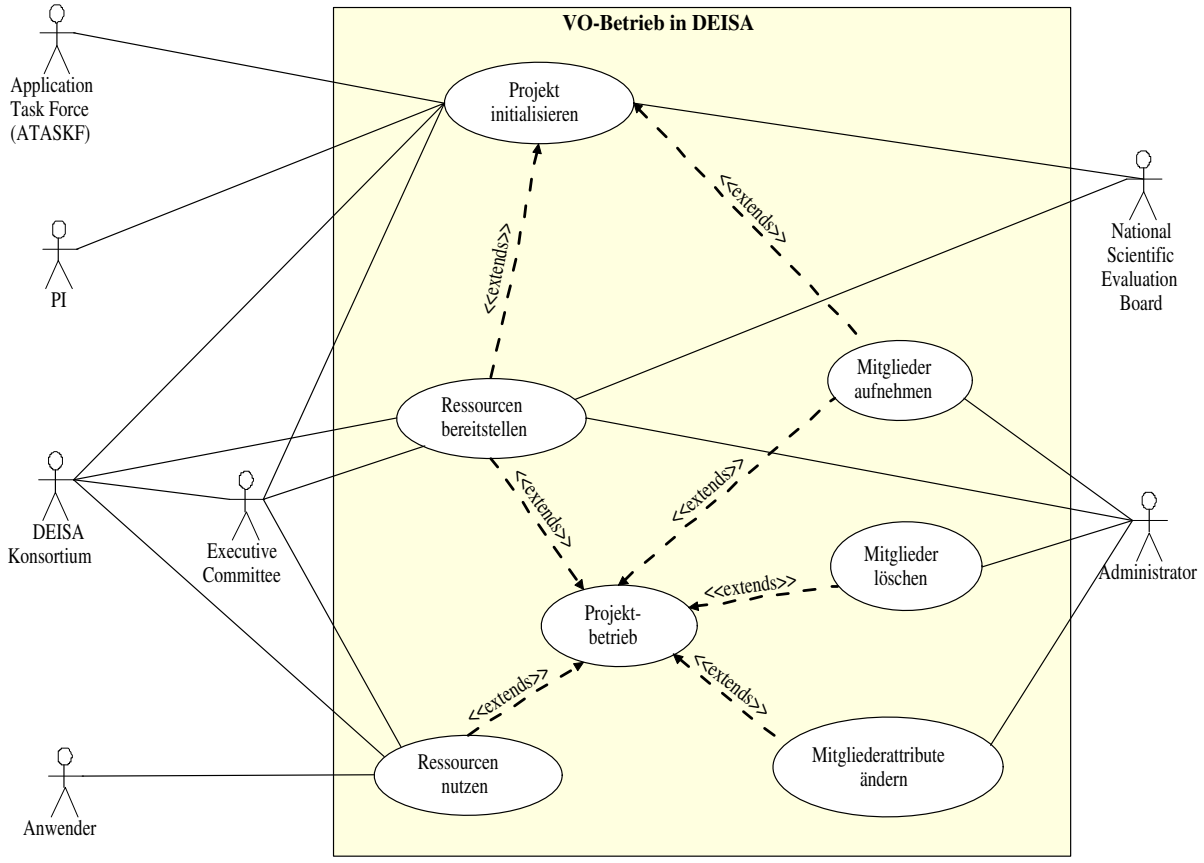


Abbildung 3.5: VO-Management in DEISA: Betriebsphase

schluss der Projekte *nicht ausgeschöpft* worden sind? Die Klärung dieser Fragen obliegt dem DEC in enger Absprache mit den Lenkungsausschüssen.

VO-Management in der Auflösungsphase

Die DEISA-VO wird langfristig gegründet. Insofern sind Mechanismen zur gezielten Auflösung dieser Virtuellen Organisation zur Zeit nicht vorgesehen. Dennoch ist zu beachten, dass schon während der Betriebsphase die mögliche Auflösung der VO antizipiert wird und die Unterstützung zukünftiger *post mortem* Audits durch geeignete Maßnahmen (z.B. umfangreiches Logging) vorbereitet wird.

Szenario II: Communities und VOs im D-Grid

Der Fokus dieses Szenarios liegt auf dem Lebenszyklus projektbezogener Virtueller Organisationen, die auf einer nachhaltig angelegten Grid-Infrastruktur im Kontext diverser so genannter Communities gegründet, betrieben und aufgelöst werden. Durch diese Ausrichtung wird das VO-Management im übrigen in eine heterogene Technologielandschaft eingebunden, in der neben UNICORE (wie im DEISA-Szenario) auch gLite [LCG, 2005]

und vor allem das Globus Toolkit [Foster u. Childers, 2005] als Middleware Verwendung finden. Das hier besprochene D-Grid-Szenario wird wie in Abbildung 3.6 positioniert.

Kurzbeschreibung des Szenarios

Im Januar 2003 wurde von der deutschen Wissenschaft die D-Grid-Initiative ins Leben gerufen. Die Wissenschaftler haben dann im Juli 2003 ein grundlegendes D-Grid-Strategiepapier veröffentlicht [D-Grid-Initiative, 2004]. Dort wurden insbesondere die Auswirkungen der neuen Grid-Technologie auf das wissenschaftliche Arbeiten thematisiert und ein entsprechendes Forschungs- und Entwicklungsprogramm vorgeschlagen.

Das Bundesministerium für Bildung und Forschung (BMBF) hat daraufhin im März 2004 eine übergreifende e-Science-Initiative für Deutschland angekündigt, der die Ausschreibung für die e-Science-Bereiche e-Learning, Wissensmanagement und D-Grid folgte. In der Bekanntmachung des BMBF vom November 2005 wurde die Vision einer digitalen wissenschaftlichen Infrastruktur vorgestellt, die es global vernetzten und international kooperierenden Wissenschaftlern ermöglichen soll, den permanenten Austausch, die Dokumentation und die unmittelbare Veröffentlichung von Forschungsergebnissen durchzuführen. Effizienz und Stabilität werden auch bei sehr großen, heterogenen Mengen von Mess-, Labor- und Rechenergebnissen, Speichern und Computern angestrebt. Für diese Vision hat sich international der Begriff e-Science etabliert [Fox u. Walker, 2003].

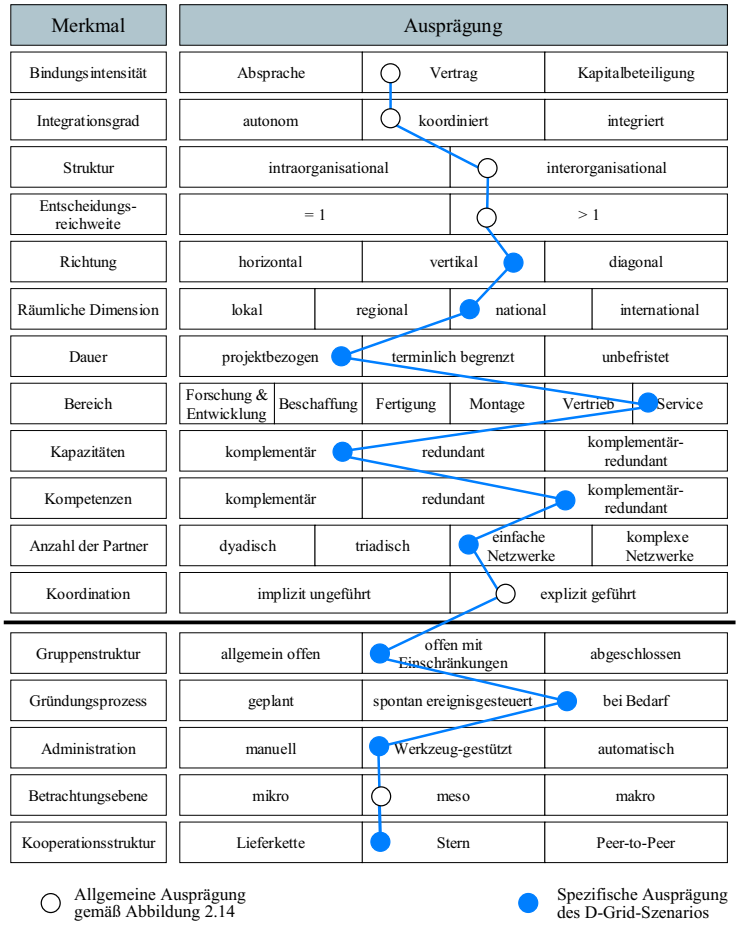


Abbildung 3.6: Positionierung des D-Grid-Szenarios im Raster von Abbildung 2.10

In Folge dieser Initiative haben am 1. September 2005 sechs so genannte Community-Projekte und das D-Grid-Integrationsprojekt damit begonnen, diese auf Nachhaltigkeit angelegte Grid-Infrastruktur aufzubauen. Zur Zeit beteiligen sich die folgenden Communities an der D-Grid-Initiative (siehe auch Abbildung 3.7): Astronomie mit dem AstroGrid-D, Klimaforschung mit dem C3-Grid, Hochenergiephysik mit dem HEP-Grid, die Ingenieurswissenchaften mit dem InGrid, die medizinische Grundlagenforschung mit dem Me-

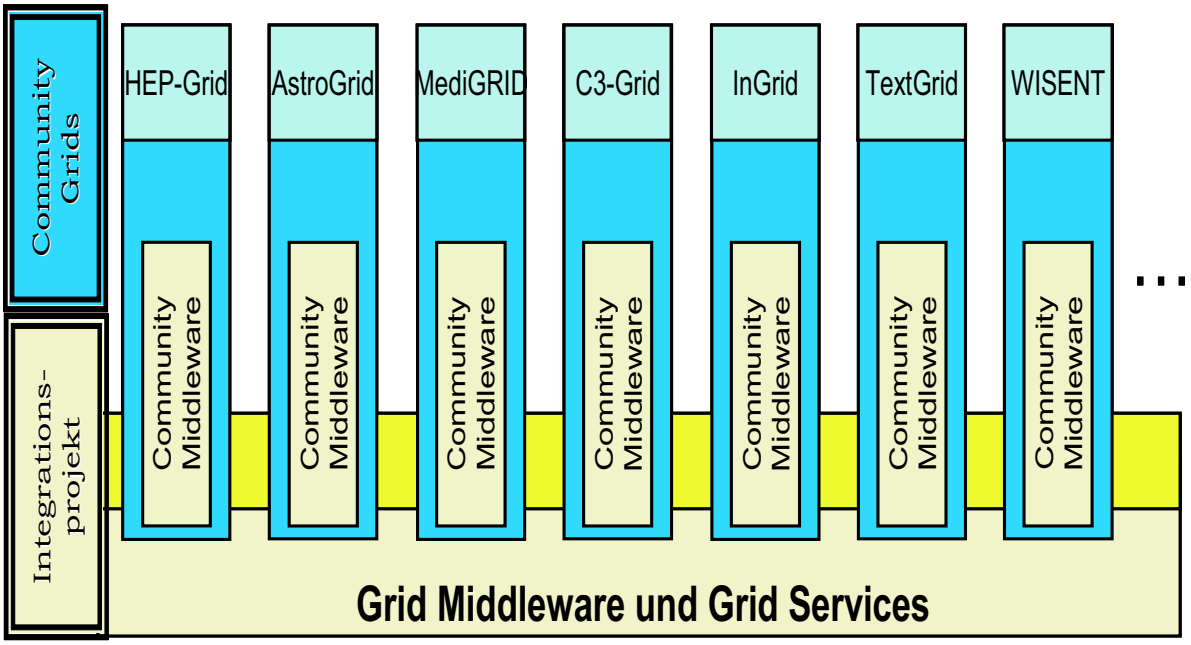


Abbildung 3.7: Allgemeine D-Grid-Projektstruktur nach [Gentzsch, 2005]

diGRID[6], die Geisteswissenschaften mit dem TextGrid und die Energiemeteorologie mit WISENT [Gentzsch, 2005].

Allen Communities gemeinsam ist – entsprechend der e-Science-Vision – der Wunsch nach einer tiefgreifenden Verbesserung der wissenschaftlichen Leistungsfähigkeit und Qualität, der gemeinschaftlichen Entwicklung und der gegenseitigen Öffnung von Arbeitsverfahren, Software, Datenbeständen, Rechnern und Instrumenten auf der Grundlage eines schnellen Kommunikationsnetzes.

VO-Management im D-Grid

Um die im D-Grid angedachten umfangreichen e-Science Aufgaben zu koordinieren, bedarf es nicht nur einer geeigneten IT-Plattform, sondern auch einer übergeordneten Instanz, die die Abwicklung und Steuerung der essenziellen Geschäftsfälle für die Anwender im D-Grid bewerkstelligt. Hierzu wird eine virtuelle Plattform bereitgestellt, über die sie Geschäftsfälle initiieren und elektronisch abgewickeln können. Anwender, Administratoren und Ressourcen bilden so eine zeitlich begrenzte Virtuelle Organisation (VO).

Im D-Grid fokussiert das VO-Management vornehmlich auf ein effizientes Mitgliedermanagement, adressiert aber auch die Sicherstellung der Verfügbarkeit von Ressourcen und Diensten in definierten Zeiträumen mit definierten Qualitätsanforderungen für eine Nutzung durch Community-VOs. Das Management von VOs umfasst damit nicht nur die Bildung personeller Kollaborationen, sondern ordnet auch menschliche und technische

[6]Die D-Grid-Community MediGRID ist nicht zu verwechseln mit dem MEDIGrid [Eftichidis, 2005], das im nachfolgenden Szenario von Bedeutung sein wird.

Ressourcen in einen organisatorischen Zusammenhang und bildet so die Schnittstelle zwischen den D-Grid-Service Providern und den Communities. Typische Geschäftsfälle, die im Kontext einer VO geregelt werden, sind die Bildung von Projektgruppen (Anmeldung, Abmeldung, etc.), die Identifizierung von Ansprechpartnern und Verantwortlichkeiten nach innen und außen, die Festlegung von Abstimmungsregularien, das Buchen von Ressourcen, der Abschluss von SLAs, die Festlegung von Abrechnungsmodalitäten für die im D-Grid in Anspruch genommenen Leistungen oder die Bereitstellung einer Kommunikationsplattform für die Mitglieder der VO (Virtuelles Büro).

Organisatorischer Rahmen des VO-Managements

VOs werden in den verschiedenen D-Grid-Communities unterschiedlich intensiv und formal genutzt. So haben einige Communities, wie etwa die Hochenergiephysik (HEP), VOs schon vor der D-Grid-Initiative für die Autorisierung von Benutzern von HEP-Ressourcen genutzt. Andere Communities, wie etwa MediGRID oder InGrid, haben dagegen Virtuelle Organisationen bisher formal nicht eingesetzt. Auch im Kern-D-Grid[7] wurden VOs bisher zur Authentifizierung nicht genutzt. Im D-Grid existieren folglich zum großen Teil Community-spezifische Lösungen, die allerdings nicht den Anspruch eines D-Grid-weiten VO-Managementstandards erheben. VOs existieren in der Regel jeweils für die lokal verwalteten Ressourcen, Community-übergreifende VOs sind nicht vorhanden. Verfahren zur VO-Gründung, die Beantragung von Mitgliedschaften oder die Beendigung von VOs sind genauso Community-spezifisch wie die verwendeten Technologien und Werkzeuge. Einige Communities favorisieren als Autorisierungsmechanismus den aus dem Enabling Grids for E-sciencE (EGEE)-Projekt[8] bekannten VO Membership Service (VOMS) [Alfieri u. a., 2003], andere erweitern diesen Ansatz um die Registrierungskomponente VO Management Registration Service (VOMRS) [VOX Project Team, 2004] oder nutzen stattdessen für Autorisierungs- und Authentifizierungszwecke eine Shibboleth-Infrastruktur [Watt u. a., 2006]. Dementsprechend ist auch der organisatorische Rahmen des VO-Managements im D-Grid Community-spezifisch, orientiert sich aber prinzipiell an den Rollen und Anwendungsfällen in Abbildung 3.8. Beispiele der unterschiedlichen Ansätze der Communities sollen hier kurz angerissen werden (siehe auch [Milke u. a., 2006b]):

- AstroGrid-D[9] setzt zur Autorisierung des Zugangs zu den ihren Ressourcen den Registrierungsdienst VOMRS und X.509 Zertifikate ein. Um der VO beizutreten, ist eine Anmeldung über VOMRS mit entsprechender Bestätigung durch das VO-Management erforderlich. Nur Benutzer, die im VOMRS registriert sind, erhalten auch eine Zugriffserlaubnis auf die vorhandenen Grid-Ressourcen, unter Umständen eingeschränkt. In Planung sind weitere Community-VOs sowie Mechanismen zur Unterstützung von Benutzern aus anderen Communities wie C3 und HEP. Dadurch treten Anforderungen an ein VO-übergreifendes Management in den Vordergrund.

[7] Basisinfrastruktur der D-Grid-Initiative
[8] `http://public.eu-egee.org/`
[9] `http://www.gac-grid.de/`

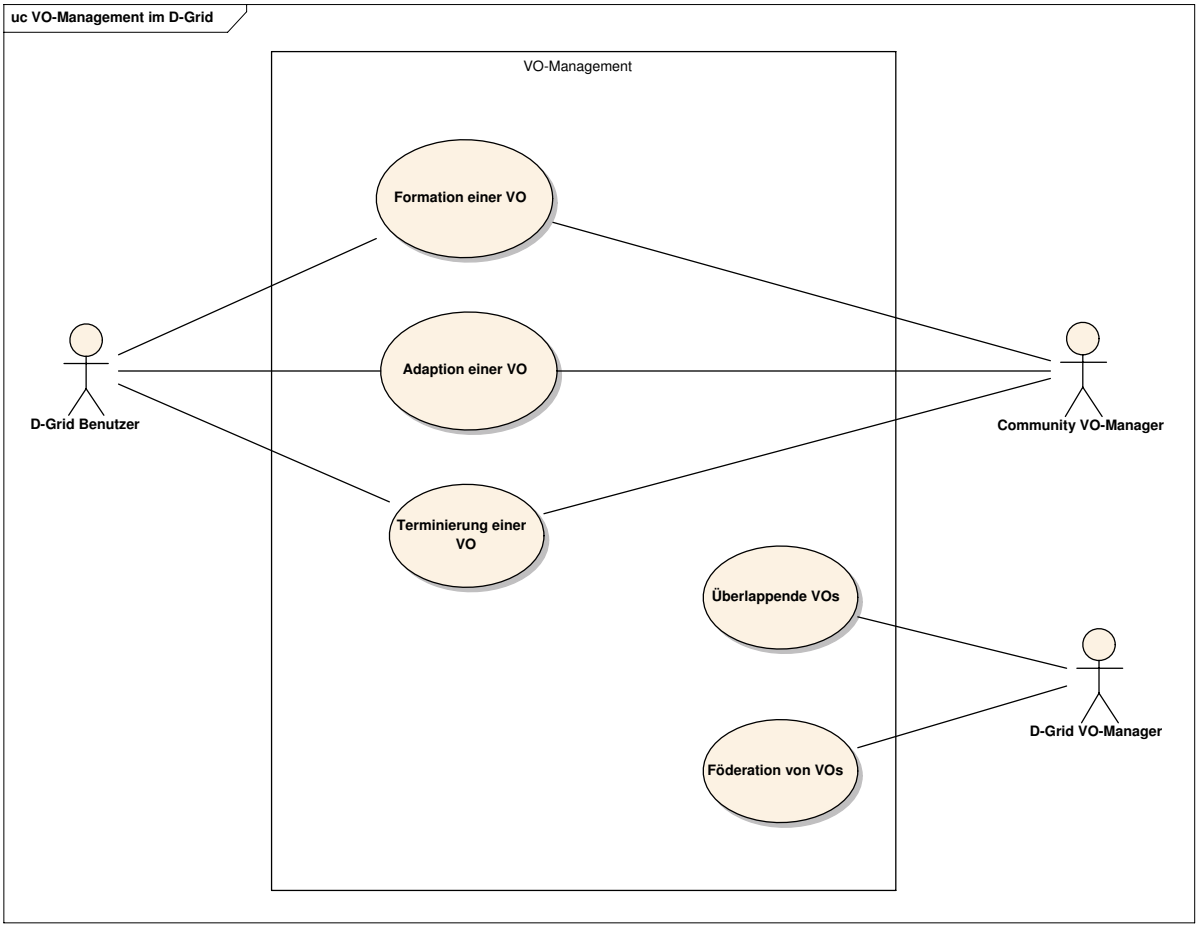


Abbildung 3.8: Rollen und Anwendungsfälle des VO-Managements im D-Grid nach [Milke u. a., 2006b]

- Im HEP-Grid[10] entsprechen die VOs den großen teilchenphysikalischen Experimenten bzw. Teilchenbeschleunigern. VOs in HEP verlangen zur Authentifizierung X.509-Zertifikate, für die Autorisierung von Mitgliedern wird statt VOMRS VOMS eingesetzt.

- Im MediGRID[11] wird – wie im DEISA-Fall – zunächst nur *eine* VO angestrebt. Mittelfristig ist die Aufteilung der verschiedenen Arbeitsgebiete in mehrere VOs, den Anwendungsprojekten, geplant. Die Authentifizierung erfolgt derzeit durch D-Grid-Zertifikate des DFN und der GridKa Certification Authorities (siehe auch Fußnote 5 auf Seite 81), eine reine Passwort-basierte Authentifizierung à la Shibboleth wird in MediGRID nicht als ausreichend betrachtet. Stattdessen wird für die Benutzerverwaltung eine zentrale Benutzerliste geführt, in der Zertifikaten bzw. Distinguished Names

[10] `https://www.d-grid.de/index.php?id=44`
[11] `http://www.medigrid.de/`

(DN) Rollen zugeordnet werden. Die Einträge und Rollenverteilungen werden von so genannten *Modulleitern* in Zusammenarbeit mit einem Security Board gepflegt. Diese Liste wird den Ressourcenanbietern zur Umsetzung zur Verfügung gestellt.

- Im InGrid[12] stehen projektspezifische, dynamische VOs im Vordergrund:

  Beispiel: Ein Ingenieurbüro bekommt den Auftrag, eine Gussform zu optimieren. Dazu wird vom Projektleiter des Ingenieurbüros eine VO aufsetzt. Er benötigt dazu Ressourcen in Form von Hardware und Software bzw. Lizenzen. Es gibt damit unter den Service-Providern die Rollen eines Hardware- Resource-Providers und die eines Software-Resource-Providers. Daneben gibt es typischerweise vier Rollen von Usern: derjenige, der die Simulation durchführt und Zugriff auf Input- wie Output-Daten sowie die Software zur Optimierung und die Software zur Gießsimulation benötigt; die Rolle des Software-Providers und Experten für die Gießsimulation, der keinen Zugriff auf die Optimierungssoftware hat und beispielsweise auch nur die für ihn relevanten Daten übermittelt bekommen darf; die Rolle des Software-Providers und Experten für die Optimierung, der keine Lizenz für die Gießsimulation hat; und der Auftraggeber des Ingenieurbüros, der lediglich die Ergebnisse analysieren will, d.h. Zugriff auf Input- und Output-Daten sowie Visualisierungssoftware benötigt. Unter Umständen wird noch ein Service-Provider für Materialdatenbanken benötigt, sowie ein Storage-Provider für die Langzeitspeicherung der Daten.

  Darüber hinausgehend werden Informationen benötigt, die für ein Abrechnungsmanagement notwendig sind: bei Verwendung lizenzpflichtiger Software können die Kosten für akademische oder kommerzielle Nutzung unterschiedlich sein. Dies ist zwar üblicherweise für das gesamte Projekt einheitlich, teilweise kann aber auch der größte Teil des Projektes kommerziell orientiert sein und nur ein Teilbereich akademisch interessant sein. Dieser wäre dann möglicherweise anders zu bepreisen bzw. die Nutzung dieses Teils der Daten und/oder Software mag nur für akademische Gruppen erlaubt sein.

VO-Management in der Formationsphase

Eine VO ist im D-Grid zur Zeit immer genau einer Community oder dem Kern-D-Grid zugeordnet. Der Sprecher einer neu zu gründenden VO muss sich deshalb zunächst einer der bestehenden Communities anschließen und der jeweilige Sprecher der Community muss der Gründung dieser VO zustimmen. Neben dieser generellen Verabredung werden in der Formationsphase Virtueller Organisationen – abhängig von der Community – in der Regel die verwendbaren Ressourcen sowie die Rechte und Pflichten der VO-Mitglieder festgelegt.

[12] `https://www.d-grid.de/index.php?id=43`

Dazu werden in der Regel die Rollen der Verantwortlichen für sämtliche organisatorischen Fragen des VO-Managements festgelegt und die zu verwendenden Werkzeuge (z.B. VOMS, VOMRS zur Mitgliederverwaltung) eingerichtet. Dies betrifft auch die Bereitstellung einer Kollaborationsplattform für die VO in Form von E-Mail-Verteilern, Web-Portalen, Wikis oder gemeinsam nutzbaren Dateiablagebereichen.

Beispiel: In der *HEP-Community* wird eine neue VO über ein entsprechendes Web-Portal registriert[13]. Für die Registrierung der VO sind dann die folgenden Angaben notwendig: der Name der VO, die *usage policy*, die Kontaktinformationen für den VO-Manager und die Sicherheits-Verantwortlichen, eine Beschreibung der VO (in Prosa) sowie die erwartete Mitgliederzahl und der erwartete Ressourcenbedarf. Die Registrierung muss anschließend von der so genannten *EGEE/LCG Operations Advisory Group* bestätigt werden.

VO-Management in der Betriebs- und Anpassungsphase

Anders als im DEISA-Szenario ist die Betriebs- und Anpassungsphase im D-Grid durch eine relativ hohe Dynamik gekennzeichnet, sowohl in den Mitgliedsstrukturen innerhalb einer VO als auch in den VO-Strukturen selbst. Folglich wird der wesentliche Aufwand des VO-Managements während dieser Phase dem breiten Spektrum der Mitgliederverwaltung zuzuordnen sein. Abhängig von der jeweiligen Community umfasst die Mitgliederverwaltung die Aufnahme neuer Mitglieder, die Zuteilung von Rechten, die Durchführung von Authentifizierungen, die Änderung von Benutzerrechten (z.B. für spezielle Aufgaben), die Änderung der Zugehörigkeit zu bestimmten Gruppen innerhalb einer VO oder die Sperrung von Mitgliedschaften im Rahmen regulärer Terminierungen (z.B. Vertragsbeendigungen) oder im Rahmen der Ahndung missbräuchlicher Ressourcennutzungen.

Während der Laufzeit einer D-Grid-VO fallen umfangreiche Protokoll-Daten an, die – Community-abhängig – für Auditierungen oder *post mortem*-Analysen unter Berücksichtigung der gesetzlichen Rahmenbedingungen archiviert werden.

Beispiel: Die Adaption einer VO im *Kern-D-Grid* ist derzeit beschränkt auf die Aufnahme neuer Mitglieder bzw. das Löschen existierender Mitglieder. Um Mitglied in einer der VOs des Kern-D-Grids zu werden, muss sich der Benutzer zunächst mit dem VO-Sprecher in Verbindung setzen. Nach erfolgter Absprache müssen zur Administration und für den Zugang Daten des Nutzers erfasst werden. Als Voraussetzung benötigt der Antragsteller ein gültiges Grid-Zertifikat, das in den von ihm verwendeten Browser importiert worden sein muss. Das elektronische Formular für die Mitgliedschaft in einer bestehenden VO ist zur Zeit unter `https://www.fz-juelich.de:8814/DGRID_DISPATCH` zu erreichen.

---

[13] `https://cic.in2p3.fr/index.php?section=vo&page=newvoregistration`

VO-Management in der Auflösungsphase

In praxi sind D-Grid-VOs bisher nicht aufgelöst worden, theoretisch wird dies allerdings durch den in der jeweiligen Community benannten Verantwortlichen (dem VO-Manager in [Milke u. a., 2006b]) in Absprache mit dem Sprecher der Community geschehen. Der VO-Manager wird im Rahmen der VO-Auflösung die Mitglieder der VO rechtzeitig über das bevorstehende Ende der VO informieren und mit den Mitgliedern der VO die Notwendigkeit, vorhandene Daten der VO (also z.B. Messergebnisse, Ergebnisse von Simulationsrechnungen) über das Ende der VO hinaus zu speichern, klären und diese gegebenenfalls für andere VOs derselben Community verfügbar zu machen (*garbage collection*). In der Regel werden derartige Policies schon bei der Gründung der VO festgelegt, der Lebenszyklus der VO kann jedoch durchaus zu neuen Anforderungen führen. Insbesondere wird in der Auflösungsphase festgelegt, welche Meta-Informationen über die VO längerfristig zur Verfügung stehen müssen (z.B. An- und Abmeldungen von Benutzern bei der VO, Änderungen von Benutzerrechten, Abrechnungen oder Statistiken der Ressourcennutzung). Während der Auflösungsphase werden bestehende – für die Lebensdauer der VO gültige – Verträge und SLAs mit den Ressourcen Providern analysiert und gegebenenfalls gekündigt.

Szenario III: Ereignisgesteuerte VOs in EmerGrid

Ganz andere VO-Managementfragen treten in den Vordergrund, wenn die im DEISA-Szenario zugrunde gelegte geplante VO-Gründung durch eine ereignisgesteuerte, spontane Formation ersetzt wird und die VOs mit Prioritäten für den Zugang zu Ressourcen ausgestattet sind. Dieser Aspekt wird im folgenden EmerGrid-Szenario betrachtet, das wie in Abbildung 3.9 positioniert wird.

Die Motivation für EmerGrid liegt in Krisenmanagement-Szenarios, in denen diverse Krisen- und Katastrophensituationen durch eine Vielzahl involvierter Kräfte gemeistert werden müssen, meistens unter der Ägide eines mit entsprechenden Vollmachten ausgestatteten Krisenstabes (siehe auch [Jungert u. a., 2006] und [Bundesamt für Bevölkerungsschutz und Katastrophenhilfe, 2007]). Beispiele finden sich im FireGrid-Szenario [Berry u. a., 2005], dem Next Generation Grid Disaster-Szenario [NGG2 Expert Group, 2004], dem EU-geförderten MEDIGrid [Eftichidis, 2005][14] und in Teilen in dem in [Bourbonnais u. a., 2004] beschriebenen Health-Grid-Szenario.

Kurzbeschreibung des Szenarios

Basierend auf der Motivation reichen in EmerGrid typische Anwendungsfälle von Grid-Technologien von der Echtzeitintegration komplexer Flutmodelle zur Überschwemmungsprognose, über Schockwellensimulationen bei Explosionen in Tunneln bis zu Entscheidungsunterstützungssystemen bei großflächigen Evakuierungen. Im eingetretenen Krisenfall (oder definierten Krisensimulationen in Trainingsmanövern) bilden die beteiligten Rettungskräfte eine EmerGrid-VO, entweder angestoßen durch dazu autorisier-

[14] nicht zu verwechseln mit der D-Grid-Community MediGRID des vorhergehenden Szenarios

te Institutionen (Krisenstäbe, Organisationen oder Individuen) oder – falls entsprechende Frühwarnsysteme vorgesehen sind – automatisch.

In beiden Fällen ist eine schnelle Reservierung von spezialisierten Ressourcen und Diensten unter harten Dienstgüteanforderungen erforderlich, gekoppelt mit einem hochprioren Zugriff darauf. Beispiele solcher Ressourcen sind HPC-Systeme, Speicherkapazitäten, Netzbandbreiten, Spezialinstrumente, aber auch Lizenzen von Simulationspaketen.

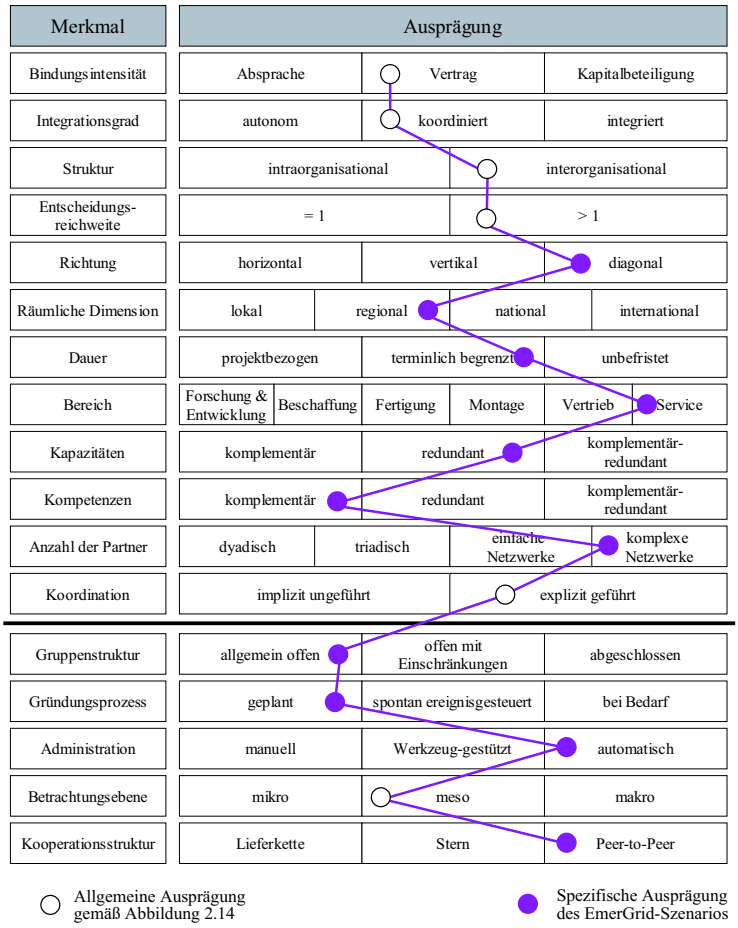


Abbildung 3.9: Positionierung des EmerGrid-Szenarios im Raster von Abbildung 2.10

Neben den nicht unerheblichen Problemen eines integrierten Informations- und Wissensmanagements unter Berücksichtigung von Semantiken und eines schnellen Case Based Reasonings (CBR) [AKT e-Response Team, 2007], stellt das adäquate Management Virtueller Organisationen in EmerGrid eine nicht-triviale Herausforderung dar [Vaccari u. a., 2006]. Die Ziele des VO-Managements liegen in einer schnellen, aufgabenorientierten Formation Virtueller Organisationen, deren zweckgebundenem Betrieb und deren Auflösung nach Erfüllung der Aufgaben. Eine EmerGrid-VO umfasst damit alle zur Erledigung der Aufgabe notwendigen Individuen, Regierungsstellen, nicht-behördliche Organisationen, automatisierte Systeme, Grid- und Web Services, intelligente Roboter und Vehikel [Micacchi u. Cohen, 2006] sowie Gebäude-, Umwelt- und Planungssystem [BMI, 2005; Broy u. a., 2002; Lillesand u. a., 2003], die die VO in die Lage versetzen, auf dynamische Ereignisse angemessen reagieren zu können. [Allsopp u. a., 2002] beschreibt ein zwar Grid-unabhängiges, aber dennoch typisches Szenario zur Kooperation in internationalen Koalitionen.

VO-Management in EmerGrid

Prinzipiell deckt EmerGrid zwei Operationsmuster ab: Im *geschlossenen* Ansatz sind die Teilnehmer, ihre Rollen und die verwendeten Dienste und Ressourcen (meistens gesetzlich) festgelegt, im *offenen* Ansatz ist a priori nicht bekannt, wie die VO auszustatten ist. In-

sofern wird es im zweiten Fall nicht ungewöhnlich sein, wenn mehrere Resource Provider und Service Provider die geforderten Ressourcen und Dienste bereitstellen können, einige konditioniert, einige ohne Einschränkungen, andere – staatliche Institutionen beispielsweise – können de jure dazu verpflichtet sein. Das Finden der „richtigen“ Dienste und der „richtigen“ Provider, das Zusammenstellen eines situativ optimalen Dienst- und Dienstleisterpaketes und die Überwachung der Adäquatheit dieses Paketes bilden jedoch in beiden Ansätzen wesentliche Managementherausforderungen. So wie die Güte von Diensten durch kasuistisch relevante Metriken permanent beurteilt wird, trifft dies auch auf die Provider zu. Für den Fall, dass sich die Güte von Diensten oder die Reputation von Providern negativ verändern sollte, kann in EmerGrid optional eine redundante VO gebildet werden mit entsprechenden *Hot Failover*-Mechanismen. Alternativ kann eine Re-Konfigurationsstrategie vorgesehen werden, falls Dienste oder Provider auszutauschen sind oder neue Anforderungen berücksichtigt werden müssen [gentschen Felde u. a., 2006]. Insofern verlangt das Szenario neben der permanenten Beurteilung der Zielerreichung auch die ständige Einschätzung der mittelbar und unmittelbar beteiligten Provider.

Organisatorischer Rahmen des VO-Managements

Der organisatorische Rahmen des VO-Managements ist in EmerGrid durch diverse gesetzliche und branchenspezifische (siehe beispielsweise [Bundesverband deutscher Banken, 2004] für den Bankenbereich) Bestimmungen vorgegeben[15]. Entscheidend für die Wirksamkeit der mit EmerGrid unterstützten Maßnahmen ist, dass die Bewältigung eines eingetretenen Krisen- oder Katastrophenfalles eine schnelle Entscheidung der autorisierten Stellen (behördlicher und nicht-behördlicher Art) zur Gründung und Initialisierung eines Krisenstabes erfordert, und dass die getroffenen Maßnahmen allen Betroffenen zeitkritisch mitgeteilt werden können und auch werden. Der Bildung des Krisenstabes entspricht in EmerGrid die Formation einer EmerGrid-*Kern-VO* mit dem Ziel eines situativ geeigneten Krisenmanagements. Zur Bewältigung spezifischer Teilziele kann die Kern-VO die Bildung weiterer – nicht notwendigerweise überlappender – *Task-VO*s initiieren, was zu einer hierarchisch angelegten VO-Struktur führt und hier in Analogie zur Organisationstheorie als „linien-orientierte“ VO-Kette bezeichnet wird.

Die Gründung und der Betrieb beider Kategorien von VOs erfordert nominell die explizite Kenntnis sowohl der notwendigen Kooperationspartner als auch der Bereitstellungsmechanismen deren Ressourcen und Dienste.

VO-Management in der Formationsphase

Die Formation einer EmerGrid-Kern-VO (und damit eines Krisenstabes) setzt die Verfügbarkeit eines umfassenden regionalen, nationalen oder internationalen Grundsatzprogrammes („Krisenmanagement-Policy“) von Regierungsstellen, Behörden, nichtbehördlichen Organisationen und involvierten Unternehmen voraus. Es beschreibt den Handlungsrahmen für Krisenfälle und legt die grundsätzlichen Strategien für den Umgang

[15]Ein Überblick über die aktuelle Gesetzeslage der Bundesrepublik Deutschland ist in [Bundesamt für Bevölkerungsschutz und Katastrophenhilfe, 2007] zu finden.

mit Krisen im Rahmen eines „Master-Plans“ fest. Vom Master-Plan werden detaillierte und abgestufte Krisenpläne mit konkreten Handlungsanweisungen instanziiert. Dies betrifft insbesondere die Spezifikation von Rollen, deren Verantwortlichkeiten und Rechte sowie die Festlegung der internen und externen Kommunikationsmechanismen. Außerdem werden im Rahmen der Initialisierung der VO die „Rufbäume“ (*call trees*) und Eskalationsprozeduren definiert.

Für das VO-Management bedeutet dies, dass dabei nicht nur die internen Berichts- und Entscheidungswege (realer) Organisationen Berücksichtigung finden müssen. Vielmehr muss eine nicht unerhebliche Menge externer Imponderabilien einbezogen werden, da im Krisenfall davon ausgegangen werden muss, dass kritische Infrastrukturbereiche nicht (mehr) verlässlich zur Verfügung stehen, partiell oder komplett, und dass kritische Ressourcen exklusiv zugeteilt werden – bestehende SLAs außer Kraft setzend.

VO-Management in der Betriebs- und Anpassungsphase

Die Formation von EmerGrid-Task-VOs geschieht nach ähnlichen Mustern, wie sie schon in den beiden vorhergehenden Szenarios diskutiert wurden. Im Unterschied zu DEISA und D-Grid ist in EmerGrid allerdings eine strikte Zielhierarchie zu erkennen, die eine ständige Überprüfung der „Adäquatheit“ der Task-VOs während der Betriebsphase impliziert und im Bedarfsfall zur Re-Konfiguration des VO-Gefüges durch die Kern-VO führt. Letzterer fällt im Rahmen der Krisenbewältigung die Rolle eines Command and Control (C2)-Zentrums [Jungert u. a., 2006] zu.

Damit unterstützt das VO-Management in dieser Phase die Zulassung und Autorisierung neuer Mitglieder und Organisationen, das Löschen oder Sperren von Mitgliedern und Organisationen, das Ändern von Attributen, den Betrieb von Informationsdiensten [Yee, 2001], die Festlegung und Belegung von Rollen und die Definition VO-weiter Policies. Neben diesen klassischen Aufgaben, müssen vom VO-Management allerdings auch – anders als in den Szenarios vorher – Mechanismen bereitgestellt werden, um Ressourcen prioritätsgesteuert zu allozieren.

VO-Management in der Auflösungsphase

In EmerGrid ist die Lebensdauer einer VO eng an Bestand der Krisensituation gekoppelt. Mit der Auflösung des Krisenstabes enden auch die Kern- und Task-VOs. Wie üblich werden auch in EmerGrid die VO-Strukturen aufgelöst und kritische Daten für spätere Auditierungen unter Berücksichtigung gesetzlicher Bestimmungen (z.B. Datenschutz und Verfassungsschutz) archiviert. Zu Trainingszwecken können zusätzlich sämtliche während der Lebenszyklen der VOs gesammelten Daten anonymisiert werden und als *crisis simulation case* hinterlegt werden. Die Auflösung der Kern-VO impliziert auch stets die Abrechnung angefallener Kosten nach vordefinierten Schlüsseln.

Szenario IV: VO-Kooperationen im Internationalen Polarjahr

Mit dem International Polar Year (IPY)-Szenario wird nun eine weitere Ebene des VO-Managements betrachtet, nämlich die in Abbildung 2.5 erwähnte Makroebene, auf der Aspekte der Kooperation Virtueller Organisationen aus der Managementsicht behandelt werden.

Kurzbeschreibung des Szenarios
Das Internationale Polarjahr (IPY)[16] stellt ein umfangreiches Wissenschaftsprogramm mit dem Fokus auf die arktischen und antarktischen Regionen dar. IPY wird organisiert durch das International Council for Science (ICSU) und die World Meteorological Organization (WMO). Es ist nach den Polarjahren 1882, 1932 und 1957 das vierte Wissenschaftsprojekt dieser Art mit einer Laufzeit vom März 2007 bis zum März 2009.

Um die arktischen und antarktischen Regionen gleichmäßig zu untersuchen, überdeckt IPY mit mehr als 200 Projekten (siehe Abbildung 3.10, in der jede Wabe ein solches Projekt symbolisiert) und etwa 50.000 Wissenschaftlern aus über 60 Ländern zwei komplette Jahreszyklen. Im Vordergrund stehen zwar primär Fragestellungen der Physik und Biologie, aber auch sozialwissenschaftliche Themenstellungen, wie der Einfluß des Tourismus auf die arktischen Regionen, werden betrachtet.

IPY ist ausgesprochen datenzentriert und basiert auf einer Datenverwaltungs-Policy, in der sämtliche Daten (inklusive aller Betriebsdaten), schnell, offen und komplett zur Verfügung gestellt werden [Allison u. a., 2007]. Ausnahmen bilden lediglich vertrauliche und mit Schutzrechten abgesicherte Daten. Insofern stellt ein adäquates Daten- und Informationsmanagement eine der größten Herausforderungen im IPY dar [Rapley u. a., 2004].

VO-Management im IPY
IPY stellt kein eigentliches Grid-Szenario dar, da in keinem der Projekte Grid-Infrastrukturen verwendet werden oder geplant sind. Dennoch treten in IPY VO-Strukturen in der Form eines losen Gefüges institutionsübergreifender und internationaler wissenschaftlicher Projekte, die sich durch unterschiedliche Größen und Laufzeiten auszeichnen und in der Regel durch Selbstorganisation entstanden sind. So umfasst beispielsweise die Aktivität „*Plate Tectonics and Polar Gateways in Earth History*“ (das in Abbildung 3.10 markierte Projekt 77 [Makedanz u. Pfeiffenberger, 2006]) mehr als 80 Personen aus 14 Ländern, die in etwa 50 Institutionen beheimatet sind. Gemeinsam ist allen Projekten, dass sie eine Führungsstruktur besitzen und über selbst*verwaltete* Ressourcen (Daten, Rechner, Mailinglisten, Webseiten) verfügen und – anders als in den meisten D-Grid-VOs – die Nutzungsrechte für diese Ressourcen selbständig vergeben können. IPY-Projekte bilden damit VOs im weiteren Sinn, deren Management sich primär auf die Kooperation mit anderen IPY-Projekten und die Autorisierung von Ressourcen-Zugängen bezieht. Unter „Kooperation“ ist in IPY nicht nur die gemeinsame Nutzung von Ressourcen und Daten gemeint (im Sinne einer Prozesssicht), sondern insbesondere auch die mitgliedermäßige

[16] `http://www.ipy.org`

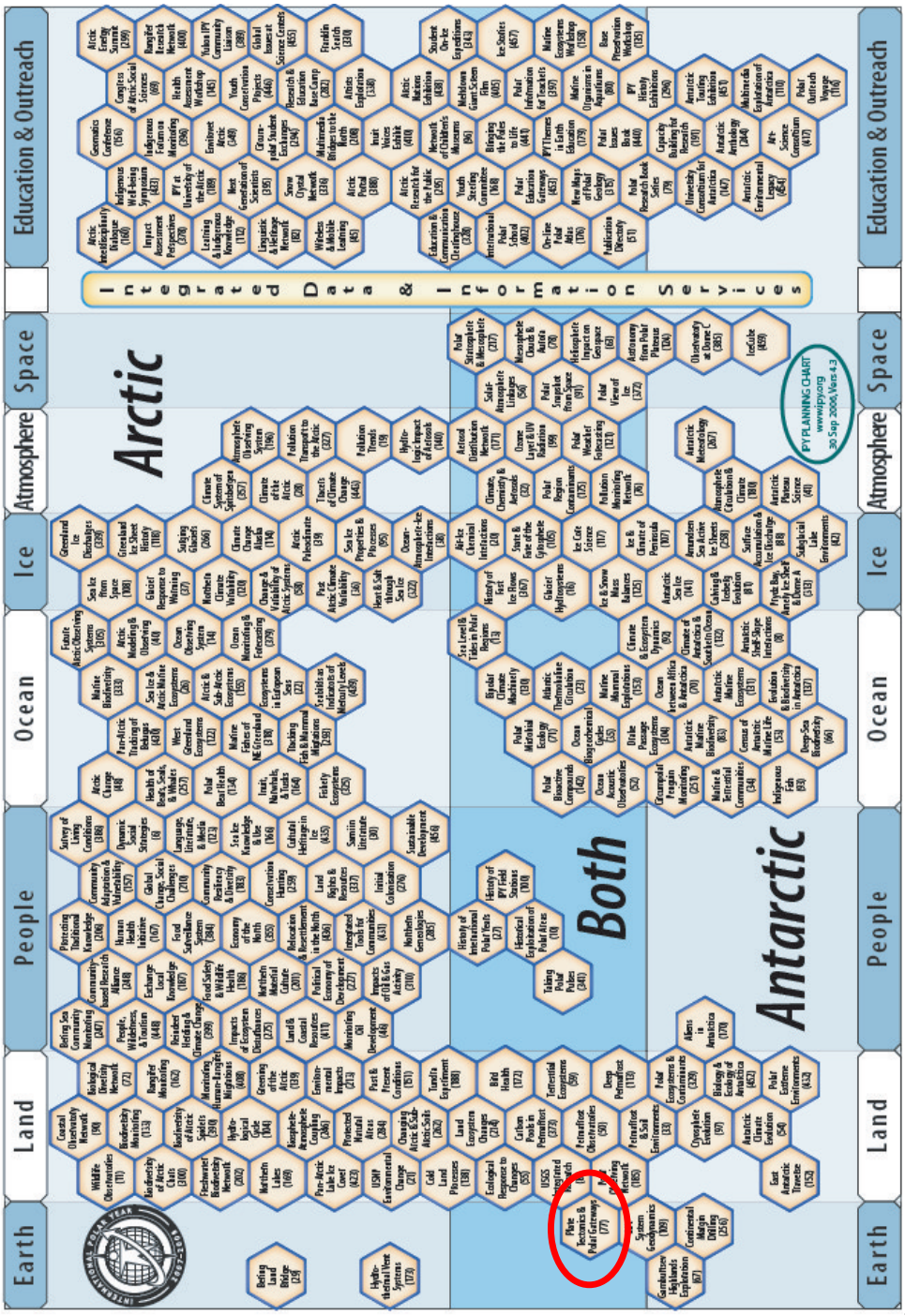


Abbildung 3.10: Projekte im International Polar Year (Version 4.3 vom 30. September 2006) nach `http://www.ipy.org`

Überlappung von VOs (im Sinne einer Struktursicht). Personen können in IPY mehreren Institutionen (also realen Organisationen) und mehreren VOs angehören. „Zugehörigkeit“ wird damit zur Relation zwischen Personen und Organisationen. Typische Attribute einer Zugehörigkeitsrelation sind Rollen, Rechte und Mitgliedsdauer.

Eine grundlegende Anforderung im IPY besteht in der möglichst einfachen Autorisierung der beteiligten Personen für einen Ressourcenzugriff unter Vernachlässigung der Benutzeridentitäten. Dazu wird IPY-weit auf eine Attribut-basierte Zugriffskontrolle (ABAC) [Grimm u. Pattloch, 2006] gesetzt. Die Schwierigkeit liegt jedoch darin, dass bestehende Ansätze wie Shibboleth [Watt u. a., 2006] zwar ideal für *Institutionen* sind, für VO-Gefüge im IPY sind sie jedoch unzureichend, da jede VO selbst als Quelle von Autorisierungen dienen kann, VOs aber im Rahmen eines föderierten Identity Managements selbst nicht als Identity Provider vorgesehen sind. Im Rahmen des D-Grid IVOM-Projektes[17] wird diese Fragestellung adressiert.

3.1.2 Abstraktes Szenario zum VO-Management

Im Kapitel 2 wurden Virtuelle Organisationen, Dienste und Managementkonzepte sowohl allgemein als auch im Grid-spezifischen Umfeld beschrieben, Abschnitt 3.1.1 betrachtete zusätzlich einige *current practices* im Grid-VO-Management. Aus der Kombination beider Abschnitte lässt sich ein abstraktes VO-Managementszenario ableiten. Dieses ist in Abbildung 3.11 schematisch dargestellt[18].

Kurzbeschreibung des Szenarios

Die Darstellung der Ausgangssituation hat die Sinnhaftigkeit des in [Camarinha-Matos u. a., 2005] diskutierten *preparedness*-Konzeptes in Form eines *breeding environments* noch einmal verdeutlicht. Ein solches Konzept wird in jedem Grid-Projekt mehr oder weniger aufwändig umgesetzt, im D-Grid beispielsweise in den diversen Community-Ansätzen, in DEISA durch das Konsortialgefüge. Werden VOs gebildet, werden konstituierende Organisationen aus der *Community* von dem VO-Initiator zur Teilnahme an der VO eingeladen. Die Teilnahme realer Organisationen beschränkt sich auf die Bereitstellung von Kontingenten lokaler Ressourcen, Dienste und Personen unter Berücksichtigung bi- und multilateral vereinbarter Rahmenbedingungen in Form von Policies und SLAs. Der Auslöser zur Gründung Virtueller Organisationen liegt entweder in der Community selbst oder – wie beispielsweise im EmerGrid – außerhalb der Community. VOs sind durch eine starke Dynamik geprägt: Mitglieder (siehe Definition 2 auf Seite 40) und Ressourcen werden hinzugefügt oder entfernt, ihre Rechte werden modifiziert, oder ihre Kooperationsstruktur wird – abhängig vom aktuellen Kontext – rekonfiguriert.

[17] `https://www.d-grid.de/index.php?id=314&L=0`

[18] Heterogene Ressourcen virtueller oder realer Art werden typischerweise gekapselt, damit ein Zugriff über standardisierte Schnittstellen erfolgen kann. Diese wird in Abbildung 3.11 der Übersichtlichkeit halber allerdings nicht explizit angezeigt.

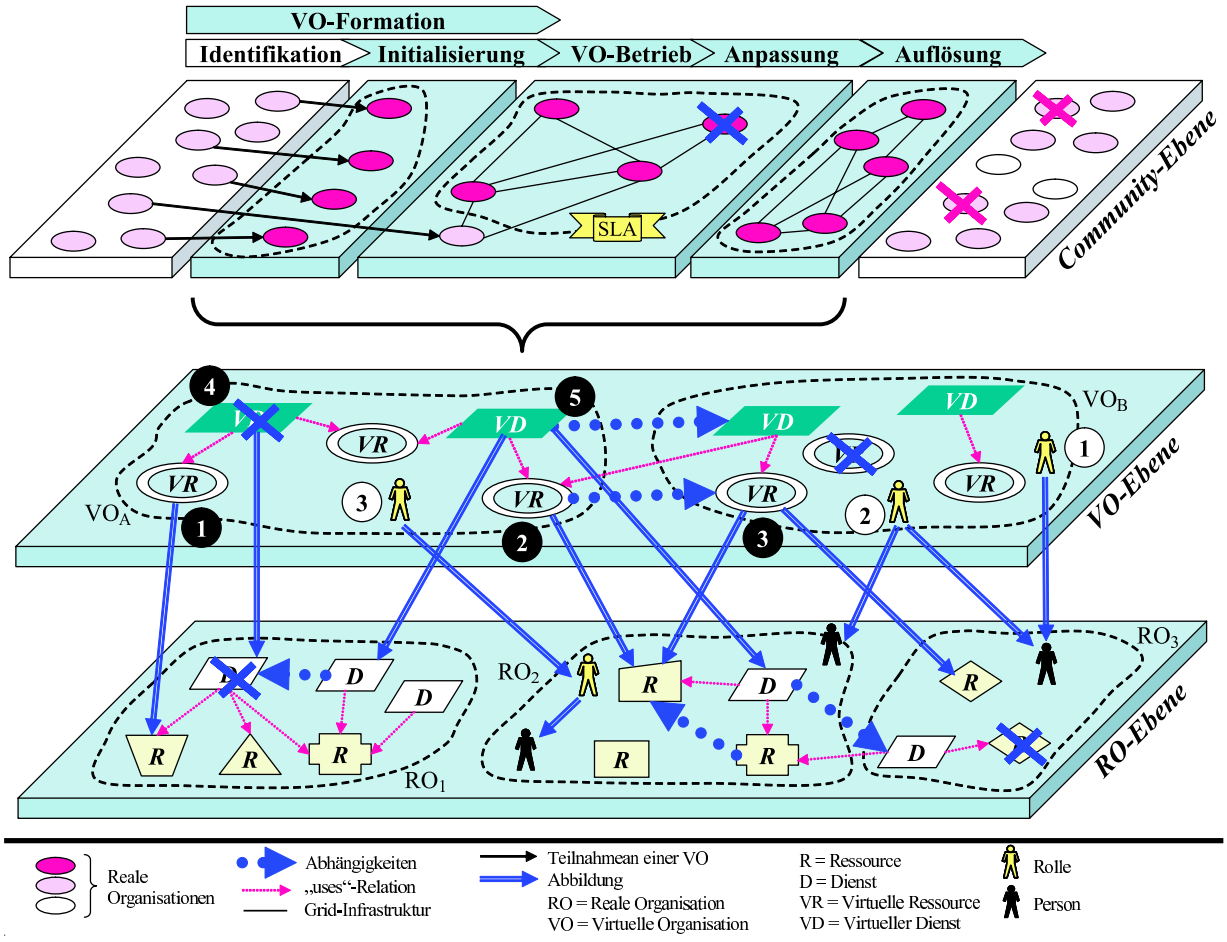


Abbildung 3.11: Allgemeines VO-Managementszenario

Konzeptionell stellen VOs virtuelle Dienste bereit, die auf virtuellen Ressourcen erbracht werden. Als *managed object* repräsentieren VOs Organisationen, deren Lebenszyklus mit prinzipiell den gleichen Managementmechanismen behandelt wird wie der klassischer Ressourcen und Dienste [Dreo Rodošek, 2002].

Virtuelle Ressourcen einer VO müssen nicht notwendigerweise stets auf genau eine reale Ressource abgebildet werden (wie die virtuelle Ressource ❶ in Abbildung 3.11), vielmehr wird in der Regel eine virtuelle Ressource mehrere reale Ressourcen „überlagern", die zudem aus mehreren Organisationen stammen (wie die virtuelle Ressource ❸ in Abbildung 3.11). Ein Beispiel stellt eine im IPY-Szenario verwendete virtuelle Speicherressource dar, die aus mehreren Speicherelementen der teilnehmenden realen Organisationen bestückt wird (z.B. RAID-Systeme der einen Organisation, Storage Area Networks (SAN) einer anderen Organisation und Bandsysteme einer dritten Organisation). Es sei angemerkt, dass virtuelle Ressourcen weitere virtuelle Ressourcen nutzen können, die zudem von anderen VOs bereitgestellt werden können (virtuelle Ressource ❷ in Abbildung 3.11). Erfordert beispielsweise die Ressource ❷ die Bereitstellung einer virtuellen HPC-Ressource, die von

VO_B angeboten wird und selbst wieder auf einem Cluster in RO_3 basiert, so stellt dies genau diesen Fall dar.

VOs treten jedoch nicht nur als Provider virtueller *Ressourcen* auf, sondern auch als Provider virtueller *Dienste*, die auf einen oder mehrere Dienste realer Organisationen abgebildet werden (Markierungen ❹ und ❺ in Abbildung 3.11). Virtuelle Dienste können in vertikalen und horizontalen Dienstketten arrangiert werden (wie z.B. der virtuelle Dienst ❺). Typische Beispiele virtueller Dienste stellen die Informationsdienste im D-Grid – mit denen Grid Operations Centers den aktuellen Status von VO-Topologien überwachen – und Experimentvisualisierungen im IPY dar. Weitere Beispiele sind in [Göhner u. a., 2006] und [Xu u. a., 2004] zu finden.

Die Analyse der Szenarios zeigt, dass VOs rollenzentriert denken, im Gegensatz zu realen Organisationen, in denen Positionen und Rollen assoziiert sind (siehe Abbildung 2.4). Rollen sind im Gegensatz zu Positionen aufgabenorientiert und daher mit aufgabenorientierten Rechten ausgestattet. Rollen werden auf Personen abgebildet, die (mindestens) einer der die VO konstituierenden realen Organisationen angehören und dort über Benutzerkennungen, Zertifikate oder andere Verfahren authentifiziert werden. Erst durch diese Abbildung werden Personen zu VO-Mitgliedern. Insofern denken VOs nicht in realen Personen, sondern betrachten allenfalls deren abstrakte Repräsentation im Rahmen der Abbildung (siehe Markierung ① in Abbildung 3.11 und Definition 2 auf Seite 40). Diese Abbildung ist allerdings weder injektiv noch surjektiv, da VO-Rollen durch mehrere Personen (durchaus unterschiedlicher realer Organisationen) wahrgenommen werden können und eine reale Person mehrere Rollen einnehmen kann (siehe Rolle ② in Abbildung 3.11). *VO-Rollen* können aber auch auf *RO-Rollen* abgebildet werden, deren Besetzung auf der VO-Ebene nicht notwendigerweise bekannt sein muss (Rolle ③ in Abbildung 3.11).

Beispiel: Beispiele der ersten Kategorie sind klassische Grid-Nutzer, die auf lokale Benutzerkennungen abgebildet werden. Im Globus Toolkit manifestiert sich diese Abbildung beispielsweise in den *gridmap*-Files [Sotomayor u. Childers, 2006]. Beispiele der zweiten Kategorie finden sich potenziell immer dann, wenn einzelne Personen Mitglieder mehrerer VOs sind und dort jeweils unterschiedliche Rollen einnehmen, oder wenn in DEISA ein Administrator des DEISA Operations Teams durch die Kooperation mehrerer Konsortialpartner realisiert wird. Ein Beispiel der dritten Kategorie tritt im D-Grid oder IPY auf, wenn die Rolle eines *Principal Investigators*, der eine VO gründet, durch die Rolle eines Projektmanagers einer realen Organisation wahrgenommen wird.

Im Zusammenhang mit diesen Ausführungen sind drei kurze Anmerkungen angebracht:

Bemerkung 1: Dienstkomposition und Dienstvirtualisierung. Obwohl häufig synonym verwendet, ist Dienstvirtualisierung nicht mit Dienstkomposition zu verwechseln: Dienstkompositionen dienen der Orchestrierung bzw. Choreographie von

Diensten, beispielsweise im Rahmen von Workflows. Dienstvirtualisierungen verwenden dagegen Aggregationsmechanismen zur Bereitstellung neuer Dienste [Tan u. a., 2004].

Bemerkung 2: Definitionsproblematik. Das hier verwendete Modell sieht keine Abbildungen von virtuellen Ressourcen auf reale Dienste und von virtuellen Diensten auf reale Ressourcen vor, da diese konzeptionell unsauber wären. Obwohl sich durchaus Beispiele solcher Kategorien finden lassen (für den zweiten Falles z.B. die Verwendung von Caches anstelle der Ausführung realer Dienste [Jagatheesan u. a., 2002]), können solche Konstellationen modellkonsistent dargestellt werden durch entsprechende Kapselungen.

Bemerkung 3: Rollen und Abbildungen. Nicht nur VOs denken in Rollen, auch Communities. Rollen der Community-Ebene werden allerdings nicht auf Rollen der VO-Ebene abgebildet, sondern direkt auf Personen (bzw. deren Repräsentation) der RO-Ebene. Streng genommen bilden Communities damit eigentlich wieder VOs. Diese Zusammenhänge werden in Abbildung 3.11 der Übersichtlichkeit halber nicht explizit dargestellt. Ebenso wenig wird der Unterschied zwischen realen Personen und deren abstrakter Repräsentation kenntlich gemacht, da Verwechselungen nicht zu befürchten sind.

VO-Management im abstrakten Szenario

Dem VO-Gedanken liegt im allgemeinen – wegen der *preparedness* – ein *Community*-Konzept zu Grunde, das die Basis für metaorganisatorische Ansätze, wie sie im VO-Management betrachtet werden, bildet. Mitglieder von Communities stehen sich prinzipiell nahe. Diese Nähe drückt sich zwar in der Regel in einer thematischen Nähe aus, kann sich aber auch, wie das EmerGrid-Beispiel zeigt, in komplementären Fähigkeiten äußern, solange diese dem VO-Zweck dienen. VOs werden auf der Community-Ebene primär auf der Basis von *Einladungen* durch einen VO-Provider (VOP) [Hommel u. Schiffers, 2006] gebildet.

Beispiel: Beispiele für „Einladungen“ sind in DEISA die Konsortialverträge, in IPY die Projektkooperationen oder die im EmerGrid auftretenden Dienstverpflichtungen im Rahmen des Katastrophenschutzes. Beispiele für „VO-Provider“ sind die PIs in DEISA, die VO-Manager in einigen D-Grid-Communities oder die Krisenstäbe für EmerGrid-Task-VOs.

Nach der Einladung zur Gründung einer VO durch den VO-Provider wird, wie die Szenarios zeigen, die VO initialisiert. Erst jetzt erhält das „lockere“ Konsortium eine zweckorientierte Struktur, werden Policies eingerichtet, SLAs verhandelt und Workflows definiert.

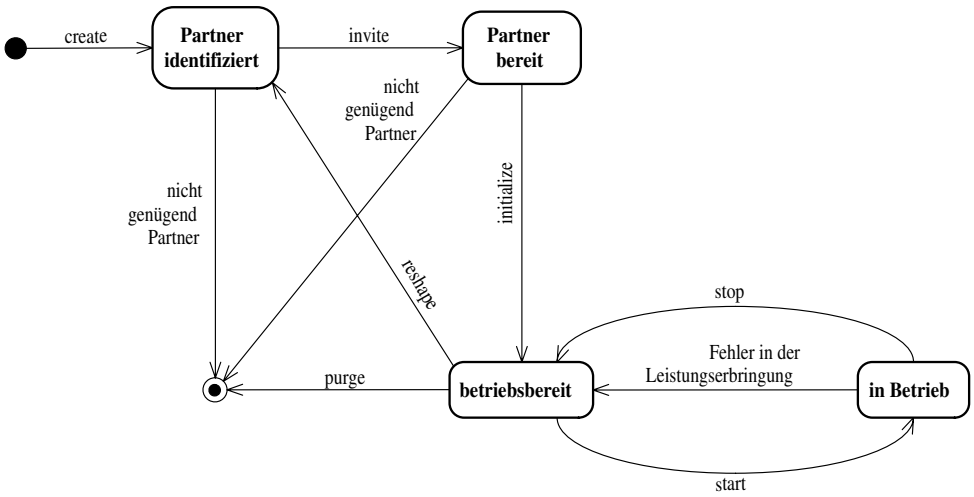


Abbildung 3.12: VO-Zustandsmodell

Kurz: Die VO-Ebene wird eingerichtet. Gleichzeitig wird die Rolle des VO-Managers institutionalisiert.

Nach der Initialisierung ist die VO bereit für den operativen Betrieb und betriebsbedingte Anpassungen. Diese werden immer dann notwendig, wenn Personen, Ressourcen, Dienste oder komplette Organisationen ausfallen, hinzugenommen oder Rechte verändert werden müssen. Als Quelle neuer Entitäten wird in der Regel die Community dienen, die DEISA- und EmerGrid-Szenarios haben aber gezeigt, dass im Bedarfsfall auch Community-fremde Organisationen integriert werden müssen. Ist das Ziel der VO erreicht oder ist sie nicht mehr betriebsfähig, wird sie durch den VO-Manager aufgelöst.

VOs befinden sich damit zu jedem Zeitpunkt in einer eindeutig definierten Lebenszyklusphase. Dies führt zu dem in Abbildung 3.12 dargestellten einfachen Zustandsmodell.

Anforderungen an das VO-Management in Grids

Die konkreten Ausprägungen der Szenarios – dargestellt in den Abbildungen 3.1 bis 3.9 und Abbildung 3.11 – führen zu Funktionalitäten, die vom VO-Management in Grids unterstützt werden müssen. Sie werden hier noch einmal kurz zusammengefasst. Dazu sei angemerkt, dass mit der nachstehenden Reihenfolge keine Gewichtung antizipiert werden soll.

Koordination mit lokalen Managementsystemen. Jede an einer VO beteiligte reale Organisation besitzt ihre eigenen lokalen Managementsysteme, mit denen Netze, Systeme, Dienste gemanagt werden. Insbesondere auch die, die der VO in Kontingenten zur Verfügung gestellt werden. Die reale Organisation überwacht und kontrolliert die Verfügbarkeit ihrer Kontingente, deren Auslastung, Schutz, Verlässlichkeit sowie die Verwendungshäufigkeit und -dauer durch die VO-Mitglieder. Das VO-Management

unterstützt dies mit der Bereitstellung VO-spezifischer Informationen an einer definierten Schnittstelle.

Bereitstellen virtueller Ressourcen und Dienste. Jede VO besitzt die Verantwortung, ihre „Geschäftsprozesse“ so zu gestalten, dass der Zweck der VO erreicht wird. Dies impliziert für das VO-Management, die technischen und organisatorischen Voraussetzungen so zu schaffen, dass die von der VO bereitgestellten virtuellen Ressourcen und Dienste zur Unterstützung dieser Geschäftsprozesse genutzt werden können.

Überwachung der Güte virtueller Dienste. Die VO bietet ihren Mitgliedern an Dienstzugangspunkten eine Reihe von (virtuellen) Diensten an. Diese besitzen wie reale Dienste auch eine Dienstgüte. Aufgabe des VO-Managements ist es, diese Güte zu überwachen und Verfahren zur Störungsmeldung und -behebung zu etablieren.

Behandlung von Rollen. VOs sind rollenzentriert. VO-Rollen werden auf Rollen oder Repräsentationen von Nutzern der RO-Ebene abgebildet. Für das VO-Management bedeutet dies einerseits die Unterstützung des Einrichtens, Modifizierens und Löschens von Rollen, andererseits die Unterstützung der Abbildungsdefinition und möglicher Modifikationen.

Einfache Handhabbarkeit. VO-Management findet in komplexen Umgebungen statt und wird nicht notwendigerweise von Spezialisten durchgeführt. Dies impliziert die Forderung nach einfach zu handhabbaren VO-Managementprozessen, möglichst unterstützt durch einfach zu bedienende Werkzeuge.

Behandlung von Mitgliedern. VOs stellen dynamische Konstrukte dar. Dieses drückt sich dadurch aus, dass im Laufe der Lebensdauer einer VO neue Mitglieder aufgenommen werden, bestehende Mitgliedschaften aufgelöst werden oder Mitgliederattribute modifiziert werden. Für das VO-Management bedeutet dies, Rollen und Mitglieder entkoppelt zu betrachten. Neben den rollenspezifischen Mechanismen sind daher Verfahren erforderlich, die es ermöglichen, neue Mitglieder aufzunehmen, Mitgliederattribute zu modifizieren und Mitglieder zu entfernen. Insbesondere ist vom VO-Management eine Unterstützung erforderlich, die Auswirkungen von Rollenumbesetzungen zu handhaben. Für die Aufnahme neuer Mitglieder muss jedoch definiert werden, wie neue Mitglieder in die VO aufgenommen werden. Dazu gehört die Klärung, welche Voraussetzungen ein Kandidat erfüllen muss und welches Authentifizierungsverfahren verwendet werden soll. Mitgliedsinformationen müssen – je nach Policy – in geeigneter Weise den Ressourcen-Anbietern zur Verfügung gestellt werden können.

Der Status eines VO-Mitglieds kann sich in verschiedenen Aspekten ändern: Änderung von Benutzerrechten (z.B. für spezielle Aufgaben) oder Zugehörigkeit zu bestimmten Gruppen innerhalb der VO oder Sperrung der Mitgliedschaft. Eine Sperrung einer Mitgliedschaft kann entweder automatisch erfolgen (z.B. wenn bei zeitlicher Befristung der Mitgliedschaft die Frist abgelaufen ist oder wenn das Zertifikat des Benutzers ungültig wird bzw. in der Certificate Revocation List (CRL) der ausstellenden CA

aufgeführt wird) oder aktiv durch einen VO-Administrator (z.B. wenn die VO von einem Ressourcen-Betreiber über eine missbräuchliche Ressourcennutzung durch das VO-Mitglied informiert wurde). Dazu müssen Verfahren definiert sein, wie das Mitglied über die Sperrung informiert wird, welche Instanzen zusätzlich informiert werden müssen, welche Maßnahmen für ein nachträgliches Audit eingeleitet werden müssen, welche Schritte das Mitglied für die Reaktivierung der Mitgliedschaft unternehmen muss und wer eine Sperrung wieder aufheben darf.

Policies. VOs sind mit diversen Policies konfrontiert: Policies der realen Organisationen regeln den Umgang mit den lokalen Ressourcen und Diensten, Policies der Communities stellen Verhaltensregeln innerhalb der Communities dar und VO-weite Policies regeln den Umgang mit virtuellen Ressourcen und Diensten. Für das VO-Management bedeutet dies, Policies zu erkennen und – wenn sie VO-spezifisch sind – durchzusetzen.

Management von Lebenszyklen. VOs unterliegen einem Lebenszyklus, der durch das Zustandsmodell in Abbildung 3.12 ausgedrückt wird. Dies bedeutet für das VO-Management, den aktuellen Zustand im Lebenszyklus sicher identifzieren zu können und die situativ erforderlichen Zustandstransitionen anstoßen zu können.

Generalisierung. VO-Management umfasst *allgemein* anwendbare Maßnahmen und darf insofern nicht fallspezifisch sein. Dies bedeutet insbesondere die Unabhängigkeit von Implementierungstechnologien, Grid-Middleware-Konzepten und speziellen Anwendungsfällen.

VO-Spektrum. VO-Management operiert auf VOs unterschiedlicher Charakteristika, die in den Abschnitten 2.1 und 3.1 adressiert wurden. Das VO-Management darf jedoch nicht abhängig von spezifischen Ausprägungen einzelner VOs sein. Dies impliziert u.a. Skalierbarkeitsanforderungen, Unabhängigkeit von Lebenszyklusgeschwindigkeiten und geographischen Verteilungen.

Workflows. Effizienz im VO-Management hängt wesentlich von den verwendeten Workflows ab. Die Workflows des VO-Managements müssen eine gut ausbalanzierte Granularität bieten, um einerseits Mehrdeutigkeiten zu vermeiden, andererseits aber auch Performanzverluste zu vermeiden. Zusätzlich müssen die VO-Management-Workflows in das Gesamtgefüge einer oder mehrerer VOs eingebunden werden. Dies bedeutet nicht nur die Interaktion mit den die VO konstituierenden realen Organisationen, sondern auch – als Folge der Makroebene in Abbildung 2.5(c) – die Unterstützung VO-übergreifender Prozesse.

Logging und Monitoring. Im Rahmen der Überwachung und Kontrolle von VO-Lebenszyklen müssen sowohl die angestoßenen Workflows als auch die VO-spezifischen Aktivitäten der VO-Mitglieder ständig überprüft werden. Dazu sind neben Maßnahmen zur Sammlung von Daten (*logging*) auch Visualisierungsmethoden und geeignete Snapshot-Techniken erforderlich. Da in diesem Zusammenhang umfangreiche Protokoll-Daten anfallen, ist festzulegen, in welchem Umfang solche Daten erfasst

und gespeichert werden sollen. Dabei sind die gesetzlichen Rahmenbedingungen zu beachten. Einerseits muss also definiert werden, welche Informationen gespeichert werden müssen, andererseits ist zu bestimmen, was aus Datenschutzgründen nicht gespeichert werden darf. Die Erfordernisse in diesem Bereich unterscheiden sich sicherlich erheblich zwischen verschiedenen Wissenschaftsbereichen (z.B. Medizin und Hochenergiephysik) oder zwischen Industrie und rein wissenschaftlichen VOs. Dies beinhaltet auch die Festlegung, wie nicht weiter verwendete Daten zu löschen sind (spezifische Sicherheits- und Datenschutzanforderungen der VO) sowie die entsprechende Löschung dieser Daten (*VO Garbage Collection*).

Informationsmodell. Für ein effizientes VO-Management ist ein adäquates Repository im Sinne einer „VO-MIB“ erforderlich, aus dem zu jedem Zeitpunkt die aktuelle Konfiguration jeder VO abgerufen werden kann. Das Repository enthält für jede VO mindestens eine Beschreibung der VO, deren Namen, den aktuellen Mitgliederstand, die aktuellen Rollen und wie diese besetzt sind, die beteiligten realen Organisationen und eine Beschreibung der von ihnen bereitgestellten Kontingente, die aktuelle Lebenszyklusphase sowie eine Vertrauensklassifikation aller Partner und eine Spezifikation aller VO-weiten Policies.

Kollaborationsplattform. Zur Koordination der VO-Aktivitäten und zu Kooperationszwecken richtet das VO-Management entsprechende Kollaborationsplattformen (z.B. Groupware, E-Mail-Verteiler, Web-Portale, Wikis, gemeinsam nutzbare Dateiablagebereiche) ein.

Ausscheiden von Mitgliedern. Neben der Sperrung von Mitgliedern (siehe Anforderung „Behandlung von Mitgliedern.“ können Mitglieder aus verschiedenen Gründen aus einer VO ausscheiden:

- reguläres Ausscheiden, d.h. das Mitglied verlässt die VO, z.B. ein Doktorand am Ende seiner Promotion oder der temporäre Arbeitsvertrag des Mitglieds endet. Es muss also ein Verfahren geben, das sicherstellt, dass das Ausscheiden eines Mitglieds auch technisch umgesetzt wird, d.h. dass das Mitglied aus der VO-Verwaltung gelöscht wird. In diesem Zusammenhang ist eine zeitliche Befristung der Mitgliedschaft sinnvoll, erfordert dann aber einen Prozess zur Verlängerung von Mitgliedschaften.
- Ausschluss eines Mitglieds aus der VO, z.B. wegen groben Fehlverhaltens. Hierbei muss festgelegt sein, wer in solchen Fällen über den Ausschluss von VO-Mitgliedern entscheidet, wie ein solches Fehlverhalten aufgedeckt werden kann und ob der Ausschluss komplett oder partiell ist.

3.2 Anforderungen an VO-Managementarchitekturen

Während im Abschnitt 3.1.2 Erwartungen an das VO-Management dargestellt wurden, werden in diesem Abschnitt daraus folgende Anforderungen an VO-Management*architekturen* abgeleitet. Diese gliedern sich in funktionale Anforderungen, in denen die erforderliche Funktionalitäten der Architektur ausgedrückt wird, und nicht-funktionale Anforderungen, die auf die Verwendbarkeit, Verfügbarkeit, Leistungsfähigkeit und Wartbarkeit zielen, aber auch Implementierungsaspekte, Schnittstellenfragen, und Ausführungsaspekte betreffen. Funktionale und nicht-funktionale Anforderungen werden im Folgenden durch Anwendungsfälle (*use cases*) beschrieben. Sie verdeutlichen in ihren Szenarios, welche Funktionalitäten eine VO-Managementarchitektur (VOMA) liefern muss und welche Eigenschaften diese besitzen. Use Cases werden hier als *Black Box* aufgefasst, um keine Realisierungsaspekte zu präjudizieren.

Allgemein bedeutet *managen* aus Sicht einer Managementarchitektur „den Zugriff auf ein Managementobjekt (MO) über Managementkommandos, die mittels Managementprotokoll an die zu managende Instanz übermittelt werden, die das Managementobjekt beherbergt“ [Hegering u. a., 1999]. Im Folgenden werden diese Funktionalitäten näher beschrieben. Sie werden aus den Managementanforderungen, wie sie im Abschnitt 3.1 dargestellt wurden, abgeleitet. Um die Anforderungen zu strukturieren, werden die Anwendungsfälle nach dem VO-Lebenszyklus geordnet. Zur Spezifikation der Anforderungen ist zunächst die Kenntnis der Aktoren, die mit VOMA interagieren, notwendig.

3.2.1 Beschreibung der Aktoren

Das Management Virtueller Organisationen in Grids ist ein komplexer Vorgang, der sich aus vielen Einzelschritten in den einzelnen Lebenszyklusphasen einer VO zusammensetzt. Die im Abschnitt 3.1 beschriebene Ausgangssituation hat darüber hinaus deutlich gemacht, dass die am VO-Management direkt oder indirekt beteiligten Aktoren in diversen Rollen am Managementprozess partizipieren. Für VOMA können die folgenden *Hauptaktoren* aus den Szenarios identifiziert werden. Dabei wird – entsprechend den Bemerkungen im Abschnitt 3.1 – angenommen, dass die VOMA-Systemgrenze die *lokalen IT-Managementarchitekturen* nicht berücksichtigt.

Die Szenarios im Abschnitt 3.1.1 haben gezeigt, dass der Impuls zur Formation von VOs von der Community-Ebene ausgeht und durch eine *Rolle* (und *nicht* durch eine Person: Michael Schiffers gründet keine VO als Michael Schiffers, sondern allenfalls in der Rolle beispielsweise eines Principal InvestigatorsPI) oder eine Gruppe – etwa in Form eines Executive Committees wie in DEISA – erfolgt. Abbildung 3.11 macht dies noch einmal deutlich. Wir abstrahieren von diesen Spezifika und bezeichnen den VO-Gründer als VO-Initiator (VO-I) (siehe Tabelle 3.1).

Von ihm geht zwar der Impuls zur Gründung einer VO aus, er wird sie aber in der Regel weder managen noch administrieren. In Analogie zu klassischen Consumer/Provider-

Aktor *VO-Initiator*	
Bezeichner	VO-I
Ebene	Community-Ebene
Beschreibung	Initiiert die Formation einer VO
Beispiele	DEISA Executive Committee, DEISA Konsortium, DEISA DECI PI, EmerGrid Krisenstab, PI in der D-Grid C3-Community
Assoziierte Anwendungsfälle (Abschnitt 3.2.2)	F01 (Initiieren einer VO-Gründung), F02 (Initialisieren einer VO), T04 (Löschen einer VO)

Tabelle 3.1: Zusammenfassung des Aktors *VO-Initiator*

Szenarios nimmt er damit eine typische Consumer-Rolle ein. Er will mit der VO einen bestimmten (Anwendungs-) Zweck erreichen und erwartet dementsprechend von VOMA Funktionalitäten, die ihn bei der Formation einer VO unterstützen, wie etwa das Starten des VO-Gründungsprozesses oder die zweckorientierte Initialisierung der VO.

Nach der Initiierung einer VO durch einen `VO-I` übernimmt in der Regel eine Person oder eine Gruppe von Personen auf der Community-Ebene die Bereitstellung der zu etablierenden VO, indem die für den Betrieb der VO notwendigen Strukturen und Prozesse konfiguriert werden. Wir bezeichnen diese Rolle als VO-Provider (VO-P), da sie – analog zu klassischen Resource und Service Providern – eine VO zur Nutzung bereitstellt (siehe Tabelle 3.2). Der `VO-P` ist der eigentliche „Owner“ der VO, so wie ein Resource Provider der „Owner“ einer Resource ist.

Die Rolle des `VO-P` wird hier allerdings insofern umfassender gesehen, als ihm hier die Verantwortung für die „reibungslose“ Bereitstellung der VO über deren gesamte Lebensdauer zukommt. Dazu erwartet er von VOMA Funktionalitäten, die ihn bei der Bereitstellung einer VO unterstützen wie etwa die Initialisierung der VO-Strukuren und -Prozessen, die Institutionalisierung des erforderlichen VO-Managements, die Festlegung von VO-Policies, die Beendigung von VOs, die Überwachung von VO-Aktivitäten, die Auditierung von VOs oder die Abrechnung von VO-eigenen Leistungen.

In der Regel werden in aktuellen Grid-Projekten die Rollen `VO-I` und `VO-P` de facto als identisch betrachtet, da sie fast immer von den gleichen Institutionen wahrgenommen werden. Dennoch sind beide Rollen zu unterscheiden, wie das DEISA-Beispiel zeigt: Während die Gründung einer VO zwar von einem DECI-PI initiiert werden kann, wird dessen Hauptaugenmerk jedoch nicht auf der Bereitstellung von VO-spezifischen Strukturen und Prozessen liegen, so wie der Einkäufer eines Produktionsbetriebes auch nicht auf die Produktionsstrukturen und -prozesse fokussieren wird.

Aktor *VO-Provider*	
Bezeichner	VO-P
Ebene	Community-Ebene
Beschreibung	Bereitstellung der VO-Strukturen und -Prozesse
Beispiele	DEISA Lenkungsausschuss, DEISA Konsortium, DEISA Operation Team, EmerGrid Krisenstab, Atlas Projektgruppe in der D-Grid HEP-Community
Assoziierte Anwendungsfälle (Abschnitt 3.2.2)	F01 (Initiieren einer VO-Gründung), F02 (Initialisieren einer VO), F03 (Start einer VO), T04 (Löschen einer VO)

Tabelle 3.2: Zusammenfassung des Aktors *VO-Provider*

Im Rahmen der Initialisierung einer VO etabliert der `VO-P` die Rolle des VO-Managers (VO-M), der folglich auf der VO-Ebene angesiedelt ist (siehe Tabelle 3.3). Der `VO-M` „führt die Geschäfte“ der VO, so wie ein Vorstandsvorsitzender die Geschäfte einer Aktiengesellschaft führt. Sein Ziel ist der reibungslose *Betrieb* der VO, um das Ziel der VO zu erreichen. Dazu erwartet er von VOMA Unterstützung beim Start der VO nach deren Initialisierung, beim Terminieren der VO, beim Festlegen und Durchsetzen VO-spezifischer Policies und beim Management von Rollen. Da VOs *virtuelle* Dienste und *virtuelle* Ressourcen anbieten, ist es eine der Aufgaben des `VO-M`, deren Bereitstellung zu managen.

Aktor *VO-Manager*	
Bezeichner	VO-M
Ebene	VO-Ebene
Beschreibung	Betrieb der VO und Rollenmanagement in der VO
Beispiele	EmerGrid Krisenstab, Community-Sprecher in den D-Grid Communities, Projektsprecher in IPY
Assoziierte Anwendungsfälle (Abschnitt 3.2.2)	F02 (Initialisieren einer VO), F03 (Start einer VO), B04 (Hinzufügen von Rollen), B05 (Löschen von Rollen), B06 (Ändern von Rollen), B07 (Aufnahme von Ressourcen/Diensten), B08 (Löschen von Ressourcen/Diensten), T02 (Kündigen von SLAs), T03 (Stoppen einer VO), T04 (Löschen einer VO)

Tabelle 3.3: Zusammenfassung des Aktors *VO-Manager*

VO-Manager sind in der heutigen Grid-Praxis nur schwer von VO-Initiatoren und VO-Providern zu unterscheiden, da diese Rollen fast ausnahmslos in Personalunion ausgefüllt werden. Eine Trennung der Rollen ist jedoch für eine systematische Behandlung von VO-Managementfragen dringend erforderlich.

Typische Beispiele eines VO-Managers bilden die Community-Sprecher im D-Grid, aber auch die Projektleiter im IPY.

Während der VO-Manager auf die Bereitstellung von VO-Betriebsstrukturen und -prozessen fokussiert, steht für den VO-Administrator (VO-A) die Administration der VO im Vordergrund. Der `VO-A` ist folglich auf der VO-Ebene (siehe Tabelle 3.4) angesiedelt. VO-Administratoren stellen die zur Zeit wohl bekanntesten Rollen des VO-Managements dar. Dies hängt damit zusammen, dass VO-Management heute im wesentlichen als reines Mitgliedsmanagement verstanden wird, eine Aufgabe, die typischerweise von VO-Administratoren wahrgenommen wird.

Aktor *VO-Administrator*	
Bezeichner	VO-A
Ebene	VO-Ebene
Beschreibung	Administration von VOs
Beispiele	DEISA Operations Team, Community-Administratoren im D-Grid
Assoziierte Anwendungsfälle (Abschnitt 3.2.2)	F03 (Start einer VO), B01 (Hinzufügen von Mitgliedern), B02 (Löschen von Mitgliedern), B03 (Ändern von Mitgliedschaften), B04 (Hinzufügen von Rollen), B05 (Löschen von Rollen), B06 (Ändern von Rollen), B07 (Aufnahme von Ressourcen/Diensten), B08 (Löschen von Ressourcen/Diensten), T02 (Kündigen von SLAs)

Tabelle 3.4: Zusammenfassung des Aktors *VO-Administrator*

Der VO-Administrator erwartet von VOMA eine Unterstützung bei der Aufnahme und Einrichtung neuer Mitglieder, dem Löschen von Mitgliedern, dem Ändern von Mitgliederattributen, der Ausgabe von Zertifikaten, dem Sperren von Ressourcen und Diensten und möglicherweise – je nach Ausprägung der VO – auch Unterstützung in Wartungs- und Supportfragen.

Auch die Unterscheidung von VO-Managern und VO-Administratoren ist zunächst nicht offensichtlich. Dennoch ist die Rollenausprägung des VO-Managers von der des VO-Administrators zu differenzieren: Während nämlich der `VO-M` die VO selbst als Entität in den Vordergrund stellt, sind es beim `VO-A` vornehmlich die Mitglieder und virtuellen Ressourcen bzw. Dienste. Als typische Beispiele eines VO-Administrators können deshalb

das DEISA Operations Team und die Community-Administratoren im D-Grid gesehen werden.

Im Laufe des Lebenszyklus einer VO wird eine Menge VO-spezifischer Informationen erhoben oder bereitgestellt. Ein Großteil dieser Informationen muss für spätere Auditierungen kurz-, mittel- oder langfristig archiviert werden. Dies ist die Aufgabe des VO-Archive-Managers (VO-AM) (siehe Tabelle 3.5).

Aktor *VO-Archive-Manager*	
Bezeichner	VO-AM
Ebene	VO-Ebene
Beschreibung	Archivierung von VO-Konfigurationen
Beispiele	keine spezifischen Rollen in den Szenarios
Assoziierte Anwendungsfälle (Abschnitt 3.2.2)	T01 (Archivieren von VOs)

Tabelle 3.5: Zusammenfassung des Aktors *VO-Archive-Manager*

Um den operativen Betrieb einer VO aufzunehmen und an wechselnde Gegebenheiten anzupassen, schließt eine VO diverse SLAs mit lokalen Providern und möglicherweise Repräsentanten anderer VOs ab. Gegenstände solcher SLAs können Zugriffsgarantien, Kontigentierungen oder Abrechnungsmodalitäten sein. Die für das SLA-Management verantwortliche Rolle ist der VO-SLA-Manager (VO-SM) (siehe Tabelle 3.6)[19].

Neben diesen Rollen mit direktem VO-Bezug sind auch Entitäten auf der RO-Ebene am VO-Management beteiligt. Deren Hauptaugenmerk liegt auf den lokalen Ressourcen und Diensten, die für eine Nutzung durch die VO bereitgestellt werden und häufig durch die VO verwaltet werden (wie im IPY-Szenario). Reale Organisationen stellen in der Regel Kontingente ihrer Ressourcen bzw. auf diese eingeschränkte Dienste bereit. ROs agieren in diesem Zusammenhang deshalb als Quota-Provider (Q-P) (siehe Tabelle 3.7).

Als `Q-P` greift diese Entität nicht in das aktive VO-Management ein, sondern definiert die lokalen Zugangs-Policies zu den bereitgestellten Kontingenten. Ein `Q-P` erwartet daher die Unterstützung bei der Delegation, der Überwachung und Freigabe seiner Kontingente. Typische Beispiele eines solchen Quota-Providers sind die lokalen Rechenzentren, die Kontingente ihrer Kapazitäen dem DEISA-Konsortium zur Verfügung stellen, oder die Organisationen der D-Grid-Community InGrid, die im Rahmen ingenieurwissenschaftlicher Anwendungen Lizenzen von Simulationspaketen aus einem Lizenzenpool bereitstellen.

[19] Ressource-Interaktionen und deren Abbildung auf SLAs in Grids werden in [Czajkowski u. a., 2002] diskutiert.

Aktor *VO-SLA-Manager*	
Bezeichner	VO-SM
Ebene	VO-Ebene
Beschreibung	Management von VO-spezifischen SLAs
Beispiele	keine spezifischen Rollen in den Szenarios
Assoziierte Anwendungsfälle (Abschnitt 3.2.2)	F02 (Initialisieren einer VO), T02 (Kündigen von SLAs)

Tabelle 3.6: Zusammenfassung des Aktors *VO-SLA-Manager*

Aktor *Quota-Provider*	
Bezeichner	Q-P
Ebene	RO-Ebene
Beschreibung	Bereitstellung lokaler Ressourcen- und Dienstkontingente
Beispiele	Rechenzentren in DEISA, Provider von Speicherkapazitäten in D-Grid und IPY, Simulationsdienste und Lizenzen in EmerGrid
Assoziierte Anwendungsfälle (Abschnitt 3.2.2)	F02 (Initialisieren einer VO), B07 (Aufnahme von Ressourcen/Diensten), B08 (Löschen von Ressourcen/Diensten)

Tabelle 3.7: Zusammenfassung des Aktors *Quota-Provider*

Die Aufnahme von Mitgliedern und die Einbringung von Ressourcen und Diensten erfordert die Kooperation mit lokalen IT-Managementsystemen unterschiedlichster Ausprägung. Im Kontext dieser Arbeit werden sie abstrakt als Local Manager (LM) (siehe Tabelle 3.8) bezeichnet.

Komplementär zu diesen Hauptaktoren stellen *Sekundäraktoren* unterstützende Funktionalitäten für das VO-Management zur Verfügung. Typische Sekundäraktoren können lokale Managementsysteme, Datenbanksysteme mit Logging- und Benutzerinformationen, Monitoringsysteme oder Grid-Middleware-Systeme sein. Zusätzlich werden noch Standardaktoren verwendet, die hier jedoch wegen ihrer Offensichtlichkeit nicht weiter beschrieben werden. Dazu zählen beispielsweise Mitglieder, Benutzer (*user*) oder Bewerber zu VOs (*applicants*). Weitere spezifische Rollen werden im Kontext der einzelnen Anwendungsfälle bei

Aktor *Local Manager*	
Bezeichner	LM
Ebene	RO-Ebene
Beschreibung	Durchführung lokaler IT-Managementoperationen
Beispiele	lokale Administratorengruppen
Assoziierte Anwendungsfälle (Abschnitt 3.2.2)	F02 (Initialisieren einer VO), T02 (Kündigen von SLAs)

Tabelle 3.8: Zusammenfassung des Aktors *Local Manager*

Bedarf dort beschrieben.

3.2.2 Beschreibung der Anwendungsfälle

Die eben beschriebenen Aktoren betrachten die Aufgaben des VO-Managements aus unterschiedlichen Sichten und erwarten daher von einer VO-Managementarchitektur auch unterschiedliche Unterstützungsmechanismen. Die folgenden Anwendungsfälle beschreiben diese Mechanismen, strukturiert nach den wesentlichen Lebenszyklusphasen einer VO. Die Beschreibung der Anwendungsfälle folgt der Vorgehensweise in [Hennicker, 2006; Larman, 2001; Robertson u. Robertson, 2006]

Anwendungsfälle im Rahmen der VO-Formation

Im Rahmen der Formation einer VO treten die folgenden Anwendungsfälle auf: Initiieren einer VO-Gründung durch den VO-Initiator, Initialisieren einer VO durch den VO-Provider und Starten einer VO durch den VO-Manager. Diese Anwendungsfälle werden nun im Detail betrachtet.

Initiieren einer VO-Gründung
Die Initiierung einer VO geht vom VO-Initiator (`VO-I`) aus. Im primären Szenario wird die Gründung einer VO eingeleitet, Sekundärszenarios adressieren Fehlerfälle wie fehlende Autorisierung, unvollständige Spezifikation, Prozessabbruch und Duplikate.

Name des Anwendungsfalls: F01 (Initiieren einer VO-Gründung).

Kurzbeschreibung: Dieser Anwendungsfall adressiert das Initiieren einer VO-Gründung innerhalb einer Community.

Initiierender Aktor: VO-Initiator (`VO-I`)

Vorbedingungen: Der `VO-I` ist Mitglied einer Organisation der Community und besitzt die erforderlichen Rechte, eine VO gründen zu dürfen. Ist die Gründung der VO mit Kosten verbunden, steht dem `VO-I` ein entsprechendes Budget zur Verfügung.

Invarianten: keine

Minimale Sicherheit: Alle Aktivitäten des `VO-I` werden protokolliert (*logging*).

Szenario: Primär

Bezeichnung: Erfolgreiche Gründung einer VO.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. `VO-I` kommuniziert Gründungsanforderung an VOMA.
2. `VO-I` spezifiziert dabei die Annahmen, Einschränkungen und Anforderungen für die Gründung einer VO. Dazu dient ihm eine Template-basierte Spezifikation der VO.
3. VOMA lokalisiert `VO-P` (entweder direkt oder möglicherweise über einen entsprechenden Broker).

Nachbedingungen: Für die Initialisierung der vom `VO-I` beantragten VO ist ein `VO-P` identifiziert und beauftragt, den weiteren Gründungsprozess gemäß der vorgegebenen Spezifikation zu managen. Das VOMA-Informationsmodell enthält ein instanziiertes, in der Regel noch nicht initialisiertes VO-Modell.

Szenario: Sekundär I

Bezeichnung `VO-I` ist nicht zur Gründung einer VO autorisiert.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. Vorgehensweise wie im Primär-Szenario.
2. Nach der Kommunikation der Gründungsanforderung überprüft VOMA die Autorisierung zur Gründung.
3. VOMA unterrichtet den `VO-I` über die fehlende Autorisierung.
4. Weitere Aktivitäten des `VO-I` werden nicht zugelassen.

Nachbedingungen: Keine Änderungen im VOMA-Informationsmodell. Das Abuse-Repository enthält einen Missbrauchseintrag.

Szenario: Sekundär II

Bezeichnung Mit dem VOMA Spezifikations-Template lässt sich die gewünschte Situation nicht beschreiben.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. Vorgehensweise wie im Primär-Szenario.
2. Im Rahmen der Spezifikation der VO wird ein flexibler Erweiterbarkeitsmechanismus angeboten.
3. Mit der erweiterten Spezifikation wird das Primär-Szenario wieder aufgenommen.

Nachbedingungen: wie Primär-Szenario. Dem `VO-I` wird angeboten, die erweiterte Schablone als Default zu speichern.

Szenario: Sekundär III

Bezeichnung Es kann kein `VO-P` lokalisiert werden.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. Vorgehensweise wie im Primär-Szenario. Nach Schritt 3:
2. `VO-I` wird autorisiert, selbst als `VO-P` zu agieren.

Nachbedingungen: wie Primär-Szenario.

Szenario: Sekundär IV

Bezeichnung Die beantragte VO existiert bereits.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. Vorgehensweise wie im Primär-Szenario. Nach der Spezifikation der VO:
2. VOMA prüft bestehende Existenz der VO.
3. VOMA kommuniziert Duplikat und verlangt neuen Namen.

Nachbedingungen: wie Primär-Szenario.

Nicht-funktionale Anforderungen: Die folgenden nicht-funktionalen Anforderungen sind für diesen Anwendungsfall relevant:

- Die VO-Initiierung muss vollständig automatisiert ablaufen können.

Eine graphische Darstellung dieses Anwendungsfalles ist in Abbildung 3.13 auf Seite 112 zu finden.

Initialisieren einer VO

Die Initialisierung einer VO geht vom VO-Provider (`VO-P`) aus. Im primären Szenario wird die Entität „VO“ initialisiert, Sekundärszenarios adressieren Fehlerfälle wie fehlerhafte Kommunikation und Kooperation mit lokalen Quota-Providern und die Zurückname des Gründungsantrags.

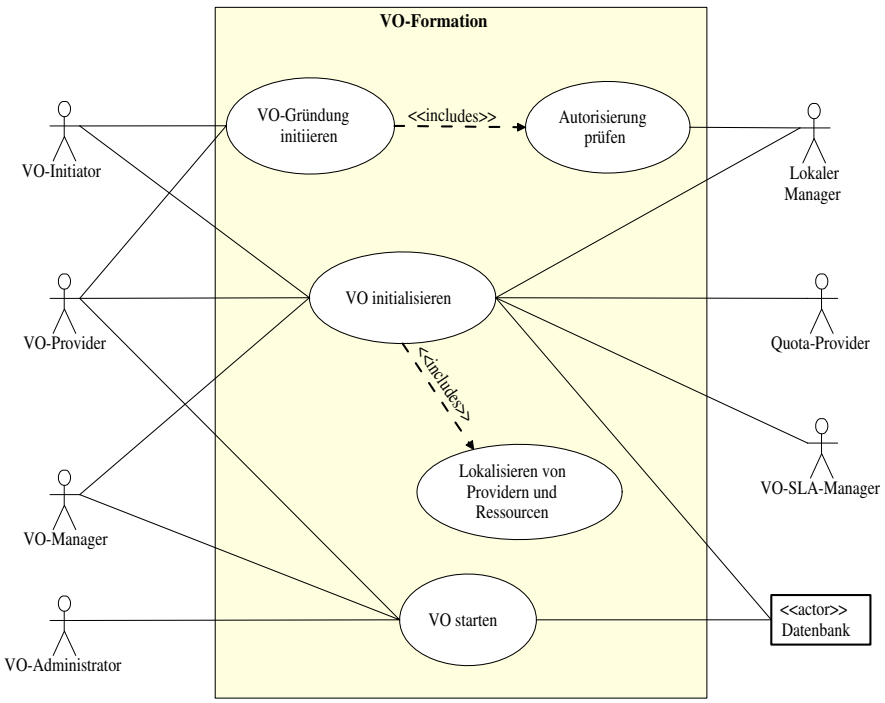


Abbildung 3.13: Formation einer VO

Name des Anwendungsfalls: F02 (Initialisieren einer VO).

Kurzbeschreibung: Dieser Anwendungsfall adressiert das Initialisieren einer VO nach deren Initiierung.

Initiierender Aktor: VO-Provider (`VO-P`)

Vorbedingungen: Der `VO-P` ist – möglicherweise aus einer Liste potenzieller `VO-P`s – identifiziert. Er ist allen Mitgliedern der Community bekannt und genießt deren Vertrauen. Er ist mit den erforderlichen Rechten ausgestattet, um den Initialisierungsprozess durchzuführen. Dem `VO-P` sind die Randbedingungen, unter denen die VO gegründet werden soll, bekannt. Dem VO sind die lokalen Manager bekannt.

Invarianten: Die Historie des VO-Templates wird fortgeschrieben.

Minimale Sicherheit: Alle Aktivitäten des `VO-P` werden protokolliert (*logging*).

Szenario: Primär

Bezeichnung: Erfolgreiche Initialisierung einer VO.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. `VO-P` erhält vom `VO-I` Auftrag zur Initialisierung einer VO.
2. `VO-P` definiert anhand des vom `VO-I` übergebenen Templates die Initialisierungs-Workflows.

3. `VO-P` identifiziert die zur Leistungserbringung der VO notwendigen Quota-Provider.
4. `VO-P` lädt diese zur Teilnahme an VO ein.
5. `VO-P` institutionalisiert die Rolle des VO-SLA-Manager (`VO-SM`).
6. `VO-P` besetzt die Rolle des `VO-SM`.
7. `VO-SM` vereinbart mit jedem Quota-Provider und Local Manager SLA und Zugangs-Policies.
8. `VO-P` stellt virtuelle Dienste und Ressourcen bereit.
9. `VO-P` spezifiziert VO-weite Policies.
10. `VO-P` institutionalisiert die Rolle des VO-Managers `VO-M`.
11. `VO-P` besetzt die Rolle des `VO-M`.
12. `VO-P` initialisiert Informationssysteme.
13. `VO-P` kommuniziert an `VO-I` Betriebsbereitschaft der VO.

Nachbedingungen: Die vom `VO-I` beantragte VO ist betriebsbereit. Das VOMA-Informationsmodell enthält ein instanziiertes und initialisiertes VO-Modell. SLAs mit Quota-Providern und dem Local Management sind abgeschlossen.

Szenario: Sekundär I

Bezeichnung Die erforderlichen Quota-Provider können nicht identifiziert werden oder verweigern die Teilnahme oder deren SLAs sind nicht konform zur VO-Spezifikation.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. Vorgehensweise wie im Primär-Szenario.
2. `VO-P` komuniziert an `VO-I` Fehler in Schritt 3.
3. `VO-P` fragt nach `VO-I` nach Alternativen.
4. `VO-P` nimmt Prozess in Schritt 3 wieder auf.

Nachbedingungen: Die vom `VO-I` beantragte VO ist betriebsbereit. Das VOMA-Informationsmodell enthält ein instanziiertes und initialisiertes VO-Modell. Falls die Iteration nach einer definierten Anzahl von Versuchen nicht terminiert, Abbruch. Damit werden alle bisherigen Initialisierungen zurückgenommen

Szenario: Sekundär II

Bezeichnung `VO-I` zieht seinen Gründungsantrag zurück.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. Vorgehensweise wie im Primär-Szenario.

2. Sobald der VO-I seinen Gründungsantrag zurückzieht, wird der VO-I um eine Bestätigung gebeten.
3. VO-P bricht die aktuellen Aktivitäten ab.
4. VO-P stellt den Ursprungszustand wieder her.
5. Die VO erhält den Status undefined.

Nachbedingungen: Sämtliche Initialisierungen sind zurückgenommen.

Nicht-funktionale Anforderungen: Die folgenden nicht-funktionalen Anforderungen sind für diesen Anwendungsfall relevant:

- Die Initialisierung muss vollständig automatisiert ablaufen können.
- Die Beauftragung des VO-P durch den VO-I muss vollständig automatisiert durchgeführt werden können.
- Der VO-P muss nicht unbedingt Mitglied einer Community sein.

Eine graphische Darstellung dieses Anwendungsfalles ist in Abbildung 3.13 auf Seite 112 zu finden.

Starten einer VO

Der Start einer VO erfolgt durch den VO-Manager, sobald die VO betriebsbereit ist. Im primären Szenario wird die VO erfolgreich gestartet, im Sekundärszenario wird eine fehlerhafte Rollenbesetzung betrachtet.

Name des Anwendungsfalls: F03 (Starten einer VO).

Kurzbeschreibung: In diesem Anwendungsfall wird eine zuvor initialisierte VO gestartet.

Initiierender Aktor: VO-Manager (VO-M)

Vorbedingungen: Nach der Initialisierung ist die VO betriebsbereit. Alle für den Zweck der VO notwendigen Ressourcen und Dienst stehen bereit. Die Rolle des VO-Managers ist besetzt. Sämtliche erforderlichen Informationssysteme sind initialisiert.

Invarianten: keine

Minimale Sicherheit: Alle Aktivitäten des VO-M werden protokolliert (*logging*).

Szenario: Primär

Bezeichnung: Die VO wird erfolgreich gestartet.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. VO-P kommuniziert an VO-M die Betriebsbereitschaft der VO.
2. VO-M startet VO-Logging.
3. VO-M legt Rollen und deren Rechte fest.
4. VO-M besetzt Rolle des VO-Administrators VO-A.
5. VO-M öffnet VO für die Aufnahme von Mitgliedern.
6. VO-M überführt VO in den Zustand in Betrieb.

Nachbedingungen: Die vom VO-I beantragte VO ist gestartet. Die Rolle des VO-Administrators ist besetzt. VO befindet sich im Zustand in Betrieb.

Szenario: Sekundär I

Bezeichnung Die VO-A-Rolle kann nicht besetzt werden.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. Vorgehensweise wie im Primär-Szenario.
2. VO-M übernimmt selbst die Administrator-Rolle.

Nachbedingungen: wie Primär-Szenario

Nicht-funktionale Anforderungen: Die folgenden nicht-funktionalen Anforderungen sind für diesen Anwendungsfall relevant:

- Der Start muss vollständig automatisiert ablaufen.

Eine graphische Darstellung dieses Anwendungsfalles ist in Abbildung 3.13 auf Seite 112 zu finden.

Anwendungsfälle im Rahmen des VO-Betriebes

Im Rahmen des Betriebes einer VO treten die folgenden Anwendungsfälle auf: Aufnahme und Löschen von Mitgliedern, Ändern von Mitgliedschaften, Hinzufügen, Löschen und Ändern von Rollen, Hinzufügen und Entfernen von Ressourcen und Diensten. Diese Anwendungsfälle werden nun im Detail betrachtet.

Aufnahme von Mitgliedern
VO-Mitglieder können aufgenommen werden, sobald die VO betriebsbereit ist. Im primären Szenario wird ein Bewerber erfolgreich als Mitglied aufgenommen, Sekundärszenarios adressieren Fehlerfälle wie die Rückweisung von Bewerbern, die Verweigerung von Super-VOs, die zu kurze verbleibende Lebensdauer einer VO und die Doppelmitgliedschaft. Eine spezielle Betrachtung erfordert der Sonderfall des Gastmitglieds.

Name des Anwendungsfalls: B01 (Aufnahme von Mitgliedern).

Kurzbeschreibung: In diesem Anwendungsfall wird die Aufnahme von Mitgliedern in eine VO betrachtet.

Initiierender Aktor: VO-Administrator (`VO-A`)

Vorbedingungen: Dem Administrator liegt eine Anforderung vor, eine Person oder eine Personengruppe als Mitglied einer VO aufzunehmen. Die VO existiert und ist gestartet.

Invarianten: keine

Minimale Sicherheit: Alle Aktivitäten des `VO-A` werden protokolliert (*logging*).

Szenario: Primär

Bezeichnung: Der Bewerber wird als Mitglied aufgenommen.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. `VO-A` überprüft die Authentizität des Bewerbers.
2. `VO-A` überprüft die Autorisierung.
3. `VO-A` überprüft Zulässigkeit der Bewerbung gegen historische Daten.
4. `VO-A` generiert Mitgliedschaft gemäß vorhandener Zertifikate.
5. `VO-A` ordnet Bewerber Rollen zu.
6. `VO-A` quittiert Bewerber die Mitgliedschaft.
7. `VO-A` überprüft Zulässigkeit der Mitgliedschaft in allen Super-VOs.
8. `VO-A` quittiert Mitgliedschaft in Super-VOs.

Nachbedingungen: Die beantragte Mitgliedschaft ist erfolgreich. Der Bewerber ist auch Mitglied in allen die VO umfassenden Super-VOs. Dem Mitglied ist eine Rolle zugeteilt.

Szenario: Sekundär I

Bezeichnung Der Bewerber kann nicht authentifiziert werden oder besitzt die erforderlichen Rechte nicht oder seine Bewerbung wird als „nicht zulässig“ eingestuft.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. Vorgehensweise wie im Primär-Szenario.
2. Nach der Überprüfung verweigert der `VO-A` die Mitgliedschaft.
3. `VO-A` bricht Aufnahmeprozess ab.

Nachbedingungen: keine Änderungen der Mitgliederkonstellation

Szenario: Sekundär II

Bezeichnung Super-VOs verweigern die Mitgliedschaft.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. Vorgehensweise wie im Primär-Szenario.
2. Nach der Überprüfung der Super-VO-Zugehörigkeit informiert `VO-A` den Bewerber.
3. `VO-A` bricht Aufnahmeprozess ab.

Nachbedingungen: wie Primär-Szenario außer Mitgliedschaft in Super-VOs.

Szenario: Sekundär III

Bezeichnung Verbleibende Lebensdauer der VO ist zu kurz.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. `VO-A` überprüft die verbleibende Lebensdauer der VO.
2. `VO-A` informiert Bewerber und Heimatorganisation, dass verbleibende Lebensdauer einen definierten Schwellwert unterschritten hat.
3. Wenn Bewerber dennoch fortfahren möchte: Weiter wie im Primär-Szenario. Sonst Abbruch.

Nachbedingungen: wie Primär-Szenario oder Sekundär-Szenario II falls Super-VOs keine ausreichende Lebensdauer besitzen.

Szenario: Sekundär IV

Bezeichnung Bewerber bewirbt sich um Gast-Mitgliedschaft.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. Vorgehensweise wie im Primär-Szenario, allerdings mit eingeschränkten Rechten und Rollen.

Nachbedingungen: wie Primär-Szenario oder Sekundär-Szenarios I bis III.

Szenario: Sekundär V

Bezeichnung Bewerber ist schon Mitglied.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. Vorgehensweise wie im Primär-Szenario.
2. `VO-A` weist Bewerber auf mögliche Doppelmitgliedschaft hin.
3. Abbruch falls Bewerber dies wünscht, Neubewerbung unter neuem Namen sonst.

Nachbedingungen: wie Primär-Szenario oder Sekundär-Szenarios I bis III.

Nicht-funktionale Anforderungen: Die folgenden nicht-funktionalen Anforderungen sind für diesen Anwendungsfall relevant:

- Die Mitgliedschaft muss sofort gültig sein.
- Gast-Mitgliedschaften sind nur für einen begrenzten Zeitrahmen gültig.

Eine graphische Darstellung dieses Anwendungsfalles ist in Abbildung 3.14 zu finden.

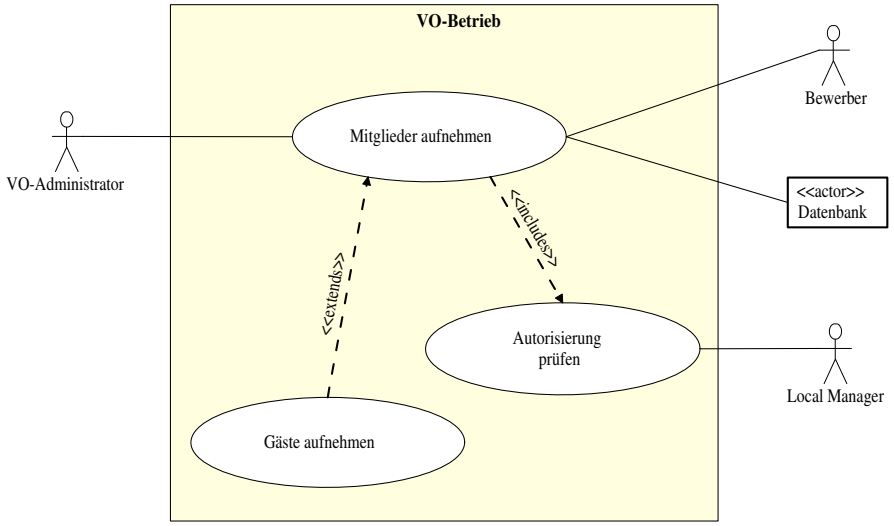


Abbildung 3.14: Hinzufügen von Mitgliedern

Löschen von Mitgliedern
VO-Mitglieder können gelöscht werden, sobald die VO betriebsbereit ist und die zu lösende Mitgliedschaft auch tatsächlich besteht. Im primären Szenario wird das Mitglied gelöscht, Sekundärszenarios adressieren Fehlerfälle wie die fehlende Autorisierung zum Löschen und eine nicht bestehende Mitgliedschaft. Einen Sonderfall stellt das gleichzeitige Löschen von Mitgliedschaften in hierarchisch über- und untergeordneten VOs dar.

Name des Anwendungsfalls: B02 (Löschen von Mitgliedern).]

Kurzbeschreibung: In diesem Anwendungsfall wird das Löschen von Mitgliedern einer VO betrachtet.

Initiierender Aktor: VO-Administrator (`VO-A`)

Vorbedingungen: Dem Administrator liegt eine Anforderung vor, eine Person oder eine Personengruppe als Mitglied einer VO zu löschen. Die VO existiert und ist gestartet.

Invarianten: keine

Minimale Sicherheit: Alle Aktivitäten des `VO-A` werden protokolliert (*logging*).

Szenario: Primär

Bezeichnung: Das Mitglied wird gelöscht.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. `VO-A` überprüft die Authentizität des Antragstellers.
2. `VO-A` überprüft die Autorisierung des Antragstellers.
3. `VO-A` stoppt alle aktuellen Aktivitäten des Mitglieds.
4. `VO-A` entfernt Mitglied aus allen Rollen.
5. `VO-A` quittiert Mitglied die Aufhebung der Mitgliedschaft.
6. `VO-A` archiviert Mitgliedsdaten in Historie.
7. `VO-A` blockiert Mitglied für neue Mitgliedschaften.
8. `VO-A` löscht Mitgliedschaft in allen Super-VOs.
9. `VO-A` löscht Mitgliedschaft in allen Sub-VOs.
10. `VO-A` quittiert Mitgliedschaft in Super-VOs.

Nachbedingungen: Die Mitgliedschaft ist gelöscht, auch in den über- und untergeordneten Super- und Sub-VOs.

Szenario: Sekundär I

Bezeichnung Der Antragsteller kann nicht authentifiziert werden oder besitzt die erforderlichen Rechte nicht.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. Vorgehensweise wie im Primär-Szenario.
2. Nach der Überprüfung verweigert der `VO-A` das Löschen der Mitgliedschaft.
3. `VO-A` bricht Löschprozess ab.

Nachbedingungen: keine Änderungen der Mitgliederkonstellation

Szenario: Sekundär II

Bezeichnung Super- oder Sub-VOs verweigern das Löschen.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. Vorgehensweise wie im Primär-Szenario.
2. Nach der Überprüfung der Zugehörigkeit zu Super- oder Sub-VOs und der Weigerung dieser VOs, die Mitgliedschaft dort zu löschen, informiert `VO-A` das Mitglied.
3. `VO-A` fährt mit Löschvorgang nur für beantragte VO fort.

Nachbedingungen: wie Primär-Szenario außer Mitgliedschaft in Super- und Sub-VOs.

Szenario: Sekundär III

Bezeichnung Zu löschendes Mitglied ist kein Mitglied.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. Vorgehensweise wie im Primär-Szenario.
2. `VO-A` kommuniziert Fehlersituation.
3. Abbruch.

Nachbedingungen: keine Änderung der Mitgliederkonstellation.

Nicht-funktionale Anforderungen: Die folgenden nicht-funktionalen Anforderungen sind für diesen Anwendungsfall relevant:

- Das Löschen der Mitgliedschaft muss sofort gültig sein.
- Gast-Mitgliedschaften werden automatisch nach einem begrenzten Zeitrahmen ungültig.

Eine graphische Darstellung dieses Anwendungsfalles ist in Abbildung 3.15 zu finden.

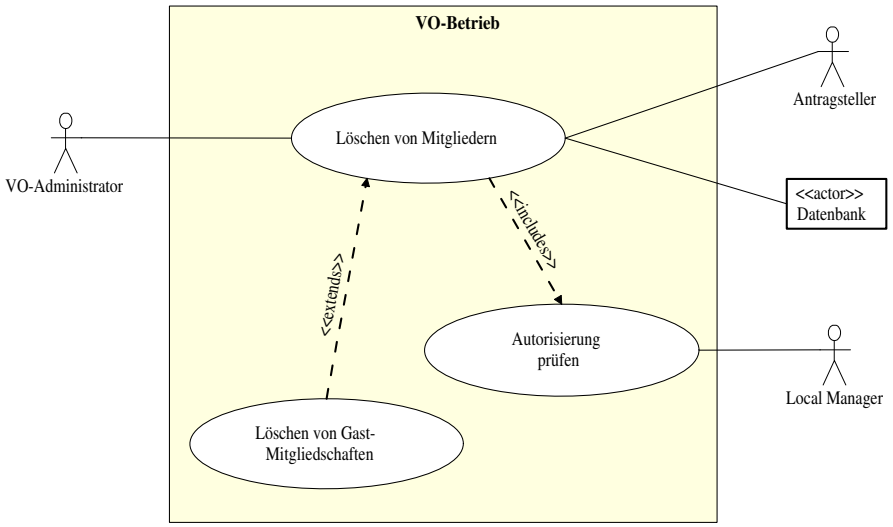


Abbildung 3.15: Löschen von Mitgliedern

Ändern von Mitgliedschaften

Die Attribute eines VO-Mitgliedes können geändert werden, sobald die VO betriebsbereit ist und die zu ändernde Mitgliedschaft auch tatsächlich besteht. Im primären Szenario ist dies der Fall. In Sekundärszenarios werden Fehlerfälle wie die fehlende Autorisierung zum Ändern von Attributen und eine nicht bestehende Mitgliedschaft behandelt. Einen Sonderfall stellt die Propagation von Attributänderungen in hierarchisch über- und untergeordneten VOs dar.

Name des Anwendungsfalls: B03 (Ändern von Mitgliedschaften).

Kurzbeschreibung: In diesem Anwendungsfall werden Mitgliederattribute geändert.

Initiierender Aktor: VO-Administrator (`VO-A`)

Vorbedingungen: Dem Administrator liegt eine Anforderung vor, die Mitgliedschaften einer Person oder einer Personengruppe zu ändern. Die VO existiert und ist gestartet.

Invarianten: Mitgliedschaften bleiben bestehen, nur die Attribute ändern sich.

Minimale Sicherheit: Alle Aktivitäten des `VO-A` werden protokolliert (*logging*).

Szenario: Primär

Bezeichnung: Mitgliederattribute werden geändert.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. `VO-A` überprüft die Authentizität des Antragstellers.
2. `VO-A` überprüft die Autorisierung des Antragstellers.
3. `VO-A` setzt alle aktuellen Aktivitäten des Mitglieds aus.
4. `VO-A` ändert die Attribute.
5. `VO-A` quittiert Mitglied die Änderung seiner Mitgliedschaftsattribute.
6. `VO-A` gibt alle Aktivitäten des Mitglieds im Rahmen der neuen Attribute wieder frei.
7. `VO-A` initiiert Attributänderung in allen betroffenen Super-VOs.

Nachbedingungen: Die Mitgliedschaft ist geändert, auch in den übergeordneten Super-VOs.

Szenario: Sekundär I

Bezeichnung Der Antragsteller kann nicht authentifiziert werden oder besitzt die erforderlichen Rechte nicht.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. Vorgehensweise wie im Primär-Szenario.
2. Nach der Überprüfung verweigert der `VO-A` die Änderung von Attributen.
3. `VO-A` bricht Änderungsprozess ab.

Nachbedingungen: keine Änderungen

Szenario: Sekundär II

Bezeichnung Super-VOs verweigern das Ändern.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. Vorgehensweise wie im Primär-Szenario.
2. Nach der Überprüfung der Zugehörigkeit zu Super-VOs und der Weigerung dieser VOs, die Mitgliedschaft dort zu ändern, informiert `VO-A` das Mitglied.
3. `VO-A` fährt mit Änderungsvorgang nur für beantragte VO fort.

Nachbedingungen: wie Primär-Szenario außer Mitgliedschaft in Super-VOs.

Szenario: Sekundär III

Bezeichnung Zu änderndes Mitglied ist kein Mitglied.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. Vorgehensweise wie im Primär-Szenario.
2. `VO-A` kommuniziert Fehlersituation.
3. Abbruch.

Nachbedingungen: keine Änderung

Nicht-funktionale Anforderungen: Die folgenden nicht-funktionalen Anforderungen sind für diesen Anwendungsfall relevant:

- Das Ändern der Mitgliedschaft muss sofort gültig sein.

Eine graphische Darstellung dieses Anwendungsfalles ist in Abbildung 3.16 zu finden.

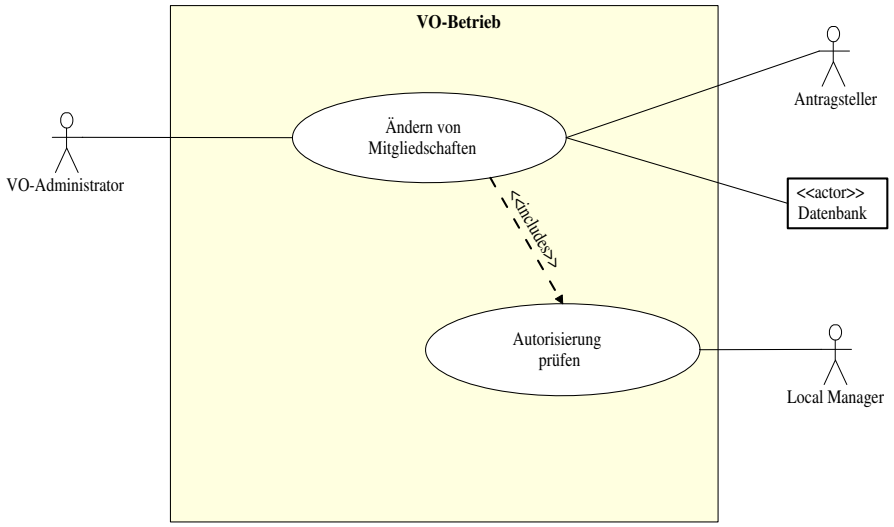


Abbildung 3.16: Ändern von Mitgliedschaften

Hinzufügen von Rollen
In analoger Weise zu Mitgliedern werden Anwendungsfälle für das Management von Rollen betrachtet. Rollen können erst hinzugenommen werden wenn die VO betriebsbereit ist. Im primären Szenario wird eine Rolle erfolgreich hinzugefügt, Sekundärszenarios adressieren Fehlerfälle wie die Inkonsistenz bestehender und neuer Rollen und die zu kurze verbleibende Lebensdauer einer VO. Eine spezielle Betrachtung erfordert der Sonderfall, dass Rollen nicht belegt werden können.

Name des Anwendungsfalls: B04 (Hinzufügen von Rollen).

Kurzbeschreibung: In diesem Anwendungsfall wird das Hinzufügen neuer Rollen zu einer existierenden VO adressiert. Der Impuls für die Aufnahme neuer Rollen kommt vom VO-Manager, der VO-Administrator setzt ihn um.

Initiierender Aktor: VO-Manager (`VO-M`)

Vorbedingungen: Die VO existiert und ist betriebsbereit.

Invarianten: keine

Minimale Sicherheit: Alle Aktivitäten des `VO-M` und des `VO-A` werden protokolliert (*logging*).

Szenario: Primär

Bezeichnung: Aufnahme einer neuen Rolle.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. `VO-M` spezifiziert die Rolle.
2. `VO-M` überprüft Rollenkonsistenz (Konflikte mit bestehenden Rollen, Konflikte mit bestehenden Rollen in Super- und Sub-VOs).
3. `VO-M` übergibt Rollenspezifikation und Rollenbelegung an `VO-A`.
4. `VO-A` richtet neue Rolle ein.
5. `VO-A` ordnet Mitgliedern Rollen zu.
6. `VO-A` informiert Mitglieder über neue Rollenzugehörigkeit.

Nachbedingungen: Die neue Rolle ist eingerichtet und bestehenden Mitgliedern zugeordnet.

Szenario: Sekundär I

Bezeichnung Rolleninkonsistenz.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. Vorgehensweise wie im Primär-Szenario.
2. Nach Feststellen der Inkonsistenz:

3. VO-M überprüft Spezifikation.
4. VO-M überprüft Spezifikationen bestehender Rollen.
5. VO-M startet Rollenaufnahme erneut.

Nachbedingungen: wie Primär-Szenario

Szenario: Sekundär II

Bezeichnung Verbleibende Lebensdauer der VO ist zu kurz.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. VO-M überprüft die verbleibende Lebensdauer der VO.
2. VO-M bricht Rollenaufnahme ab, falls verbleibende Lebensdauer einen definierten Schwellwert unterschritten hat.

Nachbedingungen: keine Änderungen in der Rollenkonstellation.

Szenario: Sekundär III

Bezeichnung Rollen können nicht mit Mitgliedern belegt werden.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. VO-M belegt die Rollen mit VO-M und VO-A vor.

Nachbedingungen: wie Primär-Szenario

Nicht-funktionale Anforderungen: Die folgenden nicht-funktionalen Anforderungen sind für diesen Anwendungsfall relevant:

- Die neue Rolle muss sofort gültig sein.

Eine graphische Darstellung dieses Anwendungsfalles ist in Abbildung 3.17 zu finden.

Löschen von Rollen

VO-Rollen können erst gelöscht werden, wenn die VO betriebsbereit ist und die zu löschende Rolle auch tatsächlich existiert. Im primären Szenario werden die Rollenbelegungen zurückgesetzt und die Rolle gelöscht. In Sekundärszenarios werden Fehler- und Sonderfälle adressiert wie die Nicht-Existenz der zu löschenden Rolle, die Ununterbrechbarkeit der bestehenden Aktivitäten der Rolleninhaber und die Unmöglichkeit, Alternativrollen anbieten zu können.

Name des Anwendungsfalls: B05 (Löschen von Rollen).

Kurzbeschreibung: In diesem Anwendungsfall wird das Löschen von Rollen einer VO betrachtet.

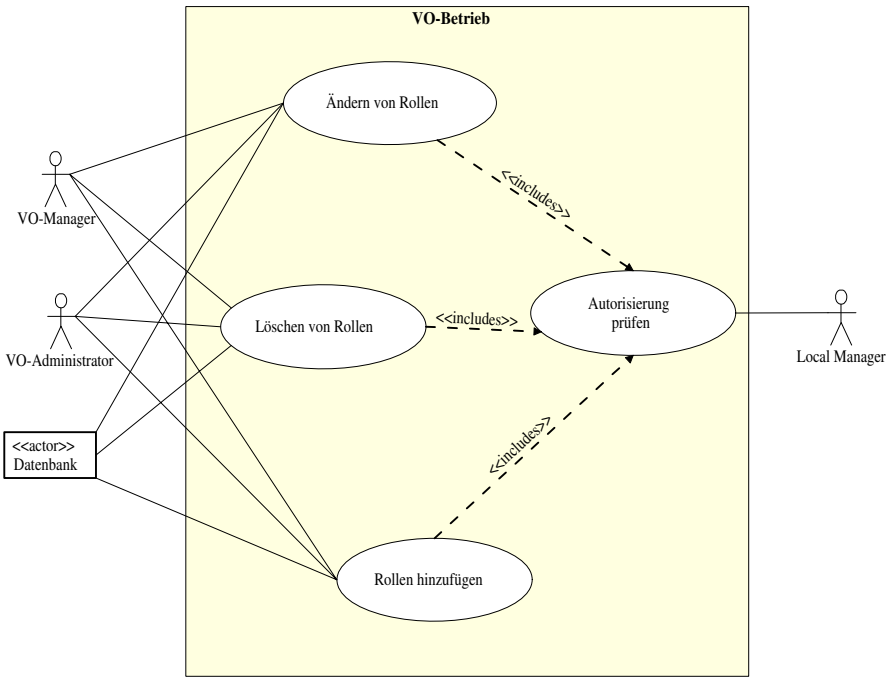


Abbildung 3.17: Rollenmanagement in VOs

Initiierender Aktor: VO-Manager (`VO-M`)

Vorbedingungen: Die VO existiert und ist gestartet.

Invarianten: keine

Minimale Sicherheit: Alle Aktivitäten des `VO-M` und des `VO-A` werden protokolliert (*logging*).

Szenario: Primär

Bezeichnung: Das Rolle wird gelöscht.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. `VO-M` überprüft die Existenz der zu löschenden Rolle und die aktuelle Belegung mit Mitgliedern.
2. `VO-A` übernimmt die weiteren Aktivitäten.
3. `VO-A` stoppt VO.
4. `VO-A` archiviert alle bisherige Rollenaktivitäten.
5. `VO-A` entfernt alle Rollenzuordnungen.
6. `VO-A` ordnet Rolleninhabern alternative Rollen zu.
7. `VO-A` startet VO in neuer Konfiguration.

Nachbedingungen: Die Rolle ist gelöscht und die bisherigen Rolleninhaber sind alternativen Rollen zugeordnet.

Szenario: Sekundär I

Bezeichnung Rolle existiert nicht.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. Vorgehensweise wie im Primär-Szenario.
2. Nach der Überprüfung bricht der Löschvorgang ab.
3. `VO-M` kommuniziert Fehlersituation.

Nachbedingungen: keine Änderungen der Rollenkonstellation

Szenario: Sekundär II

Bezeichnung Aktivitäten der Rolleninhaber lassen sich nicht stoppen.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. Vorgehensweise wie im Primär-Szenario.
2. `VO-M` kommuniziert begrenzte Lebensdauer der Rolle an Rolleninhaber.
3. `VO-M` initiiert Löschvorgang erneut nach Ende der Lebensdauer.

Nachbedingungen: wie Primär-Szenario

Szenario: Sekundär III

Bezeichnung Rolleninhabern können keine Alternativrollen angeboten werden oder Mitglieder verweigern diese Zuordnung.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. Vorgehensweise wie im Primär-Szenario.
2. `VO-A` ordnet den Mitgliedern die Standardrolle Member zu.
3. `VO-A` kommuniziert dies an die Mitglieder.

Nachbedingungen: wie Primär-Szenario

Nicht-funktionale Anforderungen: Die folgenden nicht-funktionalen Anforderungen sind für diesen Anwendungsfall relevant:

- Das Löschen der Rollen muss bis auf Szenario II sofort gültig sein.

Eine graphische Darstellung dieses Anwendungsfalles ist in Abbildung 3.17 auf Seite 125 zu finden.

Ändern von Rollen

Die Attribute und Belegungen von Rollen können geändert werden, sobald die VO betriebsbereit ist und die zu ändernde Rolle auch tatsächlich besteht bzw. belegt ist. Im primären Szenario ist dies der Fall. In Sekundärszenarios werden Fehlerfälle und Sonderfälle behandelt, wie sie schon in den Anwendungsfällen B04 und B05 auftraten.

Name des Anwendungsfalls: B06 (Ändern von Rollen).

Kurzbeschreibung: In diesem Anwendungsfall werden bestehende Rollen in ihren Attributen und Zuordnungen geändert.

Initiierender Aktor: VO-Manager (`VO-M`)

Vorbedingungen: Die VO existiert und ist gestartet.

Invarianten: Rollen bleiben bestehen, nur die Attribute und Zuordnungen ändern sich.

Minimale Sicherheit: Alle Aktivitäten des `VO-M` und des `VO-A` werden protokolliert (*logging*).

Szenario: Primär

Bezeichnung: Rollenattribute werden geändert.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. `VO-M` spezifiziert die Rollenattribute.
2. `VO-M` überprüft Rollenkonsistenz (Konflikte mit bestehenden Rollen, Konflikte mit bestehenden Rollen in Super- und Sub-VOs).
3. `VO-M` übergibt neue Rollenspezifikation und aktuelle Rollenbelegung an `VO-A`.
4. `VO-A` übernimmt die weiteren Aktivitäten.
5. `VO-A` stoppt VO.
6. `VO-A` archiviert alle bisherige Rollenaktivitäten.
7. `VO-A` ändert die Rollenattribute bzw. die Rollenzuordnungen.
8. `VO-A` quittiert Rolleninhabern die Änderung der Rollenattribute.
9. `VO-A` startet VO in neuer Konfiguration.

Nachbedingungen: Die Rolle ist geändert.

Sekundär-Szenarios: Als Sekundär-Szenarios können die analogen Szenarios der Anwendungsfälle B04 und B05 verwendet werden.

Nicht-funktionale Anforderungen: Die folgenden nicht-funktionalen Anforderungen sind für diesen Anwendungsfall relevant:

- Das Ändern der Rolle muss sofort gültig sein.

Eine graphische Darstellung dieses Anwendungsfalles ist in Abbildung 3.17 auf Seite 125 zu finden.

Hinzufügen von Ressourcen und Diensten
In den folgende Anwendungsfällen wird das Hinzufügen und Entfernen von Ressourcen und Diensten zu einer VO betrachtet. Ressourcen und Dienst, die von lokalen Quota-Providern bereitgestellt werden, können sowohl im Rahmen der Formation einer VO als auch zu einer betriebsbereiten VO hinzugefügt werden. Das primäre Szenario demonstriert den nominellen Fall, ein Fehlerfall liegt vor, wenn Ressourcen- bzw. Dienstinformationen unvollständig sind.

Name des Anwendungsfalls: B07 (Hinzufügen von Ressourcen und Diensten).

Kurzbeschreibung: In diesem Anwendungsfall wird das Hinzufügen von Ressourcen und Diensten zu einer existierenden VO adressiert.

Initiierender Aktor: VO-Manager (`VO-M`)

Vorbedingungen: Die VO existiert und ist betriebsbereit, der lokale Quota-Provider beantragt die Aufnahme einer identifizierbaren Ressource oder eines Dienstes.

Invarianten: keine

Minimale Sicherheit: Alle Aktivitäten des `VO-M` und des `VO-A` werden protokolliert (*logging*).

Szenario: Primär

Bezeichnung: Aufnahme einer neuen Ressource oder eines neuen Dienstes.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. `VO-M` überprüft Vollständigkeit der erforderlichen Information.
2. `VO-M` übergibt Spezifikation an `VO-A`.
3. `VO-A` integriert neue Ressource oder neuen Dienst in VO-Informationsmodell.
4. `VO-A` publiziert Verfügbarkeit und Zugriffs-Policies.

Nachbedingungen: Die neue Ressource oder der Dienst ist verfügbar.

Szenario: Sekundär I

Bezeichnung Unvollständige Information.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. Vorgehensweise wie im Primär-Szenario.
2. Nach Feststellen der Unvollständigkeit:
3. `VO-M` kommuniziert Unvollständigkeit an Quota-Provider.

4. VO-M bricht Aufnahme ab.

Nachbedingungen: keine Änderungen

Nicht-funktionale Anforderungen: Die folgenden nicht-funktionalen Anforderungen sind für diesen Anwendungsfall relevant:

- Die neue Ressource oder der neue Dienst muss sofort verfügbar sein.

Eine graphische Darstellung dieses Anwendungsfalles ist in Abbildung 3.18 zu finden.

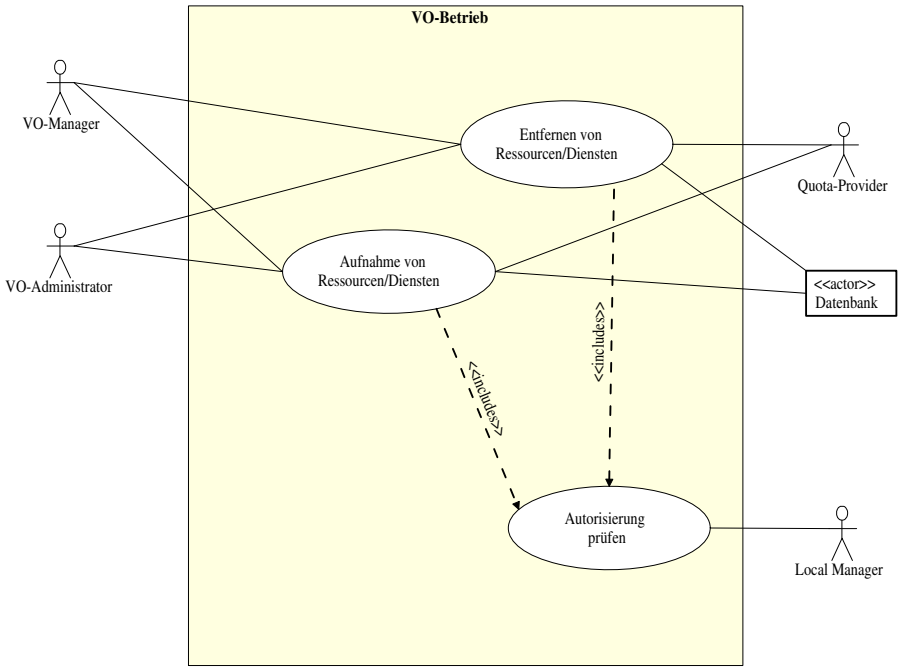


Abbildung 3.18: Ressourcen- und Dienstmanagement in VOs

Entfernen von Ressourcen und Diensten
Ressourcen und Dienste können erst entfernt werden wenn sie auch exisitieren. Dazu ist die Betriebsbereitschaft der VO erforderlich. Das Primär-Szenario drückt dies aus. Sekundär-Szenarios werden analog zu den Anwendungsfällen B05 und B07 behandelt.

Name des Anwendungsfalls: B08 (Entfernen von Ressourcen und Diensten).

Kurzbeschreibung: In diesem Anwendungsfall wird das Entfernen von Ressourcen und Diensten aus einer existierenden VO adressiert.

Initiierender Aktor: VO-Manager (VO-M)

Vorbedingungen: Die VO existiert und ist betriebsbereit, der lokale Quota-Provider beantragt die Entfernung einer identifizierbaren Ressource oder eines Dienstes. Der Quota-Provider ist *owner* des Dienstes oder der Ressource.

Invarianten: keine

Minimale Sicherheit: Alle Aktivitäten des `VO-M` und des `VO-A` werden protokolliert (*logging*).

Szenario: Primär

Bezeichnung: Entfernen einer Ressource oder eines Dienstes.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. `VO-M` überprüft Vollständigkeit der erforderlichen Information.
2. `VO-M` übergibt Identifikation der der Ressource oder des Dienstes an `VO-A`.
3. `VO-A` stoppt VO.
4. `VO-A` informiert alle aktuellen Ressource/Dienst-Nutzer.
5. `VO-A` archiviert alle Ressource/Dienst-Aktivitäten.
6. `VO-A` löscht Ressource oder Dienst.
7. `VO-A` startet VO in neuer Konfiguration.

Nachbedingungen: Die neue Ressource oder der Dienst ist verfügbar.

Sekundär-Szenarios: Als Sekundär-Szenarios können die analogen Szenarios der Anwendungsfälle B05 und B07 verwendet werden.

Nicht-funktionale Anforderungen: Die folgenden nicht-funktionalen Anforderungen sind für diesen Anwendungsfall relevant:

- Die Verfügbarkeit der Ressource oder des Dienstes erlischt sofort.

Eine graphische Darstellung dieses Anwendungsfalles ist in Abbildung 3.18 auf Seite 129 zu finden.

Anwendungsfälle im Rahmen der VO-Auflösung

Im Rahmen der Auflösung einer VO werden die folgenden Anwendungsfälle betrachtet: Archivierung von VO-Informationen, Kündigung von SLAs und Löschen einer VO. Diese Anwendungsfälle werden nun im Detail betrachtet.

Archivieren von VO-Informationen

Im Rahmen der Auflösung von VOs entsteht die Anforderung, die im Lauf des bisherigen Lebenszyklus der angefallenen Informationen zu archivieren, um sie einem späteren Audit

zuführen zu können oder zu statistischen Zwecken. Die hier angesprochenen Archivierungen erfordern eine betriebsbereite VO und für den VO-Archive-Manager den Zugang zu Archivsystemen.

Name des Anwendungsfalls: T01 (Archivieren von Informationen).

Kurzbeschreibung: In diesem Anwendungsfall wird das Archivieren von VO-bezogenen Informationen adressiert. Der Anwendungsfall erfordert zwar eine betriebsbereite VO, wird aber hier im Rahmen der Auflösung der VO betrachtet.

Initiierender Aktor: VO-Archive-Manager (`VO-AM`)

Vorbedingungen: Die VO existiert und ist betriebsbereit. Ein Archivsystem kann durch den `VO-AM`) genutzt werden.

Invarianten: Die Archivierung findet im laufenden VO-Betrieb statt.

Minimale Sicherheit: Alle Aktivitäten der beteiligten Rollen werden protokolliert (*logging*).

Szenario: Primär

Bezeichnung: Archivieren von VO-Informationen.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. `VO-AM` überprüft Verfügbarkeit des Archivsystems.
2. `VO-AM` führt Archivierung gemäß Archivierungs-Policy durch.
3. `VO-AM` schreibt Metadaten in Archivkatalog.

Nachbedingungen: Die zum Zeitpunkt der Archivierung aktuelle VO-Konstellation ist archiviert und der Metadaten-Katalog enthält einen neuen Eintrag.

Szenario: Sekundär I

Bezeichnung Archivsystem steht nicht zur Verfügung oder Kapazität ist ausgeschöpft.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. Vorgehensweise wie im Primär-Szenario.
2. Nach Feststellen der fehlenden Verfügbarkeit:
3. `VO-AM` kommuniziert an VO-Manager Undurchführbarkeit der Archivierung.
4. `VO-AM` bricht Archivierung ab.

Nachbedingungen: keine Archivierung

Nicht-funktionale Anforderungen: Die folgenden nicht-funktionalen Anforderungen sind für diesen Anwendungsfall relevant:

- Die Archivierung kann zeitgesteuert oder ereignisgesteuert erfolgen.
- Der Archiv-Eintrag steht allen VO-Teilnehmern entsprechend ihren Rechten zur Verfügung.

Eine graphische Darstellung dieses Anwendungsfalles ist in Abbildung 3.19 zu finden.

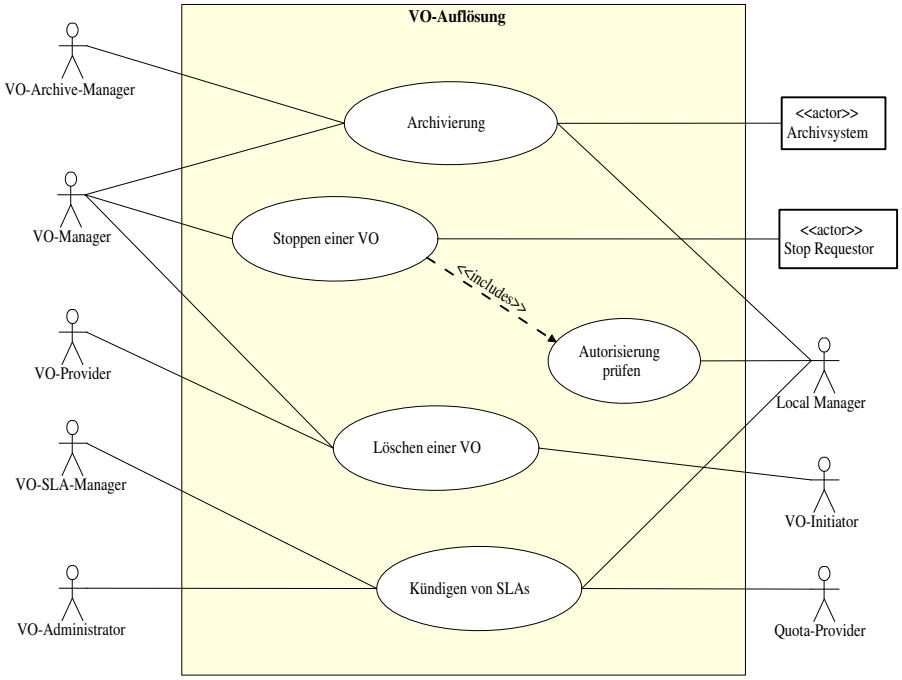


Abbildung 3.19: Auflösen von VOs

Kündigung von SLAs
Die Mitglieder und teilnehmenden Organisationen einer VO sind in ein Geflecht von mehr oder weniger formalen SLAs eingebunden. Wird die VO aufgelöst, werden einige SLAs durch das Fehlen der Geschäftsgrundlage standardmäßig auslaufen, andere werden durch Kündigung zurückgesetzt. Letzteres ist Gegenstand dieses Anwendungsfalles, der wesentlich auf den Aktivitäten des VO-SLA-Managers (`VO-SM`) beruht. Im Primär-Szenario wird ein SLA problemlos gekündigt, in Sekundär-Szenarios werden Sonderfälle behandelt, wie Seiteneffekte auf und durch andere SLAs sowie die mit Vertragsstrafen verbundene vorzeitige Kündigung [Schmidt, 2001].

Name des Anwendungsfalls: T02 (Kündigung von SLAs).

Kurzbeschreibung: In diesem Anwendungsfall wird die Kündigung von SLAs auf der VO-Ebene betrachtet

Initiierender Aktor: VO-SLA-Manager (`VO-SM`)

Vorbedingungen: Die VO existiert und ist betriebsbereit. Es besteht mindestens ein Service Level Agreement zwischen der VO – repräsentiert durch den VO-Manager – und einem lokalen RP/SP bzw. einer anderen VO, das vorzeitig zu kündigen ist.

Invarianten: keine

Minimale Sicherheit: Alle Aktivitäten der beteiligten Rollen werden protokolliert (*logging*).

Szenario: Primär

Bezeichnung: Fristgerechte Kündigung von SLAs.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. `VO-SM` überprüft Kündigungfristen.
2. `VO-SM` überprüft durch Seiteneffekte der Kündigung betroffene SLAs.
3. `VO-SM` kündigt SLA.
4. `VO-A` stoppt die Nutzung aller durch das SLA abgedeckten Ressourcen und Dienste zum Ende der zugestandenen Nutzungsphase.
5. `VO-A` entfernt die vom SLA betroffenen Ressourcen und Dienste am Ende der zugestandenen Nutzungsphase.

Nachbedingungen: SLA ist gekündigt und die noch verbleibende Nutzungsphase ist bekannt.

Szenario: Sekundär I

Bezeichnung Abhängigkeiten zwischen SLAs.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. Vorgehensweise wie im Primär-Szenario.
2. Seiteneffekte ergeben sich aus SLA-Abhängigkeiten. Wenn abhängige SLAs ebenfalls zu kündigen sind, wird das Primär-Szenario iterativ verwendet.

Nachbedingungen: wie Primär-Szenario für alle zu kündigenden SLAs

Szenario: Sekundär II

Bezeichnung Vorzeitige Kündigung von SLAs durch die VO.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. Vorgehensweise wie im Primär-Szenario.
2. Ist die Kündigung mit einer Vertragsstrafe verbunden: `VO-M` entscheidet über weitere Vorgehensweise.
3. `VO-SM` bricht Kündigung ab falls `VO-M` abbricht.
4. Sonst weiter im ab Schritt 3.

Nachbedingungen: entweder keine Änderungen oder wie im Primär-Szenario

Nicht-funktionale Anforderungen: Die folgenden nicht-funktionalen Anforderungen sind für diesen Anwendungsfall relevant:

- Die Kündigung kann zeitgesteuert oder ereignisgesteuert erfolgen.

Eine graphische Darstellung dieses Anwendungsfalles ist in Abbildung 3.19 auf Seite 132 zu finden.

Stoppen einer VO

VOs werden gestoppt, um den operativen Betrieb für einen definierten Zeitraum auszusetzen. Ein wesentlicher Anwendungsfall ist das anschließende Löschen der VO. Es sind aber auch Fälle denkbar, in denen der Betrieb der VO beispielsweise für Zwischenauditierungen ausgesetzt wird. Die VO kann nach der Unterbrechung dann wieder gestartet werden. Im Primär-Szenario wird der nominelle Fall behandelt, in Sekundär-Szenarios die Fehler- und Sonderfälle des nicht-autorisierten Stopps und der zu kurzen restlichen Lebensdauer.

Name des Anwendungsfalls: T03 (Stoppen einer VO).

Kurzbeschreibung: Dieser Anwendungsfall behandelt das Stoppen einer VO.

Initiierender Aktor: VO-Manager (`VO-M`)

Vorbedingungen: Die VO existiert und wurde gestartet.

Invarianten: aktuelle und geplante Anwendungen der VO-Mitglieder werden nicht unterbrochen.

Minimale Sicherheit: Alle Aktivitäten des `VO-M` werden protokolliert (*logging*).

Szenario: Primär

Bezeichnung: Stoppen der VO.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. `VO-M` liegt eine Anforderung zum Stopp der VO vor (hier als *Stop Requestor* bezeichnet).
2. `VO-M` überprüft die Gültigkeit der Anforderung.
3. `VO-M` unterrichtet alle Mitglieder der VO über den bevorstehenden Stopp.
4. `VO-M` versetzt VO in den Status betriebsbereit.
5. `VO-M` schreibt Statistik fort.

Nachbedingungen: VO ist gelöscht.

Szenario: Sekundär I

Bezeichnung Stopp-Anforderung ist ungültig wegen fehlender Autorisierung oder weil die VO schon gestoppt wurde.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. Vorgehensweise wie im Primär-Szenario.
2. `VO-M` kommuniziert Fehlerfall an *Stop Requestor* .
3. `VO-M` bricht Prozess zum Stoppen der VO ab.

Nachbedingungen: keine Änderungen

Szenario: Sekundär II

Bezeichnung Verbleibende Lebensdauer der VO ist zu kurz.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. `VO-M` überprüft die verbleibende Lebensdauer der VO.
2. `VO-M` informiert *Stop Requestor*, dass verbleibende Lebensdauer einen definierten Schwellwert unterschritten hat.
3. Wenn *Stop Requestor* dennoch fortfahren möchte: Weiter wie im Primär-Szenario. Sonst Abbruch.

Nachbedingungen: wie Primär-Szenario oder Sekundär-Szenario I.

Nicht-funktionale Anforderungen: Die folgenden nicht-funktionalen Anforderungen sind für diesen Anwendungsfall relevant:

- Das Stoppen der VO erfolgt automatisch.
- Das Stoppen der VO ist sofort wirksam.
- Das Stoppen kann zeitgesteuert oder ereignisgesteuert erfolgen.

Eine graphische Darstellung dieses Anwendungsfalles ist in Abbildung 3.19 auf Seite 132 zu finden.

Löschen einer VO

VOs besitzen eine begrenzte Lebensdauer, nach deren Ablauf die VO gelöscht wird. Dies geschieht im Primär-Szenario. Im Sekundär-Szenario wird der Fall adressiert, dass die VO noch nicht gestoppt wurde.

Name des Anwendungsfalls: T04 (Löschen einer VO).

Kurzbeschreibung: Dieser Anwendungsfall behandelt das Löschen einer VO.

Initiierender Aktor: VO-Provider (`VO-P`)

Vorbedingungen: Die VO existiert und ist gestoppt. Alle SLAs sind gekündigt, die VO ist archiviert und alle Ressourcen/Dienste sind freigegeben.

Invarianten: keine

Minimale Sicherheit: Alle Aktivitäten des `VO-P` werden protokolliert (*logging*).

Szenario: Primär

Bezeichnung: Löschen der VO.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. `VO-P` überprüft ob VO gestoppt wurde.
2. `VO-P` stoppt Logging..
3. `VO-P` löscht Rollen.
4. `VO-P` löscht Mitglieder.
5. `VO-P` informiert VO-Initiator (`VO-I`) über Löschung.
6. `VO-P` schreibt Statistik fort.
7. `VO-P` löscht alle VO-relevanten Daten, die nicht Statistik- und Audit-bezogen sind.

Nachbedingungen: VO ist gelöscht.

Szenario: Sekundär I

Bezeichnung VO wurde nicht gestoppt.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. Vorgehensweise wie im Primär-Szenario.
2. `VO-P` kommuniziert mit VO-Manager (`VO-M`), die VO zu stoppen.
3. Dann weiter mit Schritt 2 des Primär-Szenarios.

Nachbedingungen: wie Primär-Szenario

Nicht-funktionale Anforderungen: Die folgenden nicht-funktionalen Anforderungen sind für diesen Anwendungsfall relevant:

- Das Löschen der VO erfolgt automatisch.
- Das Löschen der VO ist sofort wirksam.

Eine graphische Darstellung dieses Anwendungsfalles ist in Abbildung 3.19 auf Seite 132 zu finden.

In den folgenden Tabellen 3.9, 3.10 und 3.11 wird kurz zusammengefasst, welche VO-Managementaktoren in welchen Grid-Szenarios des Abschnitts 3.1.1 zu finden sind, welche

Anwendungsfälle in den Szenarios abgedeckt werden und wie die Aktoren sich auf die Anwendungsfälle verteilen. Damit wird die Definierbarkeit der Anforderungen nachgewiesen. Die Vollständigkeit der Anforderungen kann naturgemäß nicht bewiesen werden, da eine solcher Nachweis Anwendungsspezifika berücksichtigen müsste, die nicht antizipierbar sind.

	Szenario			
Rolle	DEISA	D-Grid	EmerGrid	IPY
VO-Initiator	✓	✓	✓	✓
VO-Provider	✓	✓	✓	
VO-Manager		✓	✓	
VO-Administrator	✓	✓	✓	✓
VO-Archive-Manager			✓	
VO-SLA-Manager		✓	✓	
Quota-Provider	✓	✓	✓	✓
Local Manager	✓	✓	✓	✓

Tabelle 3.9: Referenz der Aktoren in den Szenarios

	Szenario			
Anwendungsfall	DEISA	D-Grid	EmerGrid	IPY
F01: Initiieren einer VO-Gründung	✓	✓	✓	✓
F02: Initialisieren einer VO	✓	✓	✓	✓
F03: Start einer VO		✓	✓	
B01: Hinzufügen von Mitgliedern	✓	✓	✓	✓
B02: Löschen von Mitgliedern	✓	✓	✓	✓
B03: Ändern von Mitgliedschaften	✓	✓	✓	✓
B04: Hinzufügen von Rollen	✓	✓	✓	
B05: Löschen von Rollen		✓	✓	
B06: Ändern von Rollen		✓		
B07: Aufnahme von Ressourcen/Diensten	✓	✓	✓	✓
B08: Löschen von Ressourcen/Diensten	✓	✓	✓	✓
T01: Archivieren von VOs	✓		✓	
T02: Kündigen von SLAs	✓		✓	
T03: Stoppen einer VO			✓	
T04: Löschen einer VO			✓	✓

Tabelle 3.10: Referenz der Anwendungsfälle in den Szenarios

	Aktoren							
Anwendungsfall	VO-I	VO-P	VO-M	VO-A	VO-AM	VO-SM	Q-P	LM
F01: Initiieren einer VO-Gründung	✓	✓						
F02: Initialisieren einer VO	✓	✓	✓			✓	✓	✓
F03: Start einer VO		✓	✓	✓				
B01: Hinzufügen von Mitgliedern				✓				
B02: Löschen von Mitgliedern				✓				
B03: Ändern von Mitgliedschaften				✓				
B04: Hinzufügen von Rollen			✓	✓				
B05: Löschen von Rollen			✓	✓				
B06: Ändern von Rollen			✓	✓				
B07: Aufnahme von Ressourcen/Diensten			✓	✓			✓	
B08: Löschen von Ressourcen/Diensten			✓	✓			✓	
T01: Archivieren von VOs					✓			
T02: Kündigen von SLAs			✓	✓		✓		✓
T03: Stoppen einer VO			✓					
T04: Löschen einer VO	✓	✓	✓					

Tabelle 3.11: Beteiligung der Aktoren an den Anwendungsfällen

3.2.3 Nicht-funktionale Anforderungen allgemeiner Art

Die Aufgabe von IT-Managementarchitekturen im Allgemeinen besteht darin, die Beschreibung und Übermittlung von Managementinformationen sowie den Zugriff auf Managementfunktionalitäten in einem organisatorischen Zusammenhang festzulegen. Ein solcher Architekturbegriff impliziert – siehe auch Abschnitt 2.3 – die Notwendigkeit der Spezifikation eines Organisations-, Informations-, Kommunikations- und Funktionsmodells. Die in den vorherigen Abschnitten ermittelten Aktoren und Anwendungsfälle adressieren in ihren funktionalen Anforderungen wesentliche Aspekte dieser Modelle, vor allem des Funktions- und des Organisationsmodells. Auch wenn einige nicht-funktionale Anforderungen dargestellt wurden, waren diese Anwendungsfall-spezifisch zu sehen. Unabhängig davon existieren jedoch nicht-funktionale Anforderungen allgemeinerer Art, die weitere Charakteristika der Teilmodelle einordnen.

Zugang zu Ressourcen und Diensten. In fast allen Szenarios bestehen Einschränkungen für den Zugriff auf Ressourcen und Dienste, die von VOMA zu

berücksichtigen sind und primär das VOMA-Funktionsmodell betreffen. Sie können wie folgt kategorisiert werden:

- zeitliche Einschränkungen (Zugriff ist erst nach einer Zeitspanne erlaubt; Zugriff ist nicht vor einem Zeitpunkt erlaubt)
- identitätsbezogene Einschränkungen (Zugriff ist nur erlaubt für bestimmte Identitäten; Zugriff ist für bestimmte Identitäten nicht erlaubt)
- gruppenspezifische Einschränkungen (Zugriff ist nur für Individuen erlaubt oder auch für Gruppen oder auch für Organisationen)
- geographische Einschränkungen (national, international, nur für bestimmte Regionen)
- Granularitätsbeschränkungen (vollständiger Zugang, partieller Zugang)
- Beschränkungen in der Zugriffsfrequenz (keine Einschränkungen, einmaliger Zugang, regulärer Zugang)
- kostenorientierte Einschränkungen (kostenloser Zugang, pauschaler Zugang, individuelle Gebührenregelung, Grad der Kreditwürdigkeit)

Technische Randbedingungen. Die technischen Randbedingungen sind vor dem Hintergrund des Grid-Umfeldes zu sehen. Sie führen zu folgenden nicht-funktionalen Anforderungen im Kontext aller Teilmodelle:

- Unabhängigkeit von der verwendeten Grid-Middleware
- Unabhängigkeit von speziellen Software-Releaseständen
- Notwendigkeit eines Migrationspfades für bestehende Lösungsansätze zum VO-Management
- Unterstützung Service-orientierter Architekturen (SOA)
- Bedienbarkeit über Web-Portale und Kommandozeilen

VOMA-Teilmodelle. Die vorher beschriebenen Aktoren und Anwendungsfälle müssen in den Kontext der VOMA-Teilmodelle gerückt werden. Sie implizieren funktionale Anforderungen an ein VOMA-Informationsmodell, ein Kommunikationsmodell, ein Organisationsmodell und vor allem ein Funktionsmodell. Dennoch sind einige nicht-funktionale Perspektiven zu berücksichtigen:

- Die VO-Managementarchitektur muss eine umfassende und Grid-weit integrierte Informationsbasis bereitstellen, deren Objekte eine einheitliche Sichtweise auf VOs darstellen. Diese Objekte müssen dabei in einer Art und Weise beschrieben werden können, die eine Abbildung auf Informationsmodelle lokaler Managementsysteme zulässt, die zur Beschreibung lokaler Ressourcen und Dienste eingesetzt werden. Dabei muss berücksichtigt werden, dass diese Informationsmodelle nicht kompatibel sind [Keller, 1998].

- Die VO-Managementarchitektur muss einen organisationsübergreifenden Kontext bieten, in dem ein problemloser Austausch von Managementinformationen und die Ausführung von Managementfunktionen möglich ist. Dazu müssen die involvierten Akteure in ihren Rollen und Positionen, ihren Gruppenzugehörigkeiten und ihren Kooperationsbeziehungen untereinander identifizierbar und spezifizierbar sein.
- Die Beschreibung von Kommunikationsvorgängen innerhalb der VO-Managementarchitektur umfasst die Festlegung der kommunizierenden Partner und der Kommunikationsmechanismen. Da der Fokus dieser Arbeit auf Grids (und damit auf dem Web Services-Umfeld) liegt, werden diese Mechanismen in die standardisierten WSA- und OGSA-Konzepte eingebettet. Weiterhin muss die VO-Managementarchitektur sowohl asynchrone als auch synchrone Kommunikationsformen unterstützen, um sowohl über kritische und abonnierte Ereignisse benachrichtigt zu werden (z.B. Fehlersituationen oder QoS-Verletzungen), als auch große Datenmengen über FTP-ähnliche Protokolle zu übertragen vermögen.
- Funktional sind bei einer VO-Managementarchitektur zwei Aspekte zu unterscheiden. Auf der einen Seite ist die Funktionalität zu betrachten, die einem Aktor an einer VO-Managementschnittstelle zur Verfügung gestellt wird. Der Aktor wird diese Funktionen nutzen, um „seine“ Perspektive zu realisieren. Aus der Sicht der Architektur müssen aber auch diejenigen Funktionsbausteine ermittelt werden, die die Abbildung der Interaktionen auf Funktionalitäten lokaler Managementwerkzeuge und -anwendungen realisieren. Die in Kapitel 7 vorzuschlagende Methodik zur Anwendung der VO-Managementarchitektur wird die Vorgehensweise an Beispielen zeigen.

3.3 Grobskizze der angestrebten Gesamtarchitektur

Basierend auf diesen Anforderungen ist in Abbildung 3.20 in erster Näherung eine grobe Einordnung der VO-Managementarchitektur im Kontext einer Grid-Infrastruktur, einer darüber liegenden Sicherheitsarchitektur (die in dieser Arbeit nur am Rande betrachtet wird) und dem Grid-Betriebsmanagement, das ebenfalls nicht Gegenstand dieser Arbeit ist, dargestellt.

Unabhängig von Implementierungsmodellen und Realisierungsdetails wird VOMA die Schnittstellen B, C und D bedienen bzw. nutzen. Wie die Anwendungsfälle zeigen, verwendet das VO-Management wesentliche Funktionalitäten einer im Grid vorhandenen Sicherheitsinfrastruktur über die Schnittstelle B. VO-Managementdienste selbst werden an der Schnittstelle C zur Verfügung gestellt. Orthogonal dazu stehen Deployment-Dienste, die die entwickelten VO-Managementlösungen dem operativen Betrieb an der Schnittstelle D zuführen.

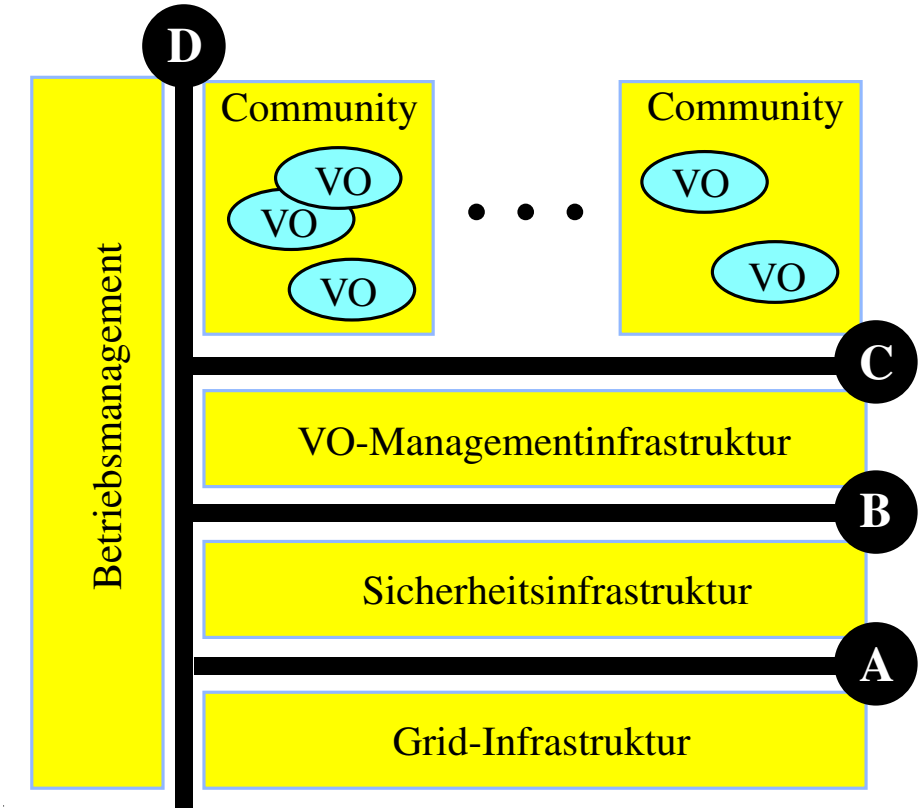


Abbildung 3.20: Einordnung der VO-Managementarchitektur

Innerhalb des VO-Managementblocks wird das Ebenenprinzip fortgesetzt. Abbildung 3.21 zeigt, dass VO-Managementanwendungen Workflows verwenden, die komplexe Managementdienste nutzen, die wiederum auf Basisdiensten aufbauen.

3.4 Zusammenfassung

Die vorher identifizierten Anforderungen funktionaler und nicht-funktionaler Art haben die Komplexität der Aufgabenstellung noch einmal deutlich gemacht. Die dort betrachteten Anwendungsfälle dienen im nachfolgenden Kapitel 4 als Instrumentarium zur Bewertung bestehender Ansätze zum VO-Management in Grids. Gleichzeitig dienen sie als „Treiber“ zur Entwicklung der VO-Managementarchitektur in den Kapiteln 5 und 6.

Als Ergebnis dieses Kapitels liegt nun ein durch Aktoren und Anwendungsfälle bestimmter Katalog funktionaler und nicht-funktionaler Anforderungen für die zu erstellende Managementarchitektur vor. Abbildung 3.20 gibt eine erste vage Vorstellung der Softwarearchitektur.

Der nächste Schritt besteht nun darin, das (statische) Informationsmodell über Klas-

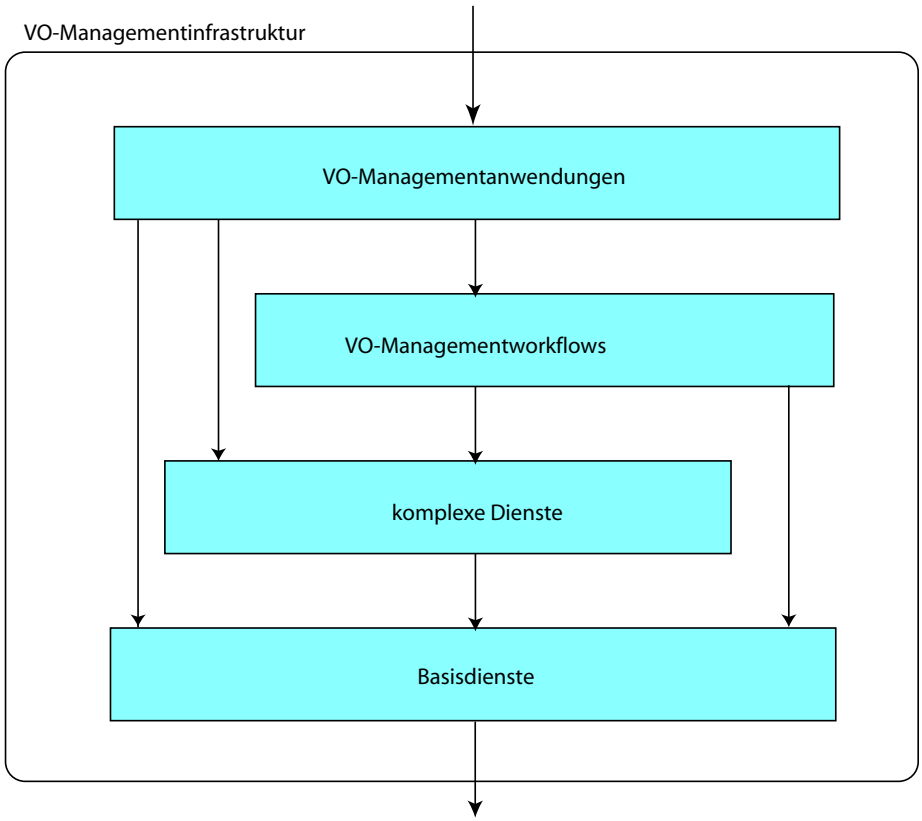


Abbildung 3.21: Basisarchitektur zum VO-Management

sendiagramme festzulegen. Diese kennzeichnen die statischen Informationsstrukturen. Das Verhalten der durch sie modellierten Objekte, also die Modellierung von Interaktionen, von Objektzuständen und von Operationsabläufen, schließt sich dann später an. Dabei ist allerdings zu entscheiden, inwieweit bestehende Ansätze übernommen werden können und welche Komponenten neu zu entwickeln sind. Dieser Analyse widmet sich das nachfolgende Kapitel.

Kapitel 4

VO-Management in Grids: Status Quo

Inhalt des Kapitels

Der in Kapitel 3 aus den Szenarios abgeleitete Anforderungskatalog für VO-Management-Architekturen wurde vor dem Hintergrund des Lebenszyklus Virtueller Organisationen und der rollenspezifischen Perspektiven auf diesen Lebenszyklus gewonnen. Um die Analyse der Ausgangssituation methodisch sauber abzuschließen, besteht der nächste Schritt nun darin, den Status Quo zur Fragestellung zu untersuchen und vorhandene Ansätze auf ihre prinzipielle Tauglichkeit zur Lösung des Gesamtkomplexes oder zumindest einiger Teilaspekte zu überprüfen. Dies geschieht durch eine Spiegelung der Ansätze an den Anwendungsfällen und den nicht-funktionalen Anforderungen, wie sie im Kapitel 3 dargestellt wurden. Eine

spezielle Betrachtung der Aktoren ist nicht notwendig, da diese implizit durch die Anwendungsfälle gegeben sind. In der Diskussion wird sich zeigen, dass kein Ansatz das komplette Spektrum der Anwendungsfälle mit allen geforderten Randbedingungen abdeckt. Dies liegt wesentlich darin begründet, dass das Management des Lebenszyklus Virtueller Organisationen in Grids bisher nicht behandelt wurde. In den folgenden Ausführungen zum Status Quo wird deshalb ein besonderes Augenmerk auf den Aspekt des Lebenszyklus gelegt.

Das Ergebnis des Kapitels wird in Aussagen zur Adäquatheit bestehender Ansätze – oder Teilaspekten davon – für das Management Virtueller Organisationen in Grids bestehen. Diese werden unter konzeptionellen und technischen Gesichtspunkten getroffen. Es sei jedoch betont, dass der Schwerpunkt dieses Kapitels *nicht* darauf liegen wird, die nachstehend diskutierten Ansätze, Standards und Arbeiten *en detail* vorzustellen, dazu wird auf die umfangreich vorhandene Literatur bzw. die Spezifikationen der Standards verwiesen. Vielmehr soll speziell Wert darauf gelegt werden, diejenigen Konzepte zu verdeutlichen, die zur Lösung des VO-Managementproblems in Grids beitragen können. Gleichzeitig ist damit ein Hinweis auf bestehende Lücken verbunden, die es zu schließen gilt.

Natürlich können im Rahmen dieses Kapitels nicht alle Arbeiten diskutiert werden, die sich mit Virtuellen Organisationen in Grids im weitesten Sinn und mit IT-Managementarchitekturen und deren Teilmodellen direkt oder indirekt befassen. Daher ist eine Eingrenzung unbedingt erforderlich. Diese geschieht über die Quelle der betrachteten Arbeiten: Im Abschnitt 4.1 werden bestehende, standardisierte Ansätze mit Bezug zur Fragestellung diskutiert. Daneben werden im Abschnitt 4.2 Forschungs- und Projektarbeiten vorgestellt, die im Umfeld des VO-Managements in Grids – so wie es in dieser Arbeit gesehen wird – von Bedeutung sind. Abschnitt 4.3 führt zum Schluss des Kapitels die Untersuchungen tabellarisch zusammen.

4.1 VO-Management in Standardisierungsgremien

In diesem Abschnitt wird mit den Beiträgen des Open Grid Forums (OGF) zur Open Grid Services Architecture (OGSA) die wesentliche Arbeit in Standardisierungsgremien vorgestellt, die auf das Management Virtueller Organisationen in Grids fokussiert. Zuvor sollen jedoch noch kurz verwandte Ansätze zu allgemeinen Managementarchitekturen (unabhängig von Virtuellen Organisationen) anderer Standardisierungsgremien dargestellt werden, um einerseits deren generelle Konzepte zu identifizieren, gleichzeitig bilden sie aber auch das Fundament, auf dem auch OGSA aufsetzt. Die aus dem OSI-Management bekannte Gliederung des Managements in die vier Teilmodelle (beschrieben in Abschnitt 2.3.1) erweist sich für diese Teilbetrachtungen als zweckmäßig.

OSI/TMN-Managementarchitektur

Das OSI-*Organisationsmodell* unterteilt die an Managementprozessen beteiligten Rollen in *Manager* und *Agenten*. Neben dieser (asymmetrischen) Manager/Agenten-Beziehung kennt OSI ein weitreichendes *Domänen*konzept, in dem Managementobjekte nach organisatorischen oder administrativen Gesichtspunkten gruppiert werden. Da keine Technologiespezifischen Kriterien zur Domänenbildung vorgesehen sind, wird stillschweigend von der Prämisse eines homogenen Umfeldes (hier: OSI) ausgegangen. Diese Annahme ist für Grid-Umgebungen jedoch nicht mehr gültig.

Das OSI-*Informationsmodell* ist objektorientiert angelegt, wobei die Managementobjekte (*managed objects*) reale Ressourcen beschreiben und Managementobjektklassen der MIB instanziieren. Die Beschränkung auf reale Ressourcen ist für Grids allerdings problematisch, da virtuelle Ressourcen nicht beschrieben werden können.

Das OSI-*Kommunikationsmodell* setzt auf dem OSI-Schichtenmodell auf und unterscheidet das schichtübergreifende Management vom Schichtenmanagement und vom Protokollmanagement [Hegering u. a., 1999]. Das schichtübergreifende Management stützt sich dabei auf Dienste (Common Management Information Service (CMIS)) und einem korrespondierenden verbindungsorientierten Protokoll (Common Management Information Protocol (CMIP)).

Das OSI-*Funktionsmodell* strukturiert den Gesamtaufgabenkomplex des technischen Managements in die fünf FCAPS-Funktionsbereiche und spezifiziert mit einer Reihe von generischen Managementfunktionen (System Management Functions (SMF)) ein breites Spektrum von Basisdiensten, die die effiziente Nutzung der Kommunikationsinfrastruktur gewährleisten sollen und häufig benötigte Managementfunktionalitäten bereitstellen. Darüber hinaus bietet das OSI-Management zahlreiche Managementdienste, die flexibel konfigurierbar sind. Hierzu zählen insbesondere Dienste zur Administration von Managementobjekten, deren Zuständen und Beziehungen zu anderen Objekten, aber auch Sicherheits- und Abrechnungsdienste. Die Basis- und Managementdienste eignen sich zwar nicht für das Management Virtueller Organisationen, das Konzept der flexiblen Konfiguration kann aber durchaus übernommen werden.

Mit dem Telecommunication Management Network (TMN) wurde die OSI-Managementarchitektur um eine Referenzarchitektur für ein verteiltes Management von Telekommunikationsnetzen erweitert, um Dienstmanagementaspekte adressieren zu können.

Obwohl die OSI-Managementarchitektur mit ihrer Ausprägung aller vier Teilmodelle als allgemeine Referenzarchitektur für Managementarchitekturen schlechthin angesehen wird, konzentriert sich das OSI-Management auf das Netz- und Systemmanagement. Aspekte des Managements Virtueller Organisationen, wie sie in den Szenarios diskutiert wurden, werden nicht berücksichtigt. Das OSI-Management stellt zwar mit seinem objektorientierten Ansatz essentielle und mächtige Konzepte bereit, ist aber dennoch als Basis zur Beschreibung einer VO-Managementarchitektur wenig geeignet: Durch eine Abstützung

der VO-Managementarchitektur ausschließlich auf das OSI-Management wäre die mit der Heterogenität und Autonomie der lokalen Managementarchitekturen verbundene Problematik des Managements *virtueller* Ressourcen und Dienste in Grids nicht ausreichend berücksichtigt. Trotzdem liefern die im OSI-Funktionsmodell definierten Funktionsbereiche und die Basisdienste ein Gerüst zur Strukturierung und Identifizierung der erforderlichen Funktionalität der VO-Managementarchitektur.

Object Management Architecture mit CORBA

Die Object Management Group (OMG) stellt dagegen mit der Object Management Architecture (OMA) ein Rahmenwerk für die Kooperation von Objekten in offenen, heterogenen Umgebungen bereit. Den Kern bildet dabei die Common Object Request Broker Architecture (CORBA). Die Teilmodelle der OMA sind in Abbildung 4.1 dargestellt. Für weitergehende Informationen verweisen wir auf [Hegering u. a., 1999; Keller, 1998; Siegel, 1996].

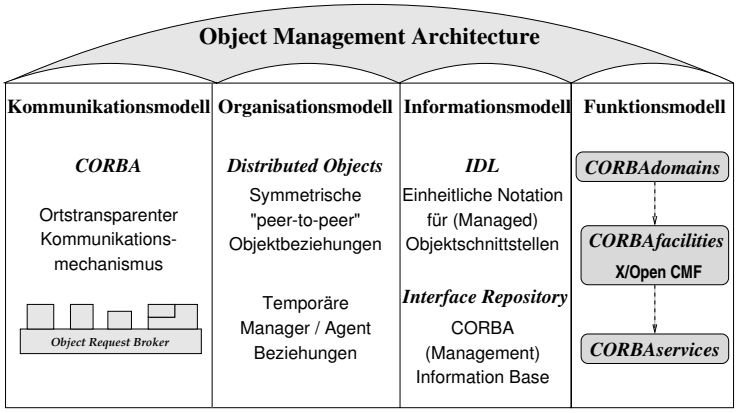


Abbildung 4.1: Teilmodelle der OMA nach [Langer, 2001]

Im CORBA-*Kommunikationsmodell*[1] stellt ein Object Request Broker eine Kommunikationsinstanz für verteilte Objekte dar, indem er Anfragen bzw. Ergebnisse zwischen Objekten weiterleitet. Im *Informationsmodell* wird neben einem übergreifenden Objektmodell mit der Interface Description Language (IDL) eine Syntax zur Beschreibung von Objektschnittstellen definiert. Im *Organisationsmodell*, das von der CORBA Interoperabilitätsarchitektur geprägt ist, wird neben einem Domänenkonzept eine symmetrische Kooperationsform zwischen gleichberechtigten Objekten festgelegt. Das *Funktionsmodell* wird auf der Basis von drei Kategorien definiert: *CORBAservices* liefern elementare Funktionen, um Objekte in verteilten Umgebungen nutzen zu können (beispielsweise die Namensgebung von Objekten, deren persistente Speicherung, Benachrichtigungen über Ereignisse),

[1] Streng genommen handelt es sich um OMA-basiertes Management und daher um das OMA-Kommunikationsmodell. Weil es sich jedoch so eingebürgert hat, wird von den CORBA-Teilmodellen und nicht von den OMA-Teilmodellen gesprochen.

die *CORBAfacilities* stellen universell einsetzbare Dienste für prinzipiell alle Anwendungstypen aus den Bereichen User Interface, Information Management, Systems Management usw. zur Verfügung und die *CORBAdomains* stellen schließlich Dienste aus speziellen Anwendungsbereichen dar.

CORBA unterstützt eine objektorientierte, Plattform- und sprachenunabhängige Informationsmodellierung, die Interaktionen zwischen Objekten in heterogener Umgebung gewährleisten. CORBA bietet zudem mit seinen Services eine Vielzahl von Funktionen an, die auch für das Management Virtueller Organisationen interessant sind. Dennoch ist CORBA als Grundlage zur Beschreibung einer VO-Managementarchitektur nicht geeignet, da die Virtualität des Umfelds nicht gebührend berücksichtigt wird. Zusätzlich basiert CORBA auf einem relativ einfachen Dienstbegriff, der keine Organisationen, Individuen und Rollen als *managed objects* kennt. Aus der Literatur sind zwar Bemühungen bekannt, die Interoperabilität zwischen CORBA und Grids herzustellen [Parashar u. a., 2002], indem Globus-Dienste auf CORBA APIs abgebildet werden, im Ergebnis werden damit jedoch nur CORBAservices für Grid-Anwendungen verfügbar gemacht. Eine allgemeine VO-Managementarchitektur stellt dies nicht dar.

Internet Management Architecture

Der Internet-Managementarchitektur liegt ein asymmetrisches *Organisationsmodell* zugrunde, indem Manager mit Agenten über das Simple Network Management Protocol (SNMP) kommunizieren. Dieses Konzept sieht auch Proxy-Agenten zur Unterstützung nicht-SNMP-konformer Ressourcen vor. Das Internet-*Informationsmodell* definiert zwar Managementobjekte, diesen liegen aber keine objektorientierten Ansätze zugrunde, stattdessen basieren sie auf Mechanismen typisierter Variablen oder Tabellen. Die Managementobjekte werden in Form eines Baumes (Internet-Registrierungsbaum) an den Blättern angeordnet und über eine Template-Sprache definiert [Hegering u. a., 1999]. Die Informationen, die ein SNMP-Agent bereitstellt, werden als Management Information Base (MIB) bezeichnet. Dieser zwar einfache Ansatz des Internet-Informationsmodells ist auch sein entscheidender Nachteil: Durch die fehlende Objektorientierung können bereits definierte Managementobjekte einer anderen MIB nicht wiederverwendet werden. Dies ist aber gerade für das VO-Management in Grids eine wesentliche Anforderung, wie die Beispiele in den Szenarios zeigen. Im Gegensatz zum OSI-Management kommunizieren im Internet-Management Manager und Agenten über ein verbindungsloses Transportsystem [Hegering u. a., 1999] durch den Austausch von Nachrichten. Die Internet-Managementarchitektur kennt kein eigentliches Funktionsmodell. Stattdessen werden komplexe Managementaufgaben auf die SNMP-Manager verlagert, nur die Remote Network Monitoring (RMON) MIB erlaubt eine gewisse Verarbeitung in den Agenten [Hegering u. a., 1999].

Vor dem Hintergrund der Einfachheit des Internet-Managementansatzes ergeben sich nicht unerhebliche Defizite für die Lösung der dieser Arbeit zu Grunde liegenden Fragestellungen. Diese resultieren insbesondere aus dem für das Informationsmodell gewählten datentyporientierten Ansatz und dessen fehlenden Wiederverwendungs-, Verfeinerungs- und

Generalisierungsmechanismen zur Beschreibung von Managementobjekten. Das Organisationsmodell ist zudem für die Modellierung hierarchischer Kooperationsmuster ungeeignet und kennt außerdem kein Domänenkonzept. Beides sind aber Anforderungen an das VO-Management.

Common Information Model (CIM)

Die Zielsetzung des Common Information Models (CIM) der Distributed Management Task Force (DMTF) [DMTF, 2006a] besteht in der Bereitstellung eines übergreifenden Informationsmodells, das eine Einbindung bestehender Modelle unter möglichst minimalen Abbildungsverlusten gestattet. Insofern ist CIM nahe an den Anforderungen an die Informationsmodellierung der hier angestrebten VO-Managementarchitektur. CIM basiert auf einem objektorientierten Ansatz und besteht aus einem Metamodell, einer Syntax zur Beschreibung von Managementobjekten und einigen Top-Level Managementobjektklassen. Das Metamodell wird mit Hilfe von UML definiert und enthält Elemente zur Beschreibung von Klassen, Methoden, Assoziationen und Referenzen. Managementobjektklassen werden über das Managed Object Format (MOF) festgelegt. Auf der Basis des Metamodells werden unter Verwendung von MOF Top-Level Managementobjektklassen im Core Model definiert, die die Basis für die Vererbungshierarchie bilden und generische Klassen und Assoziationen enthalten, die allen Managementfunktionsbereichen zugeordnet werden können. Ein Beispiel ist das `ManagedSystemElement` in Abbildung 4.2

Das Common Model spezialisiert das Core Model um technologieunabhängige Klassen, die als Grundlage für Managementanwendungen dienen. Beispiele sind die Bereiche Anwendung, Endsysteme und Geräte, Netze oder User. Erweiterungsschemata (Extension Schema) verfeinern schließlich diese Modelle für spezifische Technologien.

Die Intention von CIM, Top-Level-Managementobjektklassen zu definieren, stellt eine Grundvoraussetzung für ein integriertes Management dar und ist daher von besonderem Interesse für das VO-Management. Dadurch, dass grundlegende VO-bezogene Konzepte in CIM allerdings nur rudimentär abbildbar sind (Individuen, Rollen, Organisationen, Virtualisierung), liefert CIM keine vollständige Lösungsgrundlage für das VO-Management in Grids, kann aber dennoch wichtige Konzepte für die Beschreibung von VO-bezogenen Managementinformationen beisteuern. Die zunehmende Bedeutung von CIM im Grid-Umfeld wird im übrigen auch durch die Zusammenarbeit der Standardisierungsgremien DMTF und OGF unterstrichen [OGF and DMTF, 2006]. Dennoch sprechen die stetige Veränderung der Modelle und die ausgesprochen rudimentäre Behandlung von Organisations- und Service-Managementaspekten gegen eine direkte Verwendung von CIM im VO-Management.

Shared Information/Data Model (SID)

Das Shared Information/Data Model (SID) des TeleManagement Forums (TMF) [TMF, 2007] ist Bestandteil des Next Generation Operations Support Systems (NGOSS)-Frameworks. Aufgabe von SID ist es, einen Standard für die Verwaltung aller in der Tele-

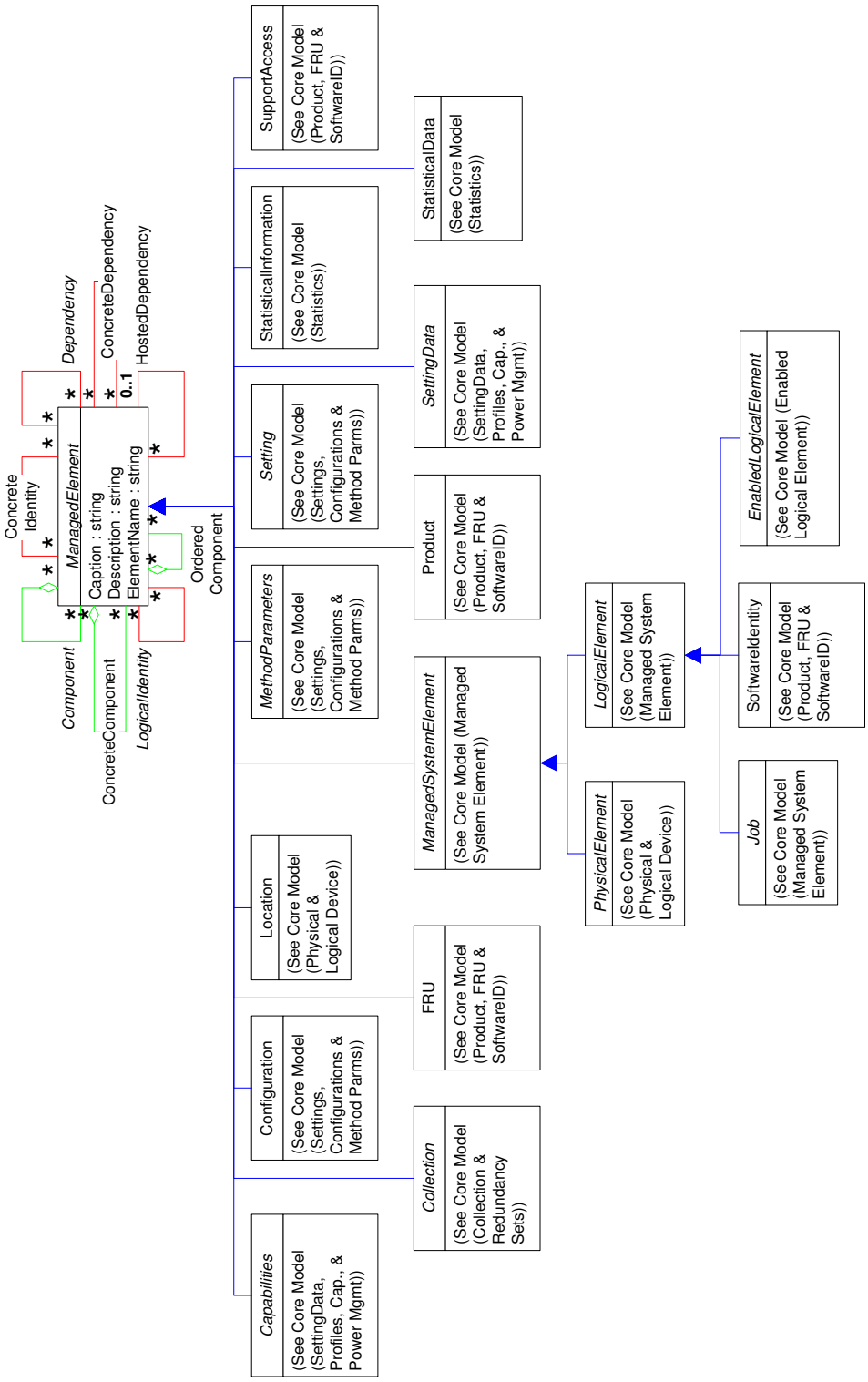


Abbildung 4.2: Übersichtsdiagramm des CIM Core Model nach [DMTF, 2007]

kommunikation notwendigen IT-Entitäten zu schaffen. SID setzt auf den Grundkonzepten des CIM auf, erweitert diese aber nicht unerheblich. SID ist zwar in vielen Bereichen noch rudimentär (es gibt lediglich eine Reihe von Klassendiagrammen, die zudem kaum Methoden oder Attribute enthalten), stellt aber mit seinem im Vergleich zu CIM ausgereifteren Organisationsmodell (Party) ein Konzept zur Verfügung, das den Vorstellungen dieser Arbeit nahe kommt. Insbesondere ist der Pattern-basierte Ansatz des SID-Modells interessant (z.B. die Verwendung des Composite-Patterns [Fowler, 1996]), um Rekursionen zuzulassen und Heterogenität auszugleichen. Abbildung 4.3 zeigt das SID-Party-Konzept im Überblick.

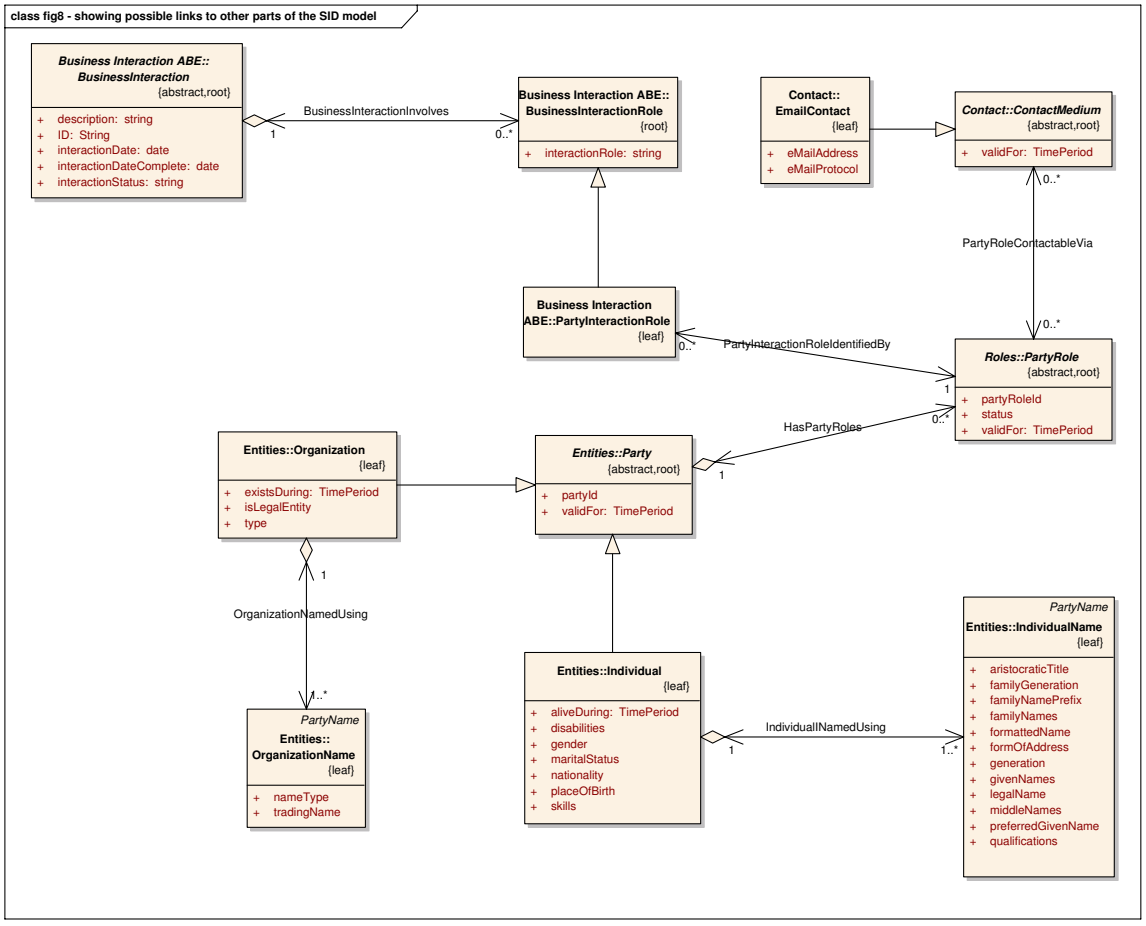


Abbildung 4.3: Party-Konzept des SID nach [TMF, 2007]

Eine Gegenüberstellung von CIM und SID ist in [Brenner u. a., 2006] zu finden. In letzter Zeit wird im übrigen durch eine gemeinsame Initiative der DMTF und der TMF versucht, beide Ansätze zu konsolidieren [TMF and DMTF, 2007]. Obwohl dies prinzipiell dem Anliegen dieser Arbeit entgegenkommt, muss im Einzelfall der Nutzen geprüft werden.

Web Services Distributed Management und korrespondierende Ansätze

In deutlich engerem Zusammenhang mit der Fragestellung dieser Arbeit stehen Managementarchitekturen auf der Basis von Web Services. Es zeigt sich allerdings, dass diese Ansätze keine vollständigen Managementarchitekturen liefern, sondern lediglich Einzelaspekte der Teilmodelle (in der Regel des Kommunikations- und Organisationsmodells) betrachten.

Web Services for System Management (WSM) [Arora u. a., 2005] (spezifiziert von der DMTF) wie auch Web Services Distributed Management (WSDM) [OASIS, 2006d, e, f, g, h] (spezifiziert vom OASIS-Konsortium) wurden für das Management von Service-orientierten Architekturen (SOA) auf der Basis von Web Services konzipiert. Der Fokus der Spezifikationen ist zweifach: Auf der einen Seite wird betrachtet, wie Web Services für die Repräsentation von und den Zugriff auf Managementschnittstellen beliebiger Ressourcen verwendet werden können (Management Using Web Services (MUWS) bzw. WSM) [Arora u. a., 2005; OASIS, 2006e, f]. Auf der anderen Seite wird ein *Manageability*-Modell für Web Services selbst definiert (Management of Web Services (MOWS)) [OASIS, 2006d], in dem der Zugriff auf Ressourcen über *Manageability Endpoints* erfolgt, die durch ein Managementsystem oder eine Web Services-Anwendung genutzt werden können (siehe Abbildung 4.4). Mit Hilfe des Endpoints kann ein *Manageability Consumer* die Ressource durch Kommandos kontrollieren bzw. Notifikationen erhalten. Die Spezifikationen lassen sowohl Agenten- als auch Agenten-lose Konstellationen im Management zu.

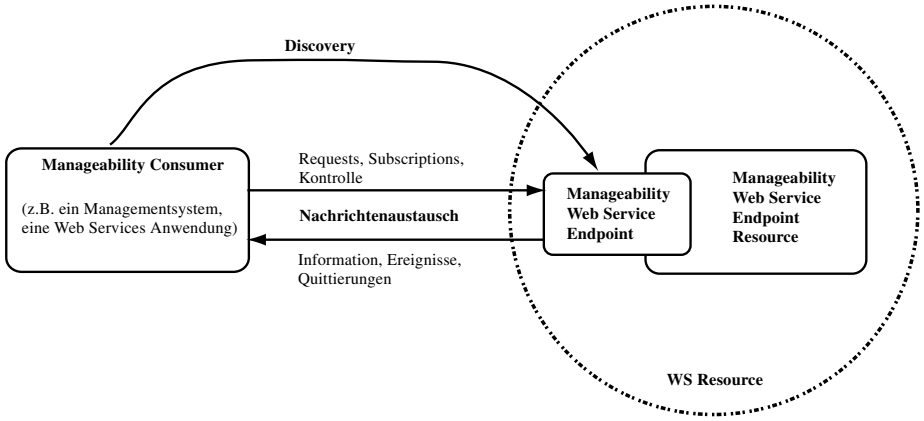


Abbildung 4.4: WSDM-Architektur nach [OASIS, 2006e]

Stellvertretend für die Klasse der Web Services-spezifischen Managementansätze wird im Folgenden der MUWS-Teil der WSDM-Spezifikationen kurz erläutert.

WSDM kennt *Manageability Consumer* und *Manageable Resources* [OASIS, 2006i] (siehe Abbildung 4.4). Manageable Resources können durch mehrere *Manageability Capabilities* charakterisiert sein. Manageability Capabilities definieren die Komponenten einer Manageable Resource, die über eine Managementschnittstelle zugänglich sind (ein Drucker kann

zum Beispiel die Capabilities besitzen, über den aktuellen Tonerstand zu informieren oder den Drucker zu deaktivieren bzw. zu aktivieren). Grundsätzlich sind die Manageability Capabilities erweiterbar. Standardmäßig muss jede Ressource einige Capabilties anbieten. Dazu gehören die Identität, eine Beschreibung, der aktuelle Status, die Verfügbarkeit, die Konfigurierbarkeit und die Beziehungen zu anderen Ressourcen.

Listing 4.1 demonstriert das Erzeugen einer Ressource.

Listing 4.1**:** Beispiel einer Ressource-Erzeugung nach [OASIS, 2006e]

```
<w:definitions
  targetNamespace="http://example.org/services/MyPdaDevice.wsdl"
  ...>

  <w:import namespace="http://docs.oasis-open.org/wsrf/2004/06/
wsrf-WS-ResourceProperties-1.2-draft-01.wsdl"
    location="http://docs.oasis-open.org/wsrf/2004/06/
wsrf-WS-ResourceProperties-1.2-draft-01.wsdl"/>

  <w:types>
    <xs:schema>
      <xs:import namespace="http://example.org/services/MyPdaDevice.xsd"
          schemaLocation="http://example.org/services/MyPdaDevice.xsd"/>

      <xs:import namespace="http://example.org/services/localDefinitions.xsd"
          schemaLocation="http://example.org/services/localDefinitions.xsd"/>

      <xs:import namespace="http://docs.oasis-open.org/wsrf/2004/06/
wsrf-WS-ResourceProperties-1.2-draft-01.xsd"
          schemaLocation="http://docs.oasis-open.org/wsrf/2004/06/
wsrf-WS-ResourceProperties-1.2-draft-01.xsd"/>
    </xs:schema>
  </w:types>

  <w:message name="ToHeader">
    <w:part name="document" element="wsa:To"/>
  </w:message>

  <w:portType name="MyPdaDevicePortType"
      wsrf-rp:ResourceProperties="pda:MyPdaDeviceProperties">
    <w:operation name="GetResourceProperty">
      <w:input  name="GetResourcePropertyRequest"
                message="wsrf-rpw:GetResourcePropertyRequest"
                wsa:Action="http://docs.oasis-open.org/wsrf/2004/06/
wsrf-WS-ResourceProperties-1.2-draft-01.wsdl/
GetResourceProperty/GetResourcePropertyRequest"/>
      <w:output name="GetResourcePropertyResponse"
                message="wsrf-rpw:GetResourcePropertyResponse"
                wsa:Action="http://docs.oasis-open.org/wsrf/2004/06/
wsrf-WS-ResourceProperties-1.2-draft-01.wsdl/
GetResourceProperty/GetResourcePropertyResponse"/>
      <w:fault  name="ResourceUnknownFault"
                message="wsrf-rpw:ResourceUnknownFault"
                wsa:Action="http://docs.oasis-open.org/wsrf/2004/06/
wsrf-WS-ResourceProperties-1.2-draft-01.wsdl/
GetResourceProperty/ResourceUnknownFault"/>
      <w:fault  name="InvalidResourcePropertyQNameFault"
                message="wsrf-rpw:InvalidResourcePropertyQNameFault"
                wsa:Action="http://docs.oasis-open.org/wsrf/2004/06/
wsrf-WS-ResourceProperties-1.2-draft-01.wsdl/
GetResourceProperty/InvalidResourcePropertyQNameFault"/>
    </w:operation>
```

```
  </w:portType>
54
  <w:binding name="MyPdaDeviceSoapOverHttpBinding"
56            type="pdaw:MyPdaDevicePortType">
    <soapw:binding
58        transport="http://schemas.xmlsoap.org/soap/http"
          style="document"/>
60
    <w:operation name="GetResourceProperty">
62    <soapw:operation style="document"/>
      <w:input>
64      <soapw:body use="literal"/>
        <soapw:header message="pdaw:ToHeader" part="document" use="literal"/>
66    </w:input>
      <w:output>
68      <soapw:body use="literal"/>
      </w:output>
70    <w:fault>
        <soapw:fault name="ResourceUnknownFault" use="literal"/>
72    </w:fault>
      <w:fault>
74      <soapw:fault name="InvalidResourcePropertyQNameFault" use="literal"/>
      </w:fault>
76  </w:operation>
  </w:binding>
78
  <w:service name="MyPdaDeviceService">
80  <w:port name="MyPdaDeviceSoapPort"
        binding="pdaw:MyPdaDeviceSoapOverHttpBinding">
82    <soapw:address
          location="http://example.org/services/MyPdaDeviceEndpoint"/>
84  </w:port>
  </w:service>
86
</w:definitions>
```

WSDM setzt auf dem WS-Resource Framework (WSRF) [Czajkowski u. a., 2005] und dessen Teilspezifikationen auf. Dazu gehören die WS-ResourceProperties-Spezifikation [OASIS, 2006l], in der definiert wird, wie Daten, die mit zustandsbehafteten Ressourcen verbunden sind, über Web Services abgefragt und manipuliert werden können (siehe Listing 4.2 für ein Beispiel), die WS-ResourceLifetime-Spezifikation [OASIS, 2006k], in der das planmäßige und außerplanmäßige Beenden einer WS-Resource beschrieben wird und die WS-BaseNotification-Spezifikation [OASIS, 2006b], die die asynchrone Übermittlung von Nachrichten zu Ereignissen festlegt, die über WS-Topics [OASIS, 2006n] „abonniert“ werden.

Listing 4.2**:** Beispiel einer ResourceProperty nach [Czajkowski u. a., 2005]

```
1 <wsdl:portType name="Process"
  wsrp:ResourceProperties="process:ProcessProperties">
3       <wsdl:operation name="findHostingOperationSystem">
        ...
5       <wsdl:operation name="GetResourceProperty">
        <wsdl:input name="GetResourcePropertyRequest"
7               message="wsrpw:GetResourcePropertyRequest" />
        <wsdl:output name="GetResourcePropertyResponse"
9               message="wsrpw:GetResourcePropertyResponse" />
        </wsdl:operation>
```

```
11          <wsdl:operation name="QueryResourceProperties">
            ...
13          </wsdl:operation name="Destroy">
            ...
15  </wsdl:portType>
```

Beide Spezifikationen (WSM und WSDM) sind sich im übrigen sehr ähnlich. Insofern ist es durchaus folgerichtig, beide Ansätze zu konsolidieren. Dies geschieht mit dem Web Services Unified Management Profile (WSUM) [IBM, 2007]. Auf WSUM soll hier allerdings nicht weiter eingegangen werden, da der angestrebte Funktionsumfang zur Zeit noch unklar ist, aber wohl eine Kombination des Umfangs der WSM/WSDM-Spezifikationen darstellen wird.

Der Fokus der Web Services-spezifischen Managementansätze liegt eindeutig auf der Festlegung des Managementzugangs zu Ressourcen. Neben einem Web Services-basierten Kommunikationsmodell wird ein rudimentäres Organisationsmodell definiert, allerdings weder ein Informationsmodell – sieht man einmal von XML als Beschreibungssprache ab – noch ein Funktionsmodell. Insgesamt liefern die Web Services-orientierten Ansätze damit zwar keine komplette Lösung für das Management Virtueller Organisationen, sie stellen allerdings ein Rahmenwerk für eine Plattform-spezifische Definition von VO-Managementanwendungen bereit. Mit der Spezifikation eines WS-CIM-Mappings wird außerdem eine Brücke zum Common Information Model geschlagen, deren erste Ergebnisse jedoch erst in Bruchstücken vorliegen [DMTF, 2006b]. Insofern sind die Anforderungen an eine VO-Managementarchitektur, wie sie im Kapitel 3 formuliert wurden, nur durch die Bereitstellung dieser fehlenden Modelle erfüllbar.

Open Grid Services Architecture (OGSA)

OGSA [Foster u. a., 2006] definiert eine auf Web Services-Technologien aufbauende Basisarchitektur für Grids, in der Dienste über WSDL-Dokumente beschrieben werden. Das grundlegende Konzepte hinter OGSA sind zustandsbehaftete Web Services, wie sie im Web Services Resource Framework (WSRF) [Czajkowski u. a., 2005] spezifiziert werden. Dadurch unterliegen die OGSA-Dienste den gleichen Mechanismen wie die vorher beschriebenen WS-Ressourcen. Sie können also dynamisch instanziiert werden, unterliegen einem Lebenszyklus-Management, erhalten Notifikationen und können insgesamt als Manageable Resource betrachtet werden [Sotomayor u. Childers, 2006]. OGSA definiert eine Reihe von Basisdiensten, die nötig sind, um so genannte *Capabilities* festzulegen. Dazu gehören Infrastrukturdienste, Execution Management Dienste, Datendienste, Resource Management Dienste, Sicherheitsdienste, Selbst-Management-Dienste und Informationsdienste.

OGSA stellt eine Spezifikation dar (Teilrealisierungen liegen mit dem Globus Toolkit (ab Version 3) und UNICORE/GS vor), die allerdings nicht auf das Management Virtueller Organisationen zielt und erst recht keine darauf ausgerichtete Managementarchitektur definiert. Zwar lassen sich prinzipiell VOs aus OGSA-Sicht als WS-Ressourcen betrachten, die dann allerdings notwendigen Konstrukte im Rahmen eines Informationsmodells und

die Prozesse im Kontext eines Organisations- und Funktionsmodells sind nicht beschrieben und bleiben offen. Inwieweit damit diese Ansicht tragfähig ist, ist deshalb fraglich und wird zur Zeit eingehend untersucht [Cojocaru, 2007].

Zu konstatieren bleibt, dass zwar eine Reihe von Konzepten für das Management von Grid-Ressourcen [Buyya u. a., 2000; Krauter u. a., 2002; Maciel, 2005; Nabrzyski u. a., 2004] existieren, eine Managementarchitektur im Sinne der Fragestellung wird dadurch jedoch nicht definiert, da keine Konzepte, Virtuelle Organisationen als *managed objects* zu behandeln, zur Verfügung gestellt werden. Die Bemühungen, zumindest ein allgemeines Grid-Informationsmodell zu spezifizieren, stehen erst am Anfang. Hier sind mit dem Grid Laboratory Uniform Environment (GLUE)-Schema [Andreozzi u. a., 2007; Strong, 2007] erste wichtige Schritte unternommen worden, die allerdings noch keine Berücksichtigung von VOs als solche zulassen. Mit dem in GLUE vorgesehenen Site-Konzept (siehe Abbildung 4.5) als administrative Domäne lassen sich jedoch durchaus Teilaspekte des VO-Managements umsetzen. In die gleiche Kategorie fallen auch die Bestrebungen, CIM als Informationsmodell Grid-weit zu nutzen [Miura, 2006]. Aber auch hier ist festzuhalten, dass es sich dabei zunächst nur um Versuche handelt, physische und virtuelle Ressourcen zu beschreiben (in erster Linie Grid-Jobs), nicht jedoch Individuen, Rollen, Mitgliedschaften oder gar VOs selbst.

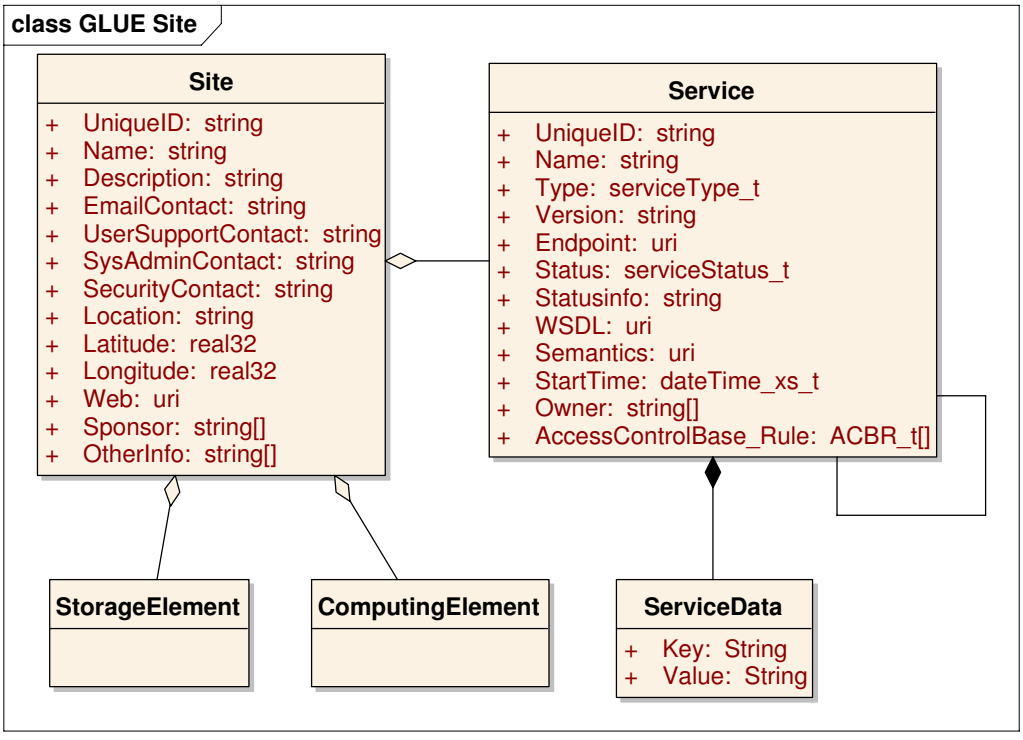


Abbildung 4.5: GLUE Site-Konzept nach [Andreozzi u. a., 2007]

Ein Grid-spezifisches Funktionsmodell zum VO-Management existiert ebenfalls nicht.

Auf Grund der Historie des Grid Computing und dem derzeitigen Fokus konzentrieren sich die Grid-spezifischen Ansätze stattdessen schwerpunktmäßig auf sehr eingeschränkte Funktionsbereiche (hauptsächlich im Security-Bereich), die allerdings auf den verwendeten Plattformen im Sinne einer Grid-Middleware unterschiedlich stark ausgeprägt sind. Sie fallen damit eher in den Bereich des operativen Grid-Managements, zu dem das VO-Management dieser Arbeit Schnittstellen bereitstellt bzw. nutzt.

Aus diesen Ausführungen und der umfangreichen Literatur zu allgemeinen Managementarchitekturen (siehe beispielsweise [Dreo Rodošek, 2002; Hegering u. a., 1999; Keller, 1998; Langer, 2001; Nerb, 2001; Pavlou, 1998; Radisic, 2003] und die dort zitierten Referenzen) leiten sich zwei zusammenfassende Erkenntnisse ab:

1. Keiner der Ansätze ist auf das Management Virtueller Organisationen in Grids ausgerichtet.
2. Dennoch lassen sich aus einigen Ansätzen grundsätzliche Konzepte ableiten, die für das Management Virtueller Organisationen in Grids übernommen werden können.

4.2 VO-Management in Forschungsvorhaben und Projekten

In sehr engem Zusammenhang mit der Fragestellung dieser Arbeit stehen VO-Managementkonzepte, die Grid-spezifisch sind und im Rahmen von Forschungsprojekten durchgeführt werden. Wegen des noch jungen Gebietes sind unterstützende Architekturen (im Sinne der Teilmodelle) komplett nicht vorhanden. Die aktuellen Forschungschwerpunkte sind stattdessen in den Bereichen „Accounting“ (siehe hier zum Beispiel [Göhner u. a., 2006]), primär aber „Authentifizierung und Autorisierung“ zu finden. Während die Accounting-Ansätze das VO-Management nur indirekt betreffen, ist die Verbindung zur Authentifizierung und Autorisierung enger. Insofern wird in den folgenden Abschnitten dieser Aspekt näher betrachtet. Dabei folgen wir im wesentlichen den Ausführungen in [Grimm u. Pattloch, 2006].

4.2.1 Authentifizierung und Autorisierung

Eine Authentication & Authorization Infrastructure (AAI) stellt einen unabdingbaren und komplexen Bestandteil jeder Grid-Infrastruktur dar, über die sich Grid-Ressourcen, Benutzer und Virtuelle Organisationen gegenseitig in Abhängigkeit ihrer jeweiligen Policies verifizieren. Hierzu werden in der Regel verteilte Beschreibungen von Berechtigungen (Attribute) und Identitäten (Zertifikate) genutzt. Eine AAI stellt sowohl Zertifizierungs- als auch Verzeichnisdienste bereit und sieht spezifische Protokolle für den Zugriff auf diese Dienste vor.

Authentifizierung ist das Beweisen einer Identität. Bei der in der Informatik klassischen Authentifizierungsmethode, nämlich der Authentifizierung durch Angabe eines Benutzernamens und eines korrespondierenden Kennworts, folgt die eigentliche Authentifizierung aus der Kenntnis des geheimen Passwortes. In Grid-Infrastrukturen werden für die Authentifizierung Public Key-Verfahren (PKI) auf der Basis von X.509-Zertifikaten eingesetzt [Barton u. a., 2006; Foster u. a., 2003; Foster u. Childers, 2005; Foster u. Kesselman, 2004b; Grimm u. Pattloch, 2006]. Durch Erweiterungen dieses Verfahren um Proxy-Zertifikate werden Ansätze für ein Single Sign-On (SSO) [Hommel, 2007] bereitgestellt.

Das SSO-Konzept ermöglicht es einem Benutzer, die ihn authentifizierenden Informationen bei jeder Arbeitssitzung nur ein einziges Mal zur Verfügung stellen zu müssen. Im einfachsten Fall wird diese relevante Information vom benutzten Client zwischengespeichert. Bei X.509 kann nun ein Proxy-Zertifikat erstellt werden, das nicht durch ein Kennwort geschützt ist, aber auch nur kurzzeitig gültig ist. Dieses Proxy-Zertifikat kann durch die Zertifizierung mit dem eigentlichen Benutzerzertifikat stellvertretend zur Authentifizierung genutzt werden. Wegen ihrer Kurzlebigkeit müssen Proxy-Zertifikate erneuerbar sein, insbesonderen dann, wenn Transaktionen einen längeren Zeitraum benötigen, als die Proxy-Zertifikate gültig sind. Hierzu steht mit MyProxy ein entsprechender Mechanismus zur Verfügung, der seit 2005 fester Bestandteil des Globus Toolkit 4 ist [Novotny u. a., 2001].

Mit dem PKI-Verfahren werden die öffentlichen Schlüssel, die für jeden Benutzer existieren, vor unbefugter Manipulation geschützt. Um sicherzustellen, dass ein öffentlicher Schlüssel nicht verändert wurde, werden vertrauenswürdige Instanzen, die Certificate Authorities (CA), benötigt. Die CAs verfügen selbst über ein Schlüsselpaar, mit dem der öffentliche Schlüssel eines Benutzers signiert wird. Das so erstellte Zertifikat des Benutzers kann dann verteilt werden. Ein Server kann davon ausgehen, dass der in dem Zertifikat enthaltene öffentliche Schlüssel tatsächlich dem Benutzer gehört, der im Zertifikat angegeben ist. Voraussetzung hierfür ist, dass der Server der ausstellenden CA vertraut und deren öffentlichen Schlüssel kennt. Mit einem zertifizierten Schlüssel können weitere Zertifikate ausgestellt werden, wodurch eine Zertifizierungshierarchie entsteht. Die erstellten Zertifikate sind zeitlich begrenzt ausgestellt. Zusätzlich besteht die Möglichkeit, ein Zertifikat für ungültig zu erklären, indem es auf eine definierte „schwarze Liste“ von Zertifikaten, der so genannten Certificate Revocation List (CRL), aufgenommen wird. Jede Stelle, die eine Authentifizierung aufgrund von Zertifikaten implementiert, muss die zur Verfügung stehenden CRLs aktuell halten und dagegen prüfen.

Die Authentifizierung in Grids findet zur Zeit ausschließlich über Zertifikate gemäß X.509 statt. Um den Betrieb einer VO zu gewährleisten, müssen daher die Zertifizierungsstellen (CAs) von der European Grid Policy Management Authority (EUGridPMA) akzeptiert werden (siehe auch Abschnitt 3.1.1)[2].

[2]2005 haben sich EUGridPMA und The Americas Grid Policy Management Authority (TAGPMA) sowie die Asia Pacific Grid Policy Management Authority (APGridPMA) zur International Grid Trust Federation (IGTF) [Groep, 2005] zusammengeschlossen.

Das Ziel der Autorisierung bzw. Zugriffskontrolle ist es, Aktionen und Operationen von Benutzern so zu beschränken, dass nicht gegen Sicherheitsrichtlinien verstoßen werden kann, die in (realen, aber auch virtuellen) Organisationen gelten. Dazu gehört insbesondere, Prozesse gemäß vorgegebener Benutzerrechte einzuschränken oder Benutzern Rechte zu verschiedenen Bereichen eines Systems zu gewähren oder zu verwehren. Es gibt inzwischen eine Vielzahl von VO-Autorisierungssystemen mit jeweils unterschiedlichem Fokus. Manche konzentrieren sich auf VO-Mitgliedschaften (z.B VOMS), andere – wie der Community Authorization Service (CAS) [Sotomayor u. Childers, 2006] – adressieren die Verwaltung von Mitgliedern *und* Ressourcen. Stellvertretend für diese Ansätze soll der Virtual Organization Membership Service (VOMS) [Alfieri u. a., 2003] kurz angerissen werden.

VOMS dient der Verwaltung der Mitglieder einer Virtuellen Organisation und von Informationen über den Status eines Benutzers. Die Eigenschaften des Benutzers in einer VO werden über die Zugehörigkeit zu Gruppen sowie der Zuordnung von Rollen und Capabilities definiert. Diese Informationen werden in einem „Pseudo-Zertifikat“ abgelegt, das im Gegensatz zu einem X.509-Zertifikat keine Schlüssel enthält, aber vom VOMS-Server mit dem privaten Schlüssel seines X.509-Host-Zertifikats signiert wird. Das Pseudo-Zertifikat integriert der Benutzer als nichtkritische, private Zertifikatserweiterung in sein Proxy-Zertifikat [Grimm u. Pattloch, 2006].

Ein Proxy-Zertifikat kann prinzipiell mehrere Attribut-Zertifikate von verschiedenen Virtuellen Organisationen enthalten. Damit kann sich ein Benutzer zu verschiedenen VOs anmelden und mehrere VO-Attribute nutzen. Dem Benutzer ist es zudem möglich, Untermengen seiner Attribute zu definieren, die in einem Proxy-Zertifikat abgelegt werden sollen. Dies kann sinnvoll sein, wenn er dem zu erstellenden Proxy-Zertifikat nicht alle Rechte übertragen möchte, über die er verfügt, eine wesentliche Voraussetzung für adäquate Privacy-Konzepte.

Im Rahmen des gLite-Projektes [LCG, 2005] wurde der VOMS-Ansatz mit dem Local Centre Authorization Service (LCAS) um Policy Decision Points (PDP) und dem Local Credential Mapping Service (LCMAPS) für die Abbildung von Grid Credentials auf lokale Accounts kombiniert [Ferro u. a., 2005].

4.2.2 VO-Management mit Shibboleth und GridShib

Um den Umfang dieser Arbeit nicht zu sprengen, orientieren sich die die folgenden Ausführungen stark an den Analysen in [Grimm u. Pattloch, 2006] und [Gietz u. a., 2007]. Für weitergehende Informationen sei deshalb auch auf diese Quellen verwiesen.

Shibboleth wurde im Rahmen des Middleware Architecture Committee for Education (MACE)-Projektes des Internet2 Konsortiums [Watt u. a., 2006] entwickelt. Unter Shibboleth wird jedoch landläufig die Open-Source-Implementierung eines verteilten Systems zur Nutzung von zugangsgeschützten Web-Ressourcen im Rahmen von Föderationen verstanden. Im Bibliotheks- und eLearning-Bereich findet das System international eine stark zunehmende Verbreitung [Gietz u. a., 2007]. Im Grid-Umfeld wird Shibboleth zunehmend

als mögliche Ergänzung bzw. Ersatz für PKI-Strukturen angesehen [Grimm u. Pattloch, 2006], wie beispielsweise das GridShib-Projekt [Barton u. a., 2006; D.W.Chadwick u. a., 2006; Gemmill u. Robinson, 2006] zeigt. Shibboleth denkt in Transaktionen zwischen Identity Providern (IdP) und Service Providern. Der IdP verwendet dabei die Identity Management (IdM)-Verfahren der Heimateinrichtung des Nutzers, der SP repräsentiert eine Web-Ressource. IdPs und SPs bilden typischerweise Föderationen, deren Policies die Vertrauensbasis der teilnehmenden Partner bildet [Hommel u. Reiser, 2005]. In den Policies wird festgelegt, welches Attributschema verwendet wird. Als internationaler Quasi-Standard gilt das *eduPerson*-Schema [Internet2, 2003], das gegebenenfalls durch nationale Erweiterungen ergänzt wird. Shibboleth basiert auf der Security Assertion Markup Language (SAML) [OASIS, 2005], um Zusicherungen (*assertions*) über die Autorisierung eines Nutzers vom IdP an einen SP zu übertragen. SAML wird in Shibboleth, in GridShib, im Globus Toolkit 4 und im Rahmen der OGSA-Entwicklung für Autorisierungszwecke genutzt.

Das Ziel des GridShib-Projektes [Barton u. a., 2006; D.W.Chadwick u. a., 2006; Squicciarini u. a., 2007] ist die Interoperabilität zwischen dem Globus Toolkit und Shibboleth. Der Fokus liegt dabei insbesondere auf der Erweiterung des Globus Toolkits um Mechanismen zur Interpretation und Verwendung von SAML-Assertions im Rahmen des Globus Autorisierungs-Frameworks. Für ein adäquates Management Virtueller Organisationen kooperieren die GridShib- und myVocs-Projekte [Gemmill u. Robinson, 2006], um Identitätsattribute Virtueller Organisationen in föderierten Shibboleth-Umgebungen verwenden zu können (siehe [Gietz u. a., 2007] für eine gründliche myVocs-Analyse).

In Shibboleth kommt das Modell Attribut-basierter Zugriffskontrolle [Squicciarini u. a., 2007] zum Einsatz. Shibboleth nutzt dabei möglicherweise schon vorhandene SSO-Systeme als Authentifizierungskomponenten. Autorisierungen werden in standardisierten Attributen (z.B. eduPerson) im IdM abgelegt und in Form einer SAML Assertion an den SP übertragen. Shibboleth basiert auf drei zentralen Komponenten (siehe auch Abbildung 4.6):

- Jede Einrichtung, die Mitglied einer Föderation wird, benötigt einen IdP. Der IdP besteht primär aus der Attribute Authority (AA), dem Handle Service (HS), dem institutionellen IdM (Directory Server oder Datenbank) und dem lokalen SSO. Während AA und HS Bestandteil von Shibboleth sind, muss der IdP-Betreiber die IdM- und SSO-Komponenten bereitstellen.
- Jede Einrichtung, die einen Dienst in einer Föderation bereitstellt, benötigt einen SP. Die wichtigsten Komponenten des SP sind der Assertion Consumer Service (ACS), der Attribute Requester (AR) und der Resource Manager (RM).
- Der Where Are You From (WAYF)-Dienst repräsentiert die zentrale Stelle einer Föderation, die alle teilnehmenden IdPs und deren Adressen kennt. Er gestattet es einem Nutzer, seinen IdP zu identifizieren.

In [Grimm u. Pattloch, 2006] wird auf eine zunehmende Tendenz zur Grid-„Shibbolisierung“ hingewiesen. Darunter wird eine umfassende Integration von Shibboleth und

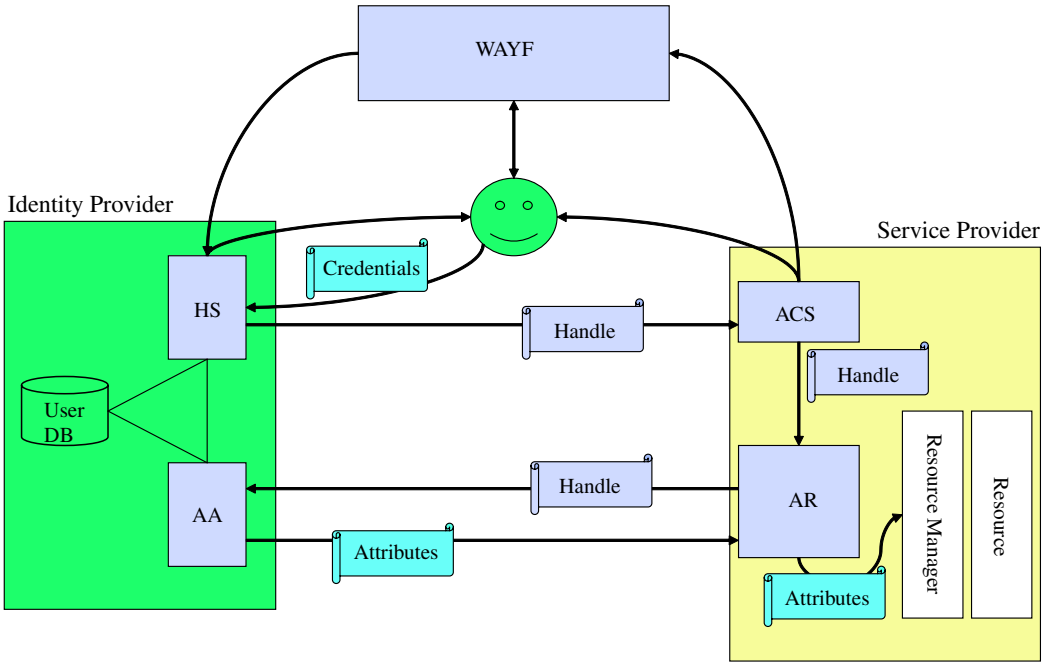


Abbildung 4.6: Shibboleth-Architektur nach [Gietz u. a., 2007]

Grids zur Authentifizierung verstanden. Typische Forschungsprojekte in diesem Kontext sind GridShib [Barton u. a., 2006; D.W.Chadwick u. a., 2006; Squicciarini u. a., 2007], das Meta-Access Management System (MAMS)-Projekt [Lin u. a., 2006], die Projekte Shibboleth Enabled Bridge to Access the National Grid Service (SHEBANGS)[3] und ShibGrid[4].

Im Shibboleth-Konzept bildet der IdP der Heimateinrichtung eines Nutzers die Quelle der Autorisierung (Attribute Authority). Eine VO ist aus dieser Sicht eine weitere, eigene Quelle der Autorisierung. Da Shibboleth eine Attribute Authority jedoch nur als Bestandteil eines IdP vorsieht, können VOs mit Shibboleth-Mitteln nur mit folgenden Ansätzen verwaltet werden:

1. Die VO-Verwaltung wird auf die IdPs der zugehörigen Mitglieder verteilt und die an der VO beteiligten Institutionen (die realen Organisationen) einigen sich auf ein einheitliches Schema mit seinen Attributen.

2. Die VO wird durch einen eigenen IdP realisiert, was dem VOMS-Modell mit seinen Vor- und Nachteilen entsprechen würde.

3. Eine Person und ihre Attribute werden im Identity Management (IdM) der Heimateinrichtung (IdP) verwaltet, die VO-spezifischen Attribute aber in der VO. Damit werden für die VO Zugangsrechte auf das IdM der IdPs erforderlich. Für dieses Szenario geeignete Werkzeuge werden mit *Grouper* und *Signet* im Rahmen des Internet2 Middleware-Projektes entwickelt [Barton u. McRae, 2005].

[3] `http://www.jisc.ac.uk/whatwedo/programmes/programme_middleware/proj_shebangs.aspx`
[4] `http://www.jisc.ac.uk/whatwedo/programmes/programme_middleware/proj_shibgrid.aspx`

4.3 Zusammenfassende Bewertung

Dieses Kapitel hat gezeigt, dass in Standardisierungsgremien als auch in Forschungsprojekten Anstrengungen unternommen werden, Virtuelle Organisationen einem formaleren Managementprozess zuzuführen. Dieser ist jedoch in der Regel darauf beschränkt, die in Grids typischen Kooperationen und Koordinationen „abzusichern“ und den Zugriff auf Ressourcen und Dienste geeigneten Authentifizierungs- und Autorisierungsverfahren zu überlassen. Das VO-Management-Verständnis reduziert sich damit im wesentlichen auf die Unterstützung der Betriebsphase des Lebenszyklus. In vielen Projekten werden jeweils spezifische Teilfragestellungen untersucht, die für das Management Virtueller Organisationen in Grids in dedizierten Einzelaspekten von Belang sind, VOs als *managed objects* werden dadurch jedoch nicht behandelt. Insofern sind zur Zeit auch keine Ansätze für eine formale Managementarchitektur (im Sinne [Hegering u. a., 1999]) zu erkennen. Damit ist von allen betrachteten Ansätzen keiner in der Lage, die Anwendungsfälle und nicht-funktionalen Anforderungen des Kapitels 3 zu erüllen.

Tabelle 4.2 zeigt dies noch einmal im Überblick unter Verwendung der Tabelle 4.1 als Legende.

Symbol	Bedeutung
○	Das Kriterium wird nicht abgedeckt.
◔	Das Kriterium wird teilweise abgedeckt.
●	Das Kriterium wird vollständig abgedeckt.

Tabelle 4.1: Legende zur Einordnung bestehender Ansätze

Weder werden Virtuelle Organisationen als *managed objects* betrachtet, noch werden Lebenszyklen Virtueller Organisationen vollständig abgedeckt, noch sind die bestehenden Managementarchitekturen organisationsorientiert, allenfalls sind sie Ressourcen- oder Dienst-orientiert. Das Konzept eines „virtualisierten“ Managements wird nirgendwo realisiert. Ein methodisch sauberer und systematischer Ansatz zur Konzeption und Anwendung einer VO-Managementarchitektur ist daher zwingend erforderlich. Dieser wird in den nächsten Kapiteln erarbeitet.

	Ansatz						
Anwendungsfall	OSI, CORBA, Internet	CIM, SID	WSDM	OGSA	Shibboleth & Co.	VOMS & Co.	Globus & Co.
F01: Initiieren einer VO-Gründung	○	○	○	○	◉	◉	○
F02: Initialisieren einer VO	○	○	○	○	◉	◉	◉
F03: Start einer VO	○	○	○	○	○	○	◉
B01: Hinzufügen von Mitgliedern	○	○	◉	◉	◉	●	◉
B02: Löschen von Mitgliedern	○	○	◉	◉	◉	●	◉
B03: Ändern von Mitgliedschaften	○	○	◉	◉	●	●	◉
B04: Hinzufügen von Rollen	○	◉	◉	◉	◉	●	◉
B05: Löschen von Rollen	○	◉	◉	◉	◉	●	◉
B06: Ändern von Rollen	○	◉	◉	◉	◉	●	◉
B07: Aufnahme von Ressourcen/Diensten	◉	◉	◉	◉	○	○	◉
B08: Löschen von Ressourcen/Diensten	◉	◉	◉	◉	○	○	◉
T01: Archivieren von VOs	○	○	○	○	○	○	◉
T02: Kündigen von SLAs	◉	○	◉	◉	○	○	◉
T03: Stoppen einer VO	○	○	◉	◉	○	○	◉
T04: Löschen einer VO	○	○	◉	◉	○	◉	◉
Informationsmodell	●	●	○	○	○	◉	○
Organisationsmodell	●	◉	◉	◉	◉	◉	◉
Kommunikationsmodell	●	○	◉	○	○	◉	◉
Funktionsmodell	●	○	◉	○	◉	◉	◉
Plattform-unabhängig	●	●	○	●	◉	◉	○

Tabelle 4.2: Einordnung bestehender Ansätze zum Management Virtueller Organisationen in Grids

Kapitel 5

Entwicklung einer Architektur für das VO-Management in Grids

Inhalt des Kapitels

Im Kapitel 3 wurden die Anforderungen an eine VO-Managementarchitektur (VOMA) aus den Szenarios und den allgemeinen Betrachtungen des Kapitels 2 abgeleitet. Anschließend wurde im Kapitel 4 dargestellt, inwieweit bestehende Architekturansätze dem Management Virtueller Organisationen in Grids zugänglich sind, zumindest in Teilen. Dabei zeigte sich

(siehe auch Tabelle 4.2), dass zur Lösung des VO-Managementproblems kein vollständiges Rahmenwerk existiert, obwohl einzelne Aspekte des Problems durchaus adressiert werden können. Diese Lücke wird in diesem Kapitel mit einer systematischen Umsetzung der Anforderungen im Kapitel 3 zum ersten Mal geschlossen.

Als Managementarchitektur beschreibt VOMA ein *Rahmenwerk*, das keine Implementierung antizipiert, also plattformunabhängig ist. Um VO-Managementlösungen auf einem Trägersystem implementieren zu können, muss sich methodisch deshalb eine Phase anschließen, in der VOMA *plattformspezifisch* umgesetzt wird. Dieser Schritt folgt später im Kapitel 6 mit der Einbettung von VOMA in Grid- bzw. Web Services-Rahmenwerke wie WSRF und WSDM. Eine Anwendung der durch VOMA induzierten Methodik wird schließlich im Kapitel 7 demonstriert.

Mit VOMA wird in Analogie zu bestehenden Managementarchitekturen das Ziel verfolgt, die Basis für ein Grid-weites VO-Management durch die Festlegung der erforderlichen Teilmodelle zu schaffen. Wie die Szenarios im Kapitel 3 gezeigt haben, liegt eine der Hauptschwierigkeiten darin, dass das Management Virtueller Organisationen eine mehrstufige Virtualisierung von Managementarchitekturen verlangt, die einerseits die Betrachtung Virtueller Organisationen als *managed objects* unterstützt, andererseits aber eine „virtuelle Managementarchitektur“ konzipiert werden muss, um dem Co-Managementproblem gerecht zu werden. Dies folgt zum einen aus dem Managementfokus selbst, der dynamische *Virtuelle* Organisationen in den Vordergrund stellt, zum anderen aus der Anforderung, VO-Management und lokales Ressourcen- bzw. Dienstmanagement getrennt zu betrachten. Der virtuelle Charakter der VO-Managementarchitektur wird weiterhin dadurch unterstrichen, dass das VOMA-Informationsmodell eine logische Sicht auf vorhandene Informationsmodelle lokaler Managementarchitekturen ermöglichen muss, dass das Kommunikationsmodell an SOA-Konzepte gekoppelt ist, dass das Organisationsmodell ohnehin rollenorientiert ausgerichtet ist und dass das Funktionsmodell auf Objekten mit virtuellem Charakter operiert. Daraus ergeben sich einige Konsequenzen für den Entwurf und den Entwurfsprozess der Architektur:

- Der Entwurf muss auf eine Abstraktionsebene gehoben werden, die eine einfache Abbildung der VOMA-Konzepte auf Konzepte der lokalen Managementarchitekturen gestattet und eine aggregierte Sicht auf alle an VO-Managementprozessen beteiligten Entitäten erlaubt.
- Die VOMA-Teilmodelle sollten bestehende Konzepte, Regeln und Modellierungsansätze möglichst integrieren. Es ist nicht das Ziel, diese zu ersetzen. Daher muss für jedes Teilmodell *separat* entschieden werden können, welche Prinzipien genutzt werden können und sollen und welche nicht. Dies betrifft jedoch nicht nur die Frage der Wiederverwendbarkeit von Modellierungskonzepten und -werkzeugen, sondern insbesondere auch die Integrierbarkeit von und Interoperabilität mit bestehenden Modellen (eine Frage, der sich auch [Ziegler u. Grimm, 2006] widmet).
- Grid-Infrastrukturen werden zunehmend auf Nachhaltigkeit ausgerichtet [Gentzsch,

2005]. VO-Managementarchitekturen müssen daher technologie-agnostisch konzipiert werden. Die eigentlichen Architekturkonzepte (*Was* soll realisiert werden?) werden sich nämlich in der Regel wenig ändern und oftmals über Jahre oder sogar Jahrzehnte hinweg konstant bleiben (z.B. Transaktionskonzepte oder Modelle), die Realisierung (*Wie* soll etwas realisiert werden?) hingegen muss sich mit a priori nicht vorhersagbaren Technologiesprüngen auseinander setzen.

Trotz des generischen Ansatzes, soll VOMA (als Metamodell) jedoch keinesfalls den Anspruch eines allgemeinen Organisations-Metamodells erheben, sondern nur diejenigen Informationen und Funktionen beschreiben, die für das Management Virtueller Organisationen notwendig sind. Deshalb erfolgt der Entwurf der Architektur modellgetrieben, indem ein plattformunabhängiges Modell möglichst automatisch auf spezifische Plattformen – hier die Grid- bzw. Web Services Plattformen – abgebildet wird. Erreicht wird dadurch eine Trennung von Modellen und deren Implementierungen.

Abschnitt 5.1 stellt die prinzipiellen Aspekte dieser Vorgehensweise noch einmal kurz zusammen, die in den sich anschließenden Abschnitten auf den VOMA-Entwurf angewandt werden. Im Teilabschnitt 5.2 geschieht dies für das Informationsmodell, im Teilabschnitt 5.3 für das Organisationsmodell, im Teilabschnitt 5.4 für das Kommunikationsmodell, und schließlich im Teilabschnitt 5.5 für das Funktionsmodell. Abschnitt 5.6 fasst die Ergebnisse dieses Kapitels dann noch einmal kurz zusammen.

5.1 MDA-basierte Entwurfsmethodik

In diesem Abschnitt wird zunächst eine kurze Einführung in grundlegende Konzepte des Model Driven Architecture-Ansatzes gegeben. Anschließend wird die mit diesen Konzepten zusammenhängende Entwicklungsmethodik im Überblick vorgestellt.

5.1.1 Allgemeiner Überblick

Model Driven Architecture (MDA) bezeichnet eine Entwicklungsstrategie der Object Management Group (OMG)[1] zur modellbasierten Softwareentwicklung [Frankel, 2003; Kleppe u. a., 2003; Raistrick u. a., 2004]. Die grundlegende Idee des Ansatzes ist es, zur Erstellung eines Softwaresystems unterschiedlich konkrete Modelle des zu erstellenden Systems zu spezifizieren, bis schließlich am Ende des Entwicklungsprozesses die Implementierung des Systems aus den Modellen weitgehend automatisiert abgeleitet werden kann. Zwischen den Modellen dienen Modelltransformationen dazu, möglichst große Teile eines Zielmodells aus dem Quellmodell zu erzeugen. Das Ziel, die Wiederverwendbarkeit von Modellen – und damit des Fachwissens – zu gewährleisten, wird durch eine strikte Trennung der fachlichen und

[1] `http://www.omg.org/`

technischen Aspekte in Form von Plattform-unabhängigen Modellen (PIM) und Plattform-spezifischen Modellen (PSM) angestrebt. PIMs und PSMs bilden mit ihren Transformationen (Mappings) gerichtete kreisfreie Graphen, da – je nach Anforderungen – PIMs in PSMs transformiert werden, die wiederum als PIMs aufgefasst werden können, um sie weiter zu verfeinern. Generell wird im MDA-Ansatz eine Plattform zunächst einfach als Menge von Architekturen, Technologien und Funktionalitäten verstanden [Raistrick u. a., 2004]. Erst Plattformen kennen damit die technischen Details, die für eine spezifische Modellierung notwendig sind, für eine rein logische Beschreibung der Anwendungsfunktionalität jedoch unwichtig sind.

Beispiel: Eine typische Plattform wird von der Programmiersprache Java und der Enterprise Java Beans (EJB)-Spezifikation mit einem bestimmten EJB-Container gebildet [Frankel, 2003].

Als Modellierungssprache wird im MDA-Ansatz die Unified Modeling Language (UML) mit ihren Metamodellierungs- und Profilmechanismen verwendet. Ein UML-Profil stellt einen Standardmechanismus zur Erweiterung des Sprachumfangs der UML dar, um UML so an spezifische Einsatzbedingungen (z.B. solche Domänen, wie sie in dieser Arbeit betrachtet werden) anpassen zu können. Analog zu formalen Programmiersprachen als Grundlage für Compiler, unterstützen UML-Profile durch die Angabe von Stereotypen, Tagged Values und Constraints die Automatisierbarkeit von Modelltransformationen. Constraints werden dabei durch Konstrukte der Object Constraint Language (OCL) oder ähnliche Mechanismen ausgedrückt [OMG, 2003]. Jeder OCL-Ausdruck resultiert in einem Wert und definiert, *was* dieser Wert sein soll, nicht jedoch, *wie* der Wert berechnet wird. Abbildung 5.1 zeigt an einem einfachen Beispiel das Prinzip der Transformation eines PIMs in ein PSM und schließlich in den generierten Code.

UML-Profile sind auf der UML-Metamodellebene angesiedelt [Frankel, 2003]. Zur Zeit existieren eine Reihe von vordefinierten Profilen: Profile für die Transformation eines PIMs in ein CORBA spezifisches PSM [OMG, 2002], Profile für die Transformation eines PIMs in ein Java/EJB spezifisches PSM [OMG, 2004a] und Profile für lose gekoppelte Systeme, welche asynchron oder nachrichtenbasiert kommunizieren [OMG, 2004b]. Für IT-Managementaspekte existieren jedoch noch keine dedizierten Profile, sieht man einmal von der Darstellung von QoS- und Fehlertoleranzkonzepten ab [OMG, 2006]. Diese Lücke wird im Rahmen dieser Arbeit zumindest in Teilbereichen geschlossen.

Beispiel: Abbildung 5.1 zeigt zwei Klassen, die unter Verwendung eines EJB-UML Profils mit dem Stereotyp «`Entity`» gekennzeichnet sind. Gleichzeitig sind die Klassen über eine OCL-Einschränkung als persistente Objekte ausgewiesen. Mit dem Stereotyp «`Identifier`» wird das jeweilige Attribut als eindeutig identifizierbar markiert. Diese Plattform-unabhängige Modellierung (PIM) wird unter

Verwendung des J2EE Plattform Entity Beans-Profils in ein (EJB)-PSM transformiert. Die PIM-Klassen werden dabei auf `EntityBean`-stereotypisierte Klassen abgebildet und die auf der PIM-Ebene auftretenden «`Identifier`»-Stereotypen werden auf der PSM-Ebene zum «`Key`»-Stereotyp umgewandelt. Mit dem Tagged Value „persistence = cmp“ wird außerdem angegeben, dass die Persistenz vom EJB-Container im Rahmen der Container Managed Persistence (CMP) bereitgestellt werden soll. Ein entsprechender Codegenerator erstellt nun aus diesem PSM zwei Java-Interfaces und die eigentliche Entity Bean Klasse[2].

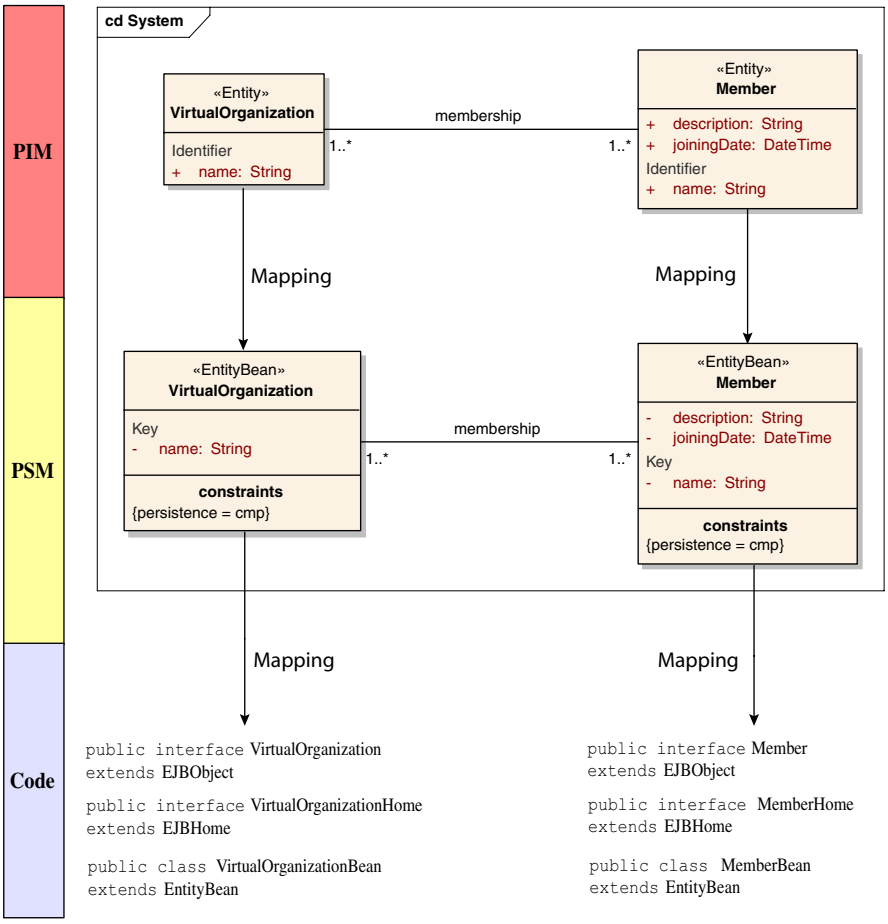


Abbildung 5.1: Einfaches Beispiel zur Verwendung von OCL und UML-Profilen

Bei jedem Transformationsschritt werden in der Regel zusätzliche Informationen benötigt, wie das Zielmodell zu spezialisieren ist. Sollen beispielsweise die in einem UML-Modell mo-

[2]Für den interessierten Leser sind komplexere Beispiele zur OCL in [Kempter, 2004] zu finden.

dellierten Klassen in Quelltexte einer EJB-Anwendung transformiert werden, muss für jede Klasse im Quelltext explizit der zu realisierende Bean-Typ (Entitiy- oder Session-Bean) angegeben werden. Diese zusätzlichen Angaben werden auch Annotation eines Modells bezüglich einer Modelltransformation genannt. Modell-zu-Plattform-Transformationen unterstützen jedoch in der Regel Plattform-spezifische Idiome (z.B. in Form von Design Patterns [Gamma u. a., 1995]), ohne dass diese noch explizit im Modell annotiert werden müssen; sie werden automatisch bei der Transformation berücksichtigt.

Beispiel: Das Problem einer potenziell hohen Netzwerklast in einer EJB-Applikation wird vom Transfer-Object-Pattern (TOP) [Sun Microsystems, 2002] adressiert: Werden Client und Entity Bean auf getrennten Hosts ausgeführt, ist für jeden Methodenaufruf bzw. jedes Attribut der Entity Bean ein Remote-Call nötig. In einer datenlastigen Anwendung, in der häufig große Mengen an Daten zwischen Client und Server ausgetauscht werden, führt dies zwangsläufig zu einer hohen Netzwerklast. TOP löst dieses Problem dadurch, dass ein so genanntes Value Object alle Daten einer Entity Bean kapselt. Der Client führt nur noch einen einzigen Remote-Aufruf auf dem Entity Bean aus und erhält als Ergebnis ein Value Object. Die Netzwerklast wird dadurch erheblich reduziert.

Ein PIM kann in mehrere PSMs transformiert werden. Dadurch stehen die Elemente eines PSMs mit Elementen anderer PSMs in Beziehung. In der Regel werden dann zusätzliche horizontale Abbildungen nötig, um Konzepte eines PSMs A in korrespondierende Konzepte eines PSMs B über eine Bridge umzusetzen. Je nach verwendetem Werkzeug wird diese Transformation automatisch generiert.

Beispiel: Spezifiziert ein PIM ein Element `ManagedElement` und wird das PIM in ein PSM für eine Java-Plattform und ein PSM für eine relationale Datenbank transformiert, so wird für das Java-PSM eine Java-Klasse generiert, für das Datenbank-PSM jedoch eine Tabelle. Der Code, der dann zum Auslesen oder Persistieren von `ManagedElement` nötig ist, wird als Code-Bridge generiert [Kleppe u. a., 2003].

Abbildung 5.2 fasst das Prinzip des Bridge-Konzeptes zusammen.

5.1.2 Der MDA-Entwicklungsprozess

MDA induziert einen inkrementellen und generativen Entwicklungsprozess [Frankel, 2003; Kleppe u. a., 2003; Raistrick u. a., 2004], der mit der Erstellung eines Plattform-unabhängigen Modells des geplanten Systems beginnt. Dieses Modell wird anschließend so annotiert, dass es in ein Plattform-spezifisches Modell transformiert werden kann. Das

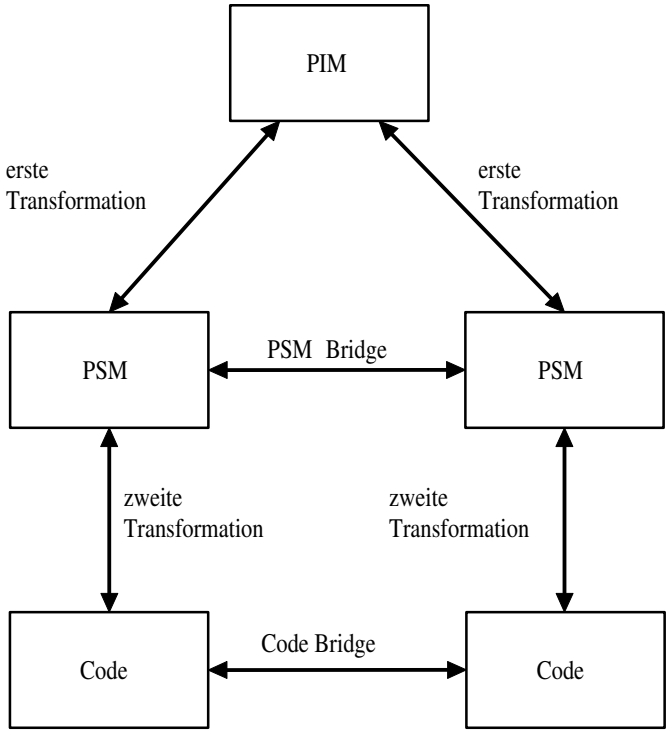


Abbildung 5.2: Modellkonsistenz durch Bridges nach [Frankel, 2003]

Zielmodell muss dann möglicherweise – unter Beibehaltung der Modellkonsistenz – manuell erweitert werden (Verfeinerung). Annotieren, Transformieren und Verfeinern werden nun so lange iteriert, bis ein Modell mit dem gewünschten Grad der Spezialisierung erreicht ist. In der Regel wird dies der Fall sein, wenn das Ergebnis einer Modelltransformation in Form von Quelltexten des geplanten Systems vorliegt. Abbildung 5.3 zeigt den MDA-Entwicklungsprozess und seine wesentlichen Artefakte im Überblick.

Bezogen auf den Entwicklungsprozess in Abbildung 5.3 liegen bisher die Anforderungen in Form von Texten und Diagrammen vor, nachdem im Kapitel 3 die beteiligten Aktoren identifiziert wurden und für jeden Aktor die relevanten Anwendungsfälle bestimmt wurden. Ziel dieses Kapitels ist nun die Erstellung des Plattform-unabhängigen VOMA-Modells (VOMA-PIM). Im sich anschließenden Kapitel 6 wird dann exemplarisch dargestellt, wie das hier erstellte VOMA-PIM auf Plattform-spezifische Modelle (VOMA-PSMs) der WSRF- und WSDM-Rahmenwerke sowie des Globus Toolkits 4.x abgebildet werden kann. An konkreten Beispielen wird die Modell-Code-Transformation ebenfalls im Kapitel 6 dargestellt.

Die Spezifikation des VOMA-PIMs erfolgt in den nachfolgenden Abschnitten in vier Schritten.

1. Die Spezifikation des Informationsmodells erfolgt im Abschnitt 5.2. Die dort entwickelten Informationsstrukturen dienen dem Ziel, ein konsolidiertes Verständnis über die

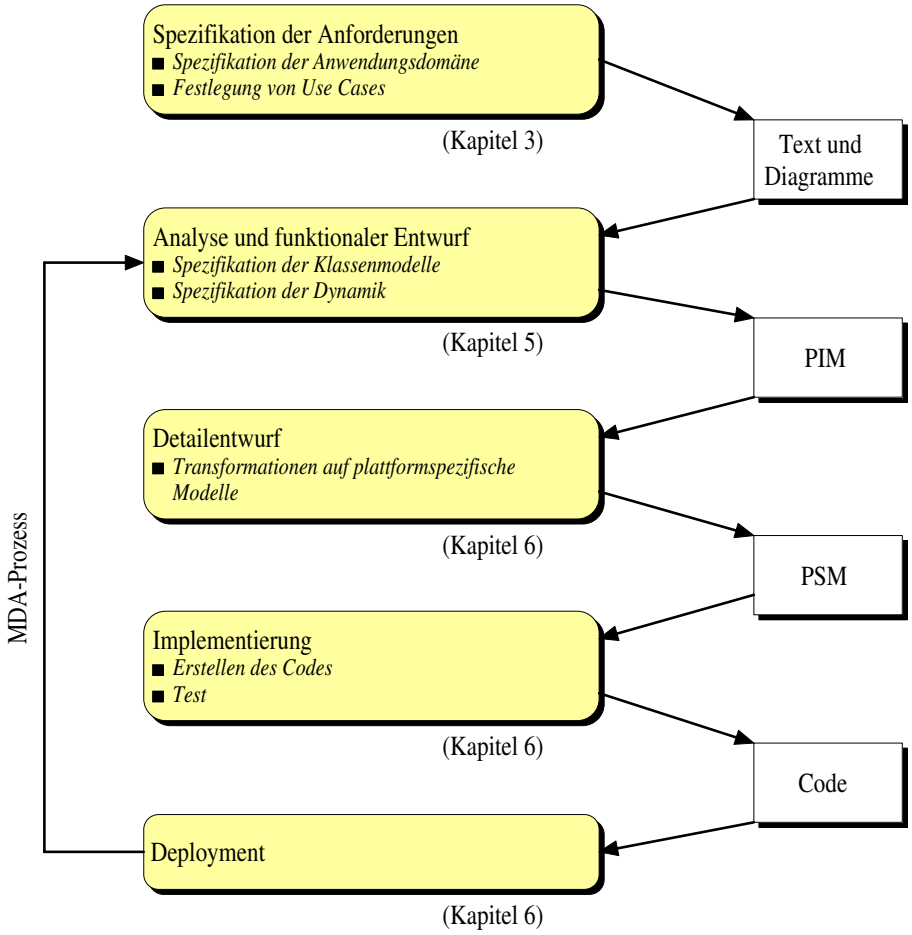


Abbildung 5.3: MDA-Entwicklungsprozess und Artefakte nach [Kleppe u. a., 2003]

Informationen zu gewinnen, die zwischen den an VO-Managementprozessen beteiligten Rollen ausgetauscht werden. Dazu wird eine adäquate VO-Management Information Base (VO-MIB) bereitgestellt[3].

2. Die Spezifikation des Organisationsmodells erfolgt im Abschnitt 5.3 auf der Basis der schon im Abschnitt 3.2.1 identifizierten Aktoren und deren Kooperationsmuster in Form managementrelevanter Interaktionen.

3. Die Spezifikation des Kommunikationsmodells erfolgt im Abschnitt 5.4. Hier werden die Nachrichten und Nachrichtenaustauschprinzipen vor dem Hintergrund Grid- bzw-Web Services-basierter Infrastrukturen erläutert und einige Basisdienste definiert.

4. Die Spezifikation des Funktionsmodells erfolgt im Abschnitt 5.5 mit der Definition VO-Managementspezifischer Funktionalitäten.

[3]Mit der Analyse und Spezifikation dienstorientierter Managementinformationen befasst sich intensiv [Sailer, 2007].

An dieser Stelle ist anzumerken, dass den nachfolgenden Diskussionen zwei verschiedene Modellbegriffe zu Grunde liegen: Einerseits verwenden die VOMA-Teilmodelle einen Modellbegriff zur Beschreibung der verschiedenen Aspekte des VO-Managements, andererseits basieren PIMs und PSMs auf Modell- und Metamodellvorstellungen des Software Engineerings. Eine Integration beider Sichten erfolgt durch die Entwicklung der VOMA-Modelle mit Methoden des MDA-Ansatzes, nämlich dann, wenn die VOMA-Teilmodelle als PIM- bzw. PSM-Artefakte betrachtet werden. Da aus dem Kontext der Diskussion jeweils ersichtlich ist, welcher Modellbegriff gerade verwendet wird, wird diese Doppelbelegung jedoch nicht zu Verwirrungen führen.

5.2 Entwurf des Informationsmodells

Das Informationsmodell einer jeden Managementarchitektur bestimmt einen eindeutigen Beschreibungsrahmen für die Gesamtheit der verwalteten Managementobjekte als Abstraktionen der Charakteristika der Ressourcen, auf denen das Management operiert – hier die VOs. Dazu bedarf es nicht nur der Festlegung einer geeigneten Spezifikationssprache (hier UML) für die Informationsmodellierung, sondern auch der Strukturierung des Sachgebietes (*universe of discourse*) sowie der Definition von abstrakten und generischen „Wurzel"-Klassen (*root entities*) als Ausgangspunkt einer Spezialisierungs- bzw. Vererbungshierarchie zur Ableitung der VO-spezifischen Objektklassen und deren Beziehungen.

An dieser allgemeinen Vorgehensweise orientiert sich auch der Entwurf der VO-MIB in diesem Abschnitt. Zunächst werden im Abschnitt 5.2.1 die für die VO-MIB verwendeten Modelldomänen im Überblick beschrieben. Anschließend wird jede identifizierte Domäne in ihrem Kontext separat betrachtet. Da das VOMA-Informationsmodell einem generischen Ansatz gehorchen soll, wird im Abschnitt 5.2.10 zudem dargestellt, wie die hier spezifizierte VO-MIB auf bestehende Informationsmodelle abgebildet werden kann. Dies wird an drei Beispielen besprochen: dem Common Information Model (CIM) der DMTF [DMTF, 2006a], dem Shared Information/Data Model (SID) des TeleManagement Forums [TMF, 2004] und dem Grid Laboratory Uniform Environment (GLUE)-Schema des OGF [Andreozzi u. a., 2007]. Eine Gesamtdarstellung der Klassenabhängigkeiten rundet die Spezifikation des VOMA-Informationsmodells im Abschnitt 5.2.11 ab.

5.2.1 Beschreibung des Domänenmodells

Im Umfeld objektorientierter Entwicklung ist eine Domäne als separierte reale, hypothetische oder abstrakte „Welt" durch eine eindeutige Menge von Klassen und einem der jeweiligen Welt entsprechenden spezifischen Verhalten charakterisiert [Mellor u. Shlaer, 1991; Raistrick u. a., 2004]. In einer VOMA-*Modell*domäne werden daher diejenigen Entitäten zusammengefasst, die einem spezifischen, konzeptionell abgeschlossenen,

VO-Managementfokus zugeordnet werden können[4], zunächst unabhängig davon, welche *funktionalen* Strukturierungen im Abschnitt 5.5 vorgenommen werden. Die dabei zu berücksichtigenden Managementperspektiven lassen sich nach einer systematischen Analyse der Szenarios und Anwendungsfälle des Kapitels 3 ableiten. Sie induzieren eine Strukturierung des Gesamtkomplexes in Managementbereiche, die in den nachstehenden VOMA-Modelldomänen zur Ordnung des Informationsmodells reflektiert sind.

VOs als *managed objects*. Als „*breeding environments*“ für Virtuelle Organisationen besitzen Communities eine spezifische Sichtweise auf VOs, indem sie VOs als lebenszyklusbehaftete Objekte, die gegründet, ausgeführt und terminiert werden, betrachten. Die Repräsentation von VOs aus dieser Perspektive wird in der Domäne `VirtualOrganization` (siehe Tabelle 5.1) vorgenommen, in der sämtliche Informationen zur Struktur, zur Identifizierung und zum Rollenverständnis einer VO aggregiert werden.

	Modelldomäne *VirtualOrganization*
Name	VO
Beschreibung	betrachtet VOs als *managed object*
Spezifikation	Abschnitt 5.2.3
Anwendungsfälle	F01, F02, F03, T01, T02, T03, T04

Tabelle 5.1: Zusammenfassung der Domäne *VirtualOrganization* des VOMA-Informationsmodells

Zur Aufrechterhaltung des Betriebes einer einzelnen VO werden in dieser Domäne die notwendigen Prozesse zur Aufnahme, Adaption und Auflösung von Rollen, Mitgliedschaften, Ressourcen und Dienste initiiert, überwacht und gegebenenfalls angepasst. Die damit verbundenen Kernaktivitäten werden an die jeweiligen Domänen `Member`, `Role`, `VirtualResource` bzw. `VirtualService` delegiert. Das Vertragsmanagement zwischen den diversen Aktoren und VO-spezifische Policies werden in der Domäne `Management` behandelt. VO-Management setzt in einigen Fragestellungen eine adäquate Vertrauensbasis voraus, die Gegenstand der Domäne `Trusted Entities` sind.

Mitgliedschaften in VOs. Die Zweckorientierung einer VO erfordert ein adäquates Management von VO-Mitgliedschaften, die für Individuen, Gruppen von Individuen oder Organisationen realer und virtueller Art bestehen können. „Mitglied“ ist zwar auch

[4]Dieser Domänenbegriff ist von dem des IT-Managements zu unterscheiden. Während im IT-Management Domänen im Rahmen eines Organisationsmodells Managementobjekte mit gemeinsamen Eigenschaften bezeichnen [Hegering u. a., 1999], sind Domänen hier thematisch orientierte Gruppierungen von Entitäten zur Strukturierung von Modellen [Raistrick u. a., 2004].

als Rolle aufzufassen, zu Modellierungszwecken werden jedoch alle eine Mitgliedschaft betreffenden Informationen in der Domäne `Member` gehalten (siehe Tabelle 5.2).

Modelldomäne *Member*	
Name	MEM
Beschreibung	Management von Mitgliedschaften zu VOs
Spezifikation	Abschnitt 5.2.5
Anwendungsfälle	F02, B01, B02, B03, T04

Tabelle 5.2: Zusammenfassung der Domäne *Member* des VOMA-Informationsmodells

Die `Member`-Domäne ist für das Hinzufügen, Entfernen und Ändern von Mitgliedschaften zu einer gegebenen VO verantwortlich.

Mitglieder werden auf Rollen abgebildet. Mitglieder können in VOs unterschiedliche Rollen spielen, die zudem dynamisch zugeordnet werden können. Rollen werden in der Domäne `Role` modelliert (siehe Tabelle 5.3), die für das Hinzufügen, Entfernen und Ändern von Rollen zu einer gegebenen VO verantwortlich ist.

Modelldomäne *Role*	
Name	ROL
Beschreibung	Domäne für das Management von Rollen
Spezifikation	Abschnitt 5.2.6
Anwendungsfälle	F02, B04, B05, B06, T01, T02, T04

Tabelle 5.3: Zusammenfassung der Domäne *Role* des VOMA-Informationsmodells

Auch hier zeigen die Anwendungsfälle, dass das Management von Rollen und deren Handlungsmaxime auf Policies und Verträgen zwischen diversen Rollen basiert. Diese werden in der `Management`-Domäne behandelt. Der Umgang von Rollen mit virtuellen Diensten und virtuellen Ressourcen ist Gegenstand der entsprechenden Domänen `VirtualResource` und `VirtualService`.

Trusted Entities. Sollen Mitglieder in eine VO aufgenommen und auf Rollen abgebildet werden, ist eine entsprechende Authentifizierung und – im Rahmen der Nutzung von Ressourcen und Diensten – eine hinreichende Autorisierung erforderlich. Dies gilt insbesondere auch dann, wenn juristische Personen (z.B. reale Organisationen) und Virtuelle Organisationen selbst die Mitgliedschaft zu einer VO beantragen. Um die dann erforderlichen Authentifizierungs- und Autorisierungsprozesse zu unterstützen, werden in der Domäne `Trusted Entities` anerkannt vertrauenswürdige Entitäten zusammengefasst (siehe Tabelle 5.4).

Modelldomäne *Trusted Entities*	
Name	TRUST
Beschreibung	Domäne vertrauenswürdiger Entitäten
Spezifikation	Abschnitt 5.2.9
Anwendungsfälle	F02, B01, B02, B03, B04, B05, B06

Tabelle 5.4: Zusammenfassung der Domäne *Trusted Entities* des VOMA-Informationsmodells

Bereitstellung virtueller Ressourcen. VOs stellen für ihre Mitglieder virtuelle Ressourcen zur Nutzung bereit. Diese werden in der Domäne `VirtualResource` modelliert (siehe Tabelle 5.5). Sie ist verantwortlich für alle Aktivitäten im Kontext virtueller Ressourcen.

Modelldomäne *VirtualResource*	
Name	VR
Beschreibung	Domäne für das Management virtueller Ressourcen
Spezifikation	Abschnitt 5.2.7
Anwendungsfälle	F02, B07, B08, T04

Tabelle 5.5: Zusammenfassung der Domäne *VirtualResource* des VOMA-Informationsmodells

Aus der Sicht des VO-Managements dieser Arbeit beinhaltet diese Domäne *nicht* die Verantwortung für die „Komposition“ virtueller Ressourcen aus realen Ressourcen, wie sie vom Quota-Provider bereitgestellt werden (siehe Abschnitt 3.2.1), sondern lediglich die Bekanntgabe und Überwachung der virtuellen Ressource und deren Zugangspolicies. Anzumerken ist, dass erstens in dieser Domäne keine Beschränkung bzgl. der Anzahl der beteiligten VOs besteht, eine virtuelle Ressource kann also von mehreren VOs bereitgestellt werden. Zweitens besteht eine „ownership“-Relation zwischen VOs und virtuellen Ressourcen. Die VO übernimmt insofern die Rolle eines Ressourcenanbieters. Drittens ist weder die Anzahl der virtuellen Ressourcen an sich beschränkt noch die Anzahl gleicher Ressourcen. Zwei Ressourcen R_1 und R_2 sind dann gleich, wenn sie die gleiche Spezifikation besitzen; als virtuelle Ressource können sie jedoch von unterschiedlichen Gruppen von Resource Providern bereitgestellt (und damit implementiert) werden, was deren Gleichheit jedoch nicht in Frage stellt.

Bereitstellung virtueller Dienste. VOs stellen für ihre Mitglieder virtuelle Dienste auf virtuellen Ressourcen bereit. Diese werden in der Domäne `VirtualService` modelliert. In Analogie zur `VirtualResource`-Domäne verantwortet die `VirtualService`-Domäne (siehe Tabelle 5.6) alle für das VO-Management erforderlichen Aktivitäten im Kontext virtueller Dienste.

Modelldomäne *VirtualService*	
Name	VS
Beschreibung	Domäne für das Management virtueller Dienste
Spezifikation	Abschnitt 5.2.8
Anwendungsfälle	F02, B07, B08, T04

Tabelle 5.6: Zusammenfassung der Domäne *VirtualService* des VOMA-Informationsmodells

Dies beinhaltet *nicht* die „Komposition“ virtueller Dienste aus realen Diensten, sondern lediglich die Bekanntmachung und Überwachung der Dienste und deren Nutzungspolicies. Anzumerken ist auch hier, dass erstens in dieser Domäne keine Beschränkung auf die Anzahl der beteiligten VOs besteht, ein virtueller Dienst kann also von mehreren VOs bereitgestellt werden. Zweitens besteht auch hier eine „ownership“-Relation zwischen VOs und virtuellem Dienst und drittens wird auch hier nichts über die Anzahl der virtuellen Dienste an sich oder über die Anzahl der gleichen Dienste vorausgesetzt, wobei zwei Dienste S_1 und S_2 dann gleich sind, wenn sie die gleiche Spezifikation besitzen; als virtuelle Dienste können sie jedoch von unterschiedlichen Gruppen von Service Providern bereitgestellt werden.

VO-Management-Domäne. VOs werden in dieser Arbeit aus der Perspektive des IT-Managements betrachtet. Insofern ist eine Notwendigkeit gegeben, VOs als *managed objects* nach Gesichtspunkten eines effizienten und effektiven Managements gruppieren und Managementanwendungen zuführen zu können. Damit sollen zwei Ergebnisse erzielt werden: Eine Definierbarkeit von Verantwortlichkeiten (Wer managt was?) und eine Festlegung administrativ gleichgearteter Bereiche. Policies spielen dabei eine wichtige Rolle, allerdings im Kontext dieser Arbeit nur soweit sie für das *VO*-Management spezifisch sind. Aspekte dieser Art zum VO-Management werden in der Domäne `Management` modelliert (siehe Tabelle 5.7).

Modelldomäne *Management*	
Name	MAN
Beschreibung	Domäne für VO-spezifisches Policy-basiertes Management
Spezifikation	Abschnitt 5.2.4
Anwendungsfälle	alle

Tabelle 5.7: Zusammenfassung der Domäne *Management* des VOMA-Informationsmodells

Policies, die im Kontext des Managements von Mitgliedschaften, von Rollen und von virtuellen Ressourcen und Diensten von Bedeutung sind, sind primär Gegenstand lo-

kaler Managementsysteme der Quota-Provider, die den Zugang zu „ihren“ Ressourcen und Diensten über entsprechende Sicherheits- und Nutzungs-Policies regeln, auf deren Semantik das VO-Management jedoch keinen Einfluss hat. Das VO-Management muss allerdings mit entsprechenden Strukturen und Prozessen dafür Sorge tragen, dass diese Policies durchgesetzt werden können.

TopLevel-Domäne. Der Sinn der `TopLevel`-Domäne ist es, einen Satz allgemeiner VO-Management-Entitäten zu definieren, die gemeinsam als Grundlage der Architektur dienen und die Kohärenz des Informationsmodells sicherstellen. Insofern werden in dieser Domäne – aus Modellierungssicht – die Wurzel-Entitäten und die weitgehend abstrakten Oberklassen zu finden sein (siehe Tabelle 5.8).

Modelldomäne *TopLevel*	
Name	TOP
Beschreibung	TopLevel-Domäne für das VO-Management
Spezifikation	Abschnitt 5.2.2
Anwendungsfälle	alle

Tabelle 5.8: Zusammenfassung der Domäne *TopLevel* des VOMA-Informationsmodells

Ebenfalls auf dieser Abstraktionsebene angesiedelt sind die in der Domäne `BaseTypes` zusammengefassten Basistypen und UML-Profile, die zur Modellierung der VO-MIB verwendet werden.

Abbildung 5.4 zeigt die Domänen und ihre Abhängigkeiten im Überblick. Die Abhängigkeiten sind dabei nicht funktional zu verstehen, sondern im Sinne klassenspezifischer Spezialisierungen und Assoziationen. Eine detailliertere Übersicht wird später in den Abbildungen 5.33 und 5.32 in einem anderen Kontext geliefert.

Das Management Virtueller Organisationen stellt einen komplexen und vielschichtigen Problembereich dar, der in dieser Arbeit erstmals systematisch untersucht wird. Mit der Bereitstellung einer entsprechenden Managementarchitektur wird das Ziel verfolgt, VOs als *managed objects* zu behandeln. In dem in diesem Abschnitt zu entwickelnden Informationsmodell der Architektur werden die für das VO-Management erforderlichen Objekte und deren Beziehungen so festgelegt, dass sie leicht erweiterbar und wiederverwendbar sind. Es ist jedoch nicht Gegenstand dieser Arbeit, umfassende VO-Managementanwendungen zu antizipieren und zu entwickeln. Insofern wird zwar ein Informationsmodell bereitgestellt, das solche Aspekte unterstützt, die Realisierung solch wichtiger Verfahren wie beispielsweise die Auditierung zur Bewertung von VO-Governance (z.B. in Anlehnung an [IT Governance Institute, 2005]), von Grid-weiten Service- und Supportprozessen (z.B. in Anlehnung an ITIL [Brenner, 2007; Dugmore, 2006; Köhler, 2004]) oder die Realisierung eines Quality-of-VO-Konzeptes muss durch weitergehende Arbeiten erfolgen. In diesen Kontext sind auch

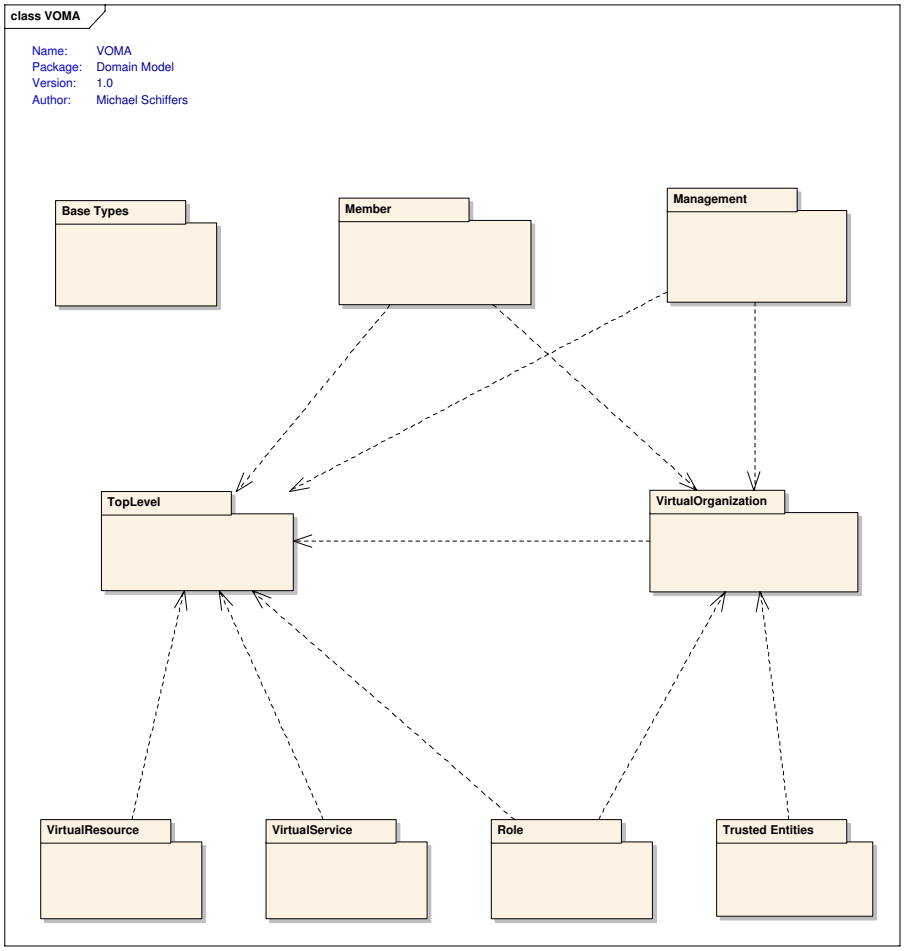


Abbildung 5.4: Domänen des VOMA Informationsmodells

effiziente Trust-Management-Konzepte zu setzen. Diese werden hier nicht diskutiert, es sei für eine weitergehende Behandlung dieser Thematik auf [Boursas u. Hommel, 2006; Dimitrakos u. a., 2004] verwiesen. Ein Quality Model für Web Services, das als Ausgangspunkt solcher Überlegungen dienen kann, wird zudem in [Kim u. Lee, 2005] vorgestellt.

Nach dieser allgemeinen Darstellung des verwendeten Domänenmodells werden die einzelnen Domänen nun detailliert diskutiert. Dazu werden die Klassen der Domänen in ihrem Kontext dargestellt und die darunter liegenden Basiskonzepte erörtert. Die Klassendiagramme beschreiben die Objekte *statisch*. In späteren Abschnitten wird deshalb diese Statik durch die Modellierung von dynamischen Interaktions- und Zustandstransitions-Phänomenen angereichert.

Bei den folgenden Darstellungen werden stillschweigend einige Annahmen getroffen:

- Kapitel 4 hat gezeigt, dass in einigen Bereichen bewährte Konzepte des Common Information Models (CIM), des Shared Information/Data Models (SID) und des OGF-GLUE-Schemas sinnvoll übernommen werden können, hin und wieder unverändert,

meistens allerdings nur in modifizierter Form. Wir werden dies an geeigneter Stelle jeweils vermerken.

- Bei der Spezifikation der einzelnen Klassen in den Domänen werden der besseren Übersichtlichkeit halber die typischen `Get-` und `Set`-Operationen für gekapselte Attribute nicht explizit aufgeführt. Dies geschieht lediglich exemplarisch für die Wurzelklasse `VOMARootEntity` in Abbildung 5.5.
- Um in den folgenden Diagrammen die im Vordergrund des Interesses stehenden Darstellungen besser einordnen zu können, werden die Klassen stets in ihrem Kontext beschrieben. Deshalb wird die jeweilige Spezialisierungshierarchie – soweit relevant – immer mit angegeben.

5.2.2 Die Domäne `TopLevel`

Der Fokus der Domäne `TopLevel` liegt auf der Bereitstellung eines Satzes allgemeiner VO-Management-Entitäten als Grundlage des VOMA-Informationsmodells. `TopLevel` enthält die abstrakten und generischen Wurzelklassen, von denen fast sämtliche VOMA-Managementobjektklassen über Vererbungsmechanismen abgeleitet werden. Abbildung 5.5 macht dies deutlich.

Das VOMA-Informationsmodell basiert auf dem klassischen Ansatz objektorientierter Modellierung mit `VOMARootEntity` als abstrakter Wurzelklasse der gesamten VOMA Klassenhierarchie. Ihr Zweck ist die Definition eines Satzes von Attributen und Methoden, die für alle VOMA-Entitäten gültig sind. Diese bestehen in der Identifizierung von Objektinstanzen über das Attribut `objectID`, die Bennenung von Entitäten (`entityName`) und einer textuellen Beschreibung der Entität (`description`). Zusätzlich wird die aktuelle VOMA-Version im Attribut `vomaVersion` hinterlegt. Die Attribute sind gekapselt und können mit den entsprechenden `get-` und `set`-Operationen gelesen bzw. verändert werden.

`VOMAEntity` bezeichnet eine abstrakte Klasse, die `VOMARootEntity` spezialisiert, um solche Objekte zu repräsentieren, die im Rahmen des Managements Virtueller Organisationen von Bedeutung sind. Da VOs nur eine begrenzte Lebensdauer besitzen, werden die Attribute der `VOMARootEntity`-Klasse um den Erstellungszeitpunkt (Attribut `creationDate`) erweitert. Die `VOMAEntity`-Klasse selbst lässt sich nun weiter spezialisieren in Entitäten, auf die das VO-Management direkt wirkt, und in solche, auf die das VO-Management zwar keinen direkten Managementeinfluss hat, die aber Managementinformationen liefern und Ressourcen oder Dienste bereitstellen. Die erste Kategorie wird durch die Klasse `ManagedVOMAEntity` repräsentiert, in der VOs wieder VOs enthalten können. Diese Rekursivität wird durch das Composite Pattern [Fowler, 1996] ausgedrückt. Aus den Szenarios im Abschnitt 3.1.1 ergibt sich die Assoziation zwischen Virtuellen Organisationen und Communities (Assoziation `HasHomeCommunity`), die das Konzept des *breeding environments* ausdrückt. Angelehnt an das GLUE-Schema [Andreozzi u. a., 2007] wird für VOMA das Konzept der `VOMASite` übernommen, um auch für VOs die Aggregation von Ressourcen und Diensten zu administrativ gleich zu behandelnden Bereichen darstellen zu können.

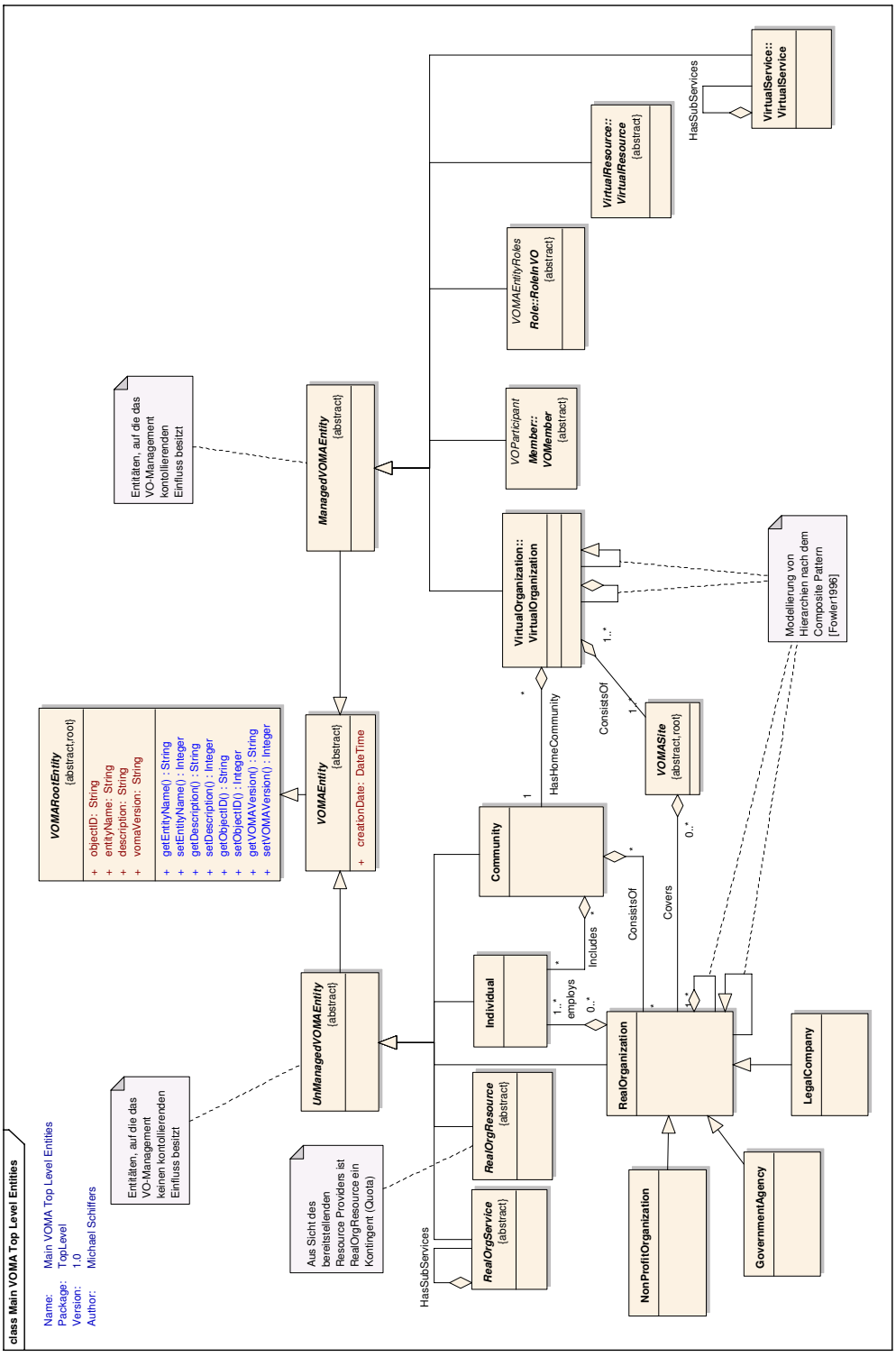


Abbildung 5.5: Wurzel-Entität und TopLevel-Hierarchie

Eine `VOMASite` ist damit nicht unbedingt mit einer Domäne im Sinne von Domain Name Systems (DNS) gleichzusetzen. Eine `VOMASite` ist aber auch keine VO, da letztere mehrere administrative Bereiche umfassen kann (siehe etwa das EmerGrid-Szenario). Allenfalls besteht eine VO aus mehreren `VOMASites`. Eine `VOMASite` ist in der Regel auch keine reale Organisation, da in ihrem Fokus virtuelle Ressourcen und Dienste stehen. Schließlich ist eine `VOMASite` auch keine Aggregation im Sinne einer OSI-Managementdomäne [Hegering u. a., 1999], sie überlagert diese eher.

Die zweite Kategorie wird durch die Klasse `UnManagedVOMAEntity` repräsentiert, deren Objekte in der Regel rein lokal gemanagt werden, für das VO-Management aber dennoch von Bedeutung sind. Dies betrifft reale Organisationen, Individuen, Dienste realer Service Provider und Ressourcen realer Resource Provider. In diesen Kontext werden auch die Communities selbst eingeordnet (Klasse `Community`), da sie aus thematisch affinen Individuen und realen Organisationen bestehen. Der Begriff der realen Organisation wird hier nicht im engen Sinn verstanden, sondern umfasst neben juristischen Personen in Form formaler Unternehmen oder behördlicher Organisationen auch weniger formale Gruppierungen und Teilorganisationen. Organisationen als `UnManagedVOMAEntity` können im Rahmen von Allianzen und Konsortien kooperieren, was wiederum durch das Composite Pattern modelliert wird.

Von der `ManagedVOMAEntity`-Klassen leiten sich nun die Entitäten der weiteren Domänen ab, die in den nachstehenden Abschnitten im Detail beschrieben werden.

5.2.3 Die Domäne `VirtualOrganization`

Der Fokus der Domäne `VirtualOrganization` liegt auf der Betrachtung Virtueller Organisationen als *managed objects*. Dementsprechend stehen in dieser Domäne Modelle im Vordergrund, die diese Perspektive mit einer Sicht auf den Lebenszyklus Virtueller Organisationen, auf deren Struktur, deren Adressierung und deren Prozesse unterstützen.

Modellierung von VOs

Virtuelle Organisationen stellen Entitäten mit einer zeitlich befristeten Lebensdauer dar. Dementsprechend sind, wie Abbildung 5.6 zeigt, Instanzen dieser Klasse gekennzeichnet durch einen Gründungszeitpunkt (Attribut `foundationDate`), eine a priori festgesetzte Lebensdauer (Attribut `validFor`) und den aktuellen Status im Lebenszyklus (Attribut `currentLifecycleStatus`). Die zu einem Zeitpunkt t noch verbleibende Lebensdauer kann durch die Operation `getTimeToLive()` berechnet werden. Da VOs in Communities gegründet werden und grundsätzlich mehrere Communities an einer VO beteiligt sein können (z.B. bei Community-übergreifenden VOs im IPY-Szenario), werden Communities mit VOs assoziiert und nicht als Attribute vorgesehen. An dieser Stelle ist für das VO-Management nur der Community-spezifische Ansprechpartner für Belange des VO-Managements interessant (Attribut `communitySpokesman`). Zusätzlich werden jedoch Community-spezifische Informationen in entsprechenden VO-Attributen hinterlegt. Dazu gehören insbesondere

der Name des VO-Gründers (Attribut `founder`) und dessen zum Gründungszeitpunkt relevante Community (Attribut `foundingCommunity`).

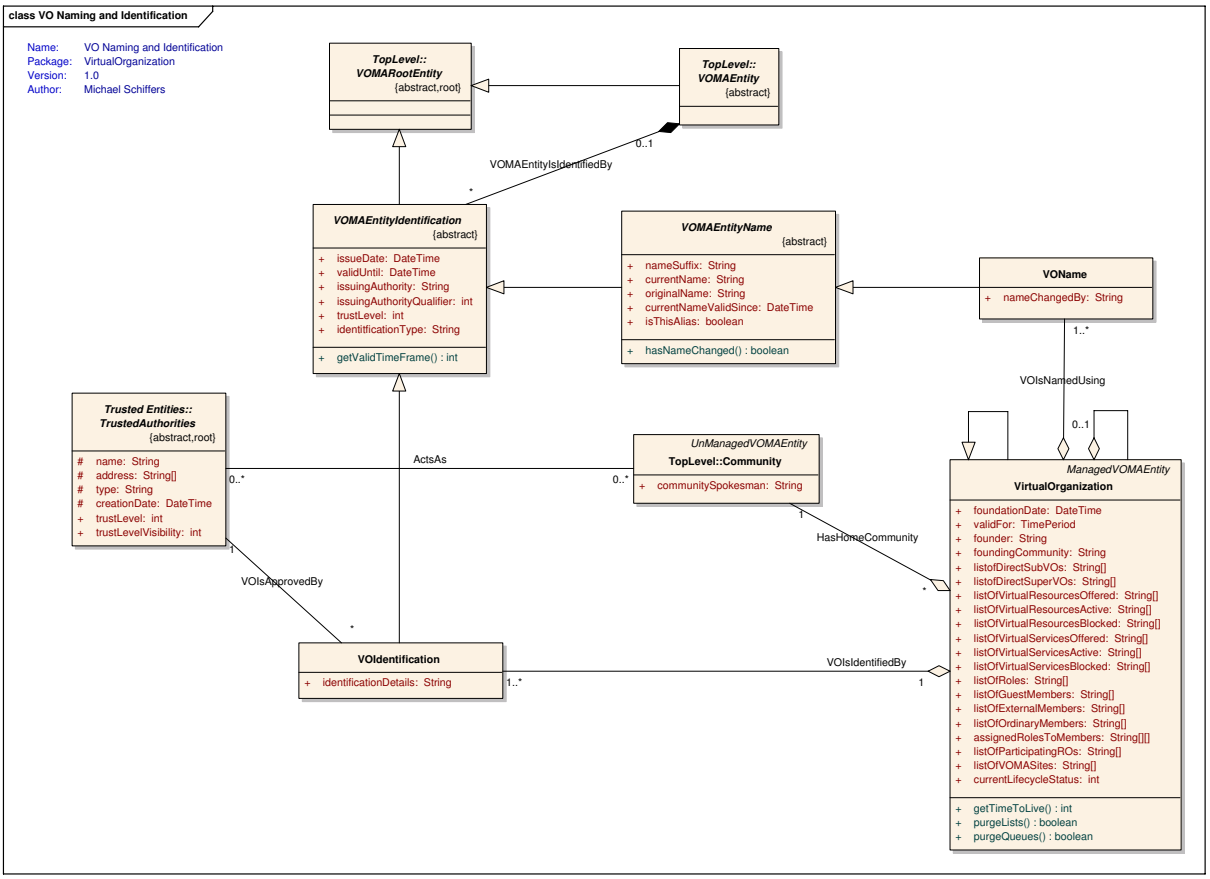


Abbildung 5.6: Identifizierung und Benennung von VOs

Nehmen VOs als Entitäten an Grid-Prozessen teil, müssen sie als solche zweifelsfrei identifizierbar sein. Die Verfahren, nach denen dies geschehen kann, sind in der Klasse `VOIdentification` hinterlegt und durch eine global anerkannte Autorität (Klasse `TrustedAuthorities`) überwacht. *Eine* solche Autorität können die Communities – vertreten durch den Community-Sprecher (Attribut `communitySpokesman`) selbst darstellen.

VOs können – wie die Szenarios EmerGrid und IPY zeigen – Sub-VO enthalten und selbst als Sub-VO Teil einer Super-VO sein. Von Interesse ist dabei das Wissen um diese Enthaltenseins-Beziehungen, die in Form gerichteter azyklischer Graphen vorliegen. Die Attribute `listofDirectSubVOs` und `listofDirectSuperVOs` drücken dies in der Klasse `VirtualOrganization` in Abbildung 5.6 aus.

Die „interne Struktur“ einer VO wird bestimmt durch die Mitglieder der VO, deren Rollen, den externen Teilnehmern (z.B. Berater), den bereitgestellten virtuellen Ressourcen und Diensten sowie den vorhandenen `VOMASites`. Die aktuellen Belegungen der Attribute `listofVirtualResourcesOffered` bis `listofVOMASites` der Klasse

`VirtualOrganization` spiegeln dies wider. Zu beachten sind für virtuelle Ressourcen und Dienste die Unterscheidbarkeit von ursprünglichem Angebot und aktuellem Angebot sowie die Möglichkeit, Ressourcen und Dienste zu blockieren, entweder proaktiv z.B zu Wartungszwecken oder als Reaktion auf Missbrauch. Diese Listen können mit der Methode `purgeLists` bereinigt werden, für Warteschlangen (z.B Jobs) geschieht dies korrespondierend mit der Operation `purgeQueues`.

Benennung von VOs

Virtuelle Organisationen sind keine formalrechtlichen Entitäten. Insofern fehlt ihnen neben einer registergerichtlichen Eintragung auch eine formale Adresse im Sinne realer Organisationen. Referenzierbar sind sie nur über Namen. Einer VO können dabei mehrere Namen zugeordnet sein. Diese sind jedoch nicht nur über die Zeit veränderbar, sie können auch mehrfach auftreten. Um dies adäquat auszudrücken, werden Namen als separate Entitäten entsprechend dem Characteristic Value Pattern (CVP) [Raistrick u. a., 2004] und nicht als Attribute modelliert (siehe Abbildung 5.6).

Beispiel: Eine Namensänderung der D-Grid-VO *HEP* zu *HEP-D-VO* ist ein typisches Beispiel für eine einfache Namensänderung. Mehrfachnamen (*aliases*) werden in IPY verwendet, wenn beispielsweise die VO *Antarctic Sea Ice* mit den zusätzlichen Namen *Antarktis 141* und *Projet Antarctique XIVa* gekennzeichnet wird.

VOs werden mit Namen (Klasse `VOName`) über die Assoziation `VOIsNamedUsing` gekoppelt. `VOName` selbst ist von der abstrakten Klasse `VOMAEntityName` abgeleitet, die wiederum als Spezialisierung die Attribute und Operationen der `TopLevel`-Klasse `VOMAEntityIdentification` erbt und nicht nur VOs als `ManagedVOMAEntity` adressiert, sondern auch die `UnManagedVOMAEntity`s. Neben einer formalen Authentifizierung einer VOMA-Entität bieten in dieser Konstruktion Namen eine alternative Möglichkeit, Entitäten zu identifizieren. Diese Vorgehensweise kann dann von Bedeutung sein, wenn ein vertrauenswürdiges Umfeld schon besteht und weiter oder wieder genutzt wird.

Die für den VO-Namen relevanten (geerbten oder klassenspezifischen) Attribute sehen neben einem Namens-Suffix (Attribut `nameSuffix`) den aktuellen Namen (Attribut `currentName`) vor. Beispiele für Namens-Suffixe sind im LCG-Grid [LCG, 2005] die Tier-Bezeichner (Tier 0, Tier1, ...) oder Zusätze wie „Consortium“ oder „Project“. Zu Auditierungs- und Trust-Managementzwecken dienen der Originalname der VO (Attribut `originalName`) und der Zeitpunkt des letzten Namenswechsels (Attribut `currentNameValidSince`), aus dem auch mit Hilfe der Operation `hasNameChanged()` abgeleitet werden kann, ob der Name geändert wurde und von wem (Attribut `nameChangedBy`). Die generalisierte Klasse `VOMAEntityIdentification` leitet aus der beschränkten Lebensdauer der VO die noch verbleibende Dauer der Namensgültigkeit (Attribut `validUntil`

ab, die über die Operation `getValidTimeFrame()`) zur Verfügung gestellt wird. Ob der angegebene Name der legale Name ist oder ob es sich um einen Alias handelt, wird über das Attribut `isThisAlias` bestimmt.

Identifizierung von VOs

Unabhängig vom Namen wird ein formaler Weg benötigt, VOs zu identifizieren und zu authentifizieren, nicht nur, um als „Kunde“ Grid-Dienste in Anspruch nehmen zu können, sondern auch im Rahmen eines allgemeinen Network-of-Trust [DEISA, 2006]. Für reale Unternehmen wird ein derartiger Authentisierungsnachweis in der Regel durch registergerichtliche Auszüge unterstützt, ein solcher Mechanismus ist jedoch für VOs nicht anwendbar. Die konkreten Identifizierungmechanismen, die im Rahmen des VO-Managements im Einzelnen zum Tragen kommen können, werden in der abstrakten `TopLevel`-Klasse `VOMAEntityIdentification`, die zur Identifizierung von VOMA-Entitäten genutzt wird, über das Attribut `identificationType` kodiert (siehe Abbildung 5.6). Eine konkrete Darstellung der möglichen Methoden kann allerdings im Rahmen dieser Arbeit nicht erbracht werden. Stattdessen wird auf [Gietz u. a., 2007] und [Hommel, 2007] verwiesen. Dennoch lassen sich einige allgemeine Merkmale feststellen: Authentifizierungsmechanismen involvieren immer eine Ausgabestelle der Identifikation, ein Ausstellungsdatum und eine Gültigkeitsbeschränkung. Diese sind respektive in den Attributen `issuingAuthority`, `issueDate` und `validUntil` manifestiert. Zusätzlich wird im Attribut `trustLevel` hinterlegt, welchen Vertrauensstatus (*trust level*) die Ausgabestelle besitzt. Über die Operation `getValidTimeFrame()` wird der noch verbleibende Gültigkeitszeitraum bestimmt. Ein weiterer Indikator für die Beurteilung der Vertrauenswürdigkeit der VO besteht in der Markierung der `issuingAuthority` durch den `issuingAuthorityQualifier`, in dem repräsentiert wird, ob die `issuingAuthority` eine `TrustedAuthority` ist, es sich um eine Selbstauthentifizierung handelt, eine allgemein vertrauenswürdige Institution oder um eine unbekannte Quelle.

Mit den Konstruktionsmerkmalen, die Benennung von VOs als Spezialfall der Identifizierung aufzufassen und die Identifizierung von VOMA-Entitäten direkt von der `VOMARootEntity` abzuleiten, steht insgesamt ein flexibles Portfolio zur Modellierung und Implementierung von Authentifizierungsmechanismen Virtueller Organisationen zur Verfügung, mit denen die spezifischen Identifizierungsanforderungen, wie sie aus den Anwendungsfällen des Kapitels 3 abgeleitet werden können, erfüllt werden können, beispielsweise im Rahmen föderierter Identitymanagement-Konzepte [Hommel, 2007].

Kollaboration und Kontaktierung

VOs können über verschiedene Medien kontaktiert werden (siehe Abbildung 5.7). Konzeptionell wird der Kontakt jedoch nicht direkt mit der VO stattfinden, sondern mit der *Rolle*, die die VO einnimmt. Damit wird die Anforderung nach rollenbasierten Kontaktlisten erfüllt: Im D-Grid möchte man nicht immer die HEP-ATLAS-VO als solche kontaktieren,

sondern die HEP-ATLAS-VO als Provider eines Massenspeicherdienstes. Gleichzeitig bietet die VO ihren Mitgliedern die Möglichkeit, im Rahmen des koordinierten Problemlösens (siehe Kapitel 1) eine Reihe von Möglichkeiten zur IT-gestützten Kollaboration.

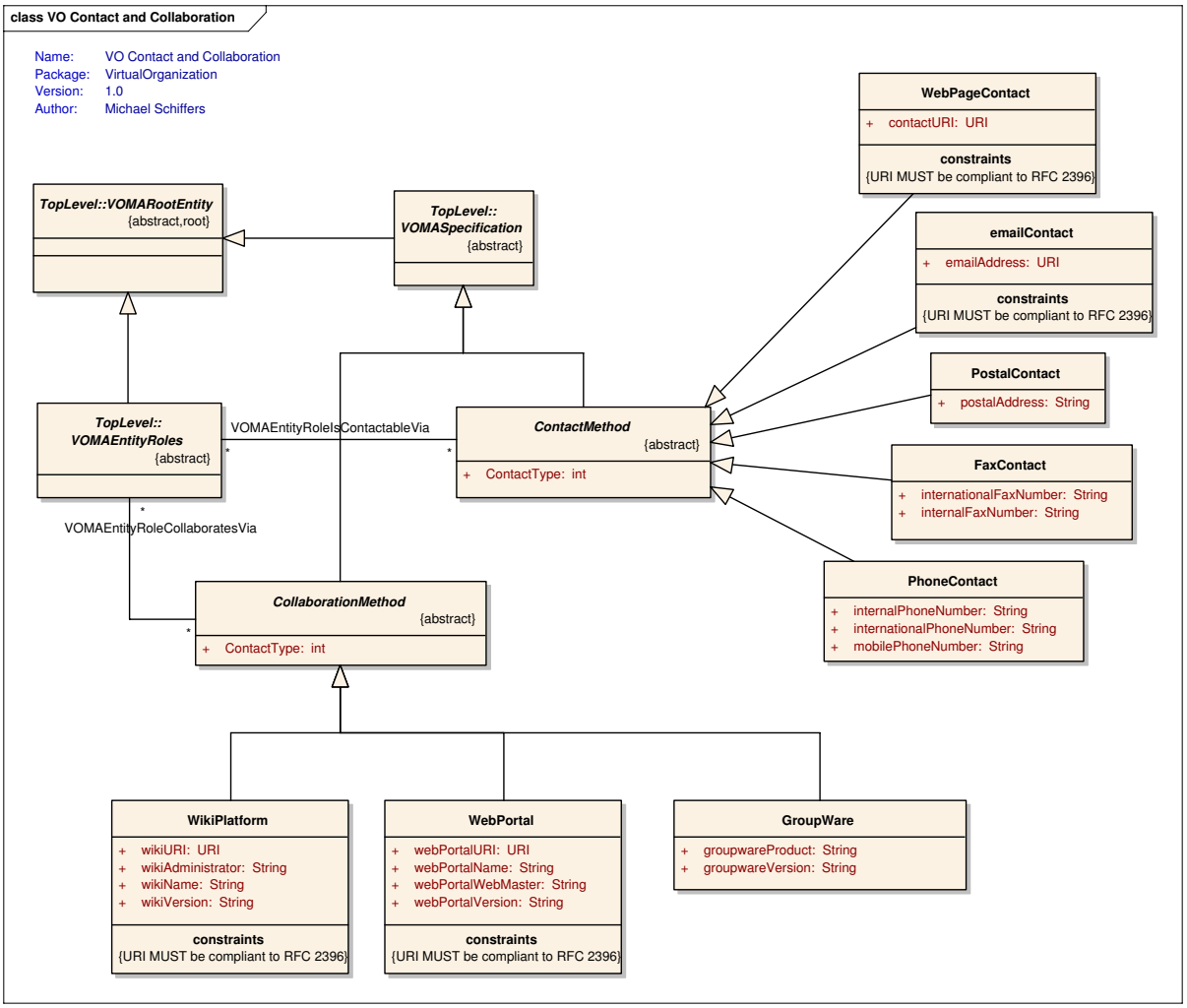


Abbildung 5.7: Kontaktierung und Kollaboration

Dementsprechend werden die abstrakten Klassen `ContactMethod` und `CollaborationMethod` als VOMA-Spezifikation begriffen und mit VO-Rollen im weitesten Sinn über `VOMAEntityRoleContactableVia` bzw. `VOMAEntityRoleCollaboratesVia` assoziiert. Beide Klassen zeigen die unterstützten Methoden und die primär verwendeten Methoden in den entsprechenden Attributen und Spezialisierungen an. Um das Modell generisch zu halten, wird vorgeschlagen, die Attribute möglichst Standard-konform zu belegen. Diese Anforderungen sind in den entsprechenden *Constraints* wiedergegeben.

Lebenszyklen Virtueller Organisationen

Als temporäre *managed objects* besitzen VOs einen Lebenszyklus, der mit der Initiierung beginnt und mit der Terminierung der VO beendet wird (siehe auch Abbildung 3.12). Abbildung 5.8 zeigt, wie im VOMA-Informationsmodell Lebenszyklen in ihrer *Struktur* repräsentiert werden, nicht in ihrer Dynamik.

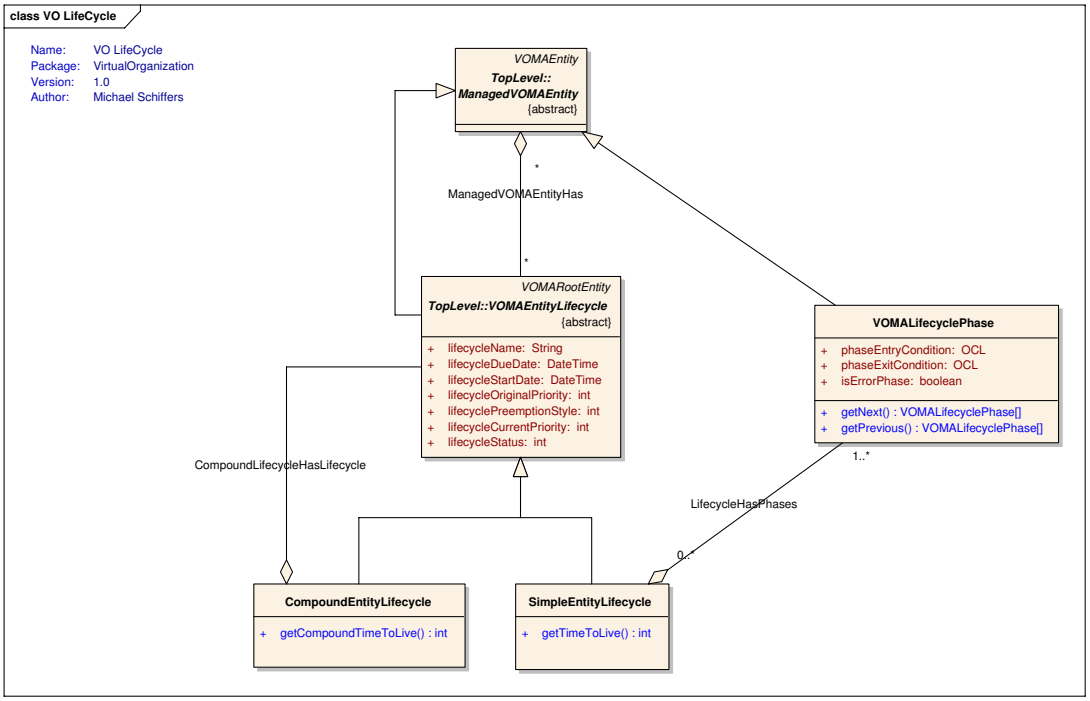


Abbildung 5.8: Lebenszyklus Virtueller Organisationen

Lebenszyklen werden selbst wieder als *managed object* durch die Ableitung von der `TopLevel`-Klasse `ManagedVOMAEntity` begriffen und allgemein modelliert durch die Angabe von Lebenszyklusphasen über die Klasse `VOMALifecyclePhase`, den Startzeitpunkt im Attribut `lifecycleStartDate` und das projektierte Ende im Attribut `lifecycleDueDate`. Die potenzielle Schachtelung von Lebenszyklen wird unter Verwendung des Composite Patterns [Fowler, 1996] modelliert. VO-Lebenszyklen können – um beispielsweise das EmerGrid-Szenario umsetzen zu können – mit Prioritäten versehen sein und als „unterbrechbar“ deklariert werden (Attribut `lifecyclePreemptionStyle`). Prioritäten können sich ändern. Insofern sind in der Klasse `VOMAEntityLifecycle` beide Attribute (`lifecycleOriginalPriority` und `lifecycleCurrentPriority`) vorgesehen. Mit den Operationen `getTimeToLive()` bzw. `getCompoundTimeToLive()` wird die jeweils verbleibende „Restlaufzeit“ des Lebenszyklus berechnet.

5.2.4 Die Domäne `Management`

Der Fokus der Domäne `Management` liegt auf der eigentlichen Durchführbarkeit und Durchführung von Managementaktivitäten gegen und mit VOMA-Entitäten, die als `ManagedVOMAEntity` oder `UnManagedVOMAEntity` vorliegen.

In der Domäne `Management` werden die Bereiche des VOMA-Informationsmodells angesprochen, die die Gruppierung von VOMA-Entitäten nach Managementgesichtspunkten unterstützen und die Einrichtung von Objektdomänen gestatten, deren administrative Gleichbehandlung entweder durch gleiche User, gleiche Gruppen von Usern, gleiche Rollen innerhalb einer VO oder gleiche Policies induziert werden[5]. Abbildung 5.9 zeigt die Modellierung der am eigentlichen VO-Management im engeren Sinn beteiligten Bereiche.

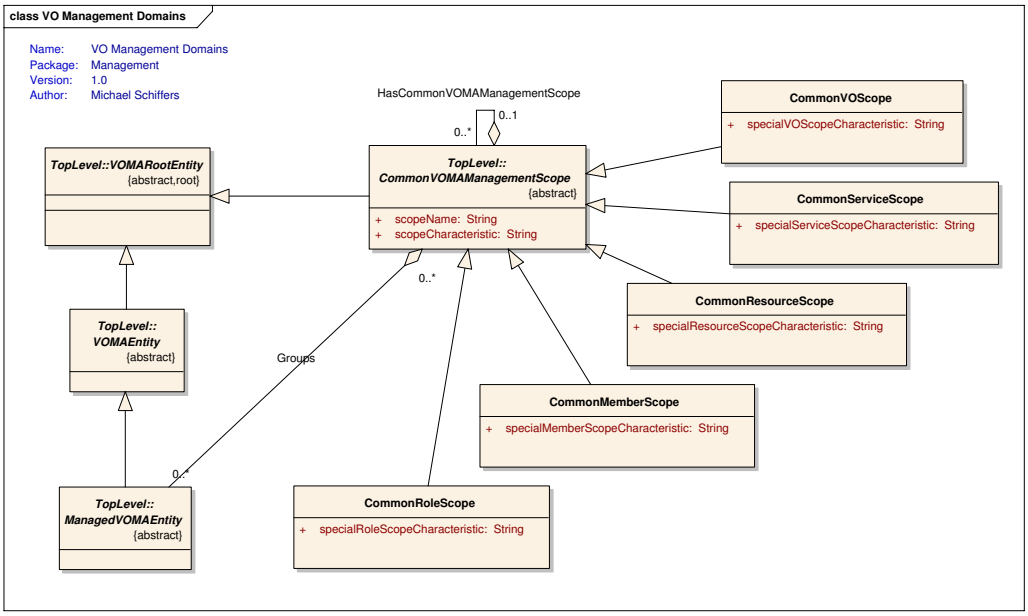


Abbildung 5.9: Managementdomänen

In VOMA können die zu managenden Entitäten in geschachtelten Gruppen zusammengefasst werden (Klasse `CommonVOMAManagementScope`), um sie einer gemeinsamen Vorgehensweise im Management, die im Attribut `scopeCharacteristic` ausgedrückt wird, auszusetzen. Mögliche Gruppen können über eine gemeinsame VO-Sicht gebildet werden oder über eine Mitgliedschaftssicht.

Beispiel: Eine Community im D-Grid möchte sämtliche ihrer VOs nach den gleichen Prinzipien managen, die im übrigen nicht unbedingt als Policies ausgedrückt werden müssen.

---

[5]Um Mehrdeutigkeiten des Domänenbegriffs zu vermeiden, wird im Folgenden die hier angesprochene *administrative* Domäne mit dem Begriff „Bereich“ (*scope*) umschrieben.

Anzumerken ist, dass nur Instanzen der `ManagedVOMAEntity` einer solchen Gruppierung zugeführt werden können, da auf eine `UnManagedVOMAEntity` ja kein Managementeinfluss ausgeübt werden kann. Die folgenden Spezialfälle einer managementrelevanten Gruppierung können aus den Anforderungen abgeleitet werden:

- Gruppierung von Mitgliedern (Klasse `CommonMemberScope`)
- Gruppierung von Rollen (Klasse `CommonRoleScope`)
- Gruppierung von virtuellen Ressourcen (Klasse `CommonResourceScope`)
- Gruppierung von virtuellen Diensten (Klasse `CommonServiceScope`)
- Gruppierung von VOs (Klasse `CommonVOScope`)

Für jede dieser Gruppierungen beschreibt das angegebene Attribut (z.B `specialRoleScopeCharacteristic`) die zusätzlichen – `scopeCharacteristic` erweiternden – Gruppierungskriterien.

VOMA-Entitäten können neben diesen Kriterien auch nach gemeinsamen Management-Policies zusammengefasst werden. Dieser Aspekt wird im nächsten Abschnitt näher ausgeführt.

Policies

Policies spielen im Rahmen des Managements Virtueller Organisationen auf verschiedenen Ebenen eine nicht unerhebliche Rolle. Zum einen sind die einer VO zugeordneten lokalen Ressourcen und Dienste auch den lokalen Policies der „Owner“ unterworfen, zum anderen werden VOs zusätzliche Regeln aufstellen, um einen reibungslosen Betrieb der Virtuellen Organisation sicherzustellen. Für das VO-Management steht deshalb nicht unbedingt die Semantik der Policies im Vordergrund, vielmehr muss im VOMA-Informationsmodell dargestellt werden, welche Entitäten an einem Policy-basierten VO-Management beteiligt sind und wie deren Beziehungsgeflecht verstanden wird. Dieser Aspekt wird im VOMA Policy Framework modelliert (siehe Abbildung 5.10), das sich in Teilen an das SID-Policy Framework [TMF, 2007] anlehnt.

Der zentrale Begriff des Frameworks ist der einer zeitlich befristeten VOMA-Policy. Diese wird in der Klasse `VOMAPolicy` mit einem Namen, einem Typ und einer Lebensdauer (Attribut `validFor`) repräsentiert. Beispiele für `vomaPolicyTypes` sind VOLifecycle-Policies, Logging-Events, VOMembership-Policies, Security-Policies, Reaktionen auf Notifikationen oder Abrechnungsregeln. Eine VOMA-Policy ist selbst wieder als *managed object* konzipiert und erbt als solches Beschreibungs-, Gründungszeitpunkt- und Identifier-Attribute. VOMA-Policies werden in Policy-Applikationen genutzt, die ebenfalls *managed objects* sind und mit ihren Aktionen auf ein `VOMAPolicyTarget` zielen. Klassische Policy-Applikationen liegen in Formen wie Policy Enforcement Points (PEP) oder Policy Decision Points (PDP) oder Policy Execution Points (PXP) vor [Strassner, 2003]. Die für VOMA relevanten Targets sind VOs, ihre Rollen und ihre Mitglieder. Im Gegensatz zu `VOMAPolicy`

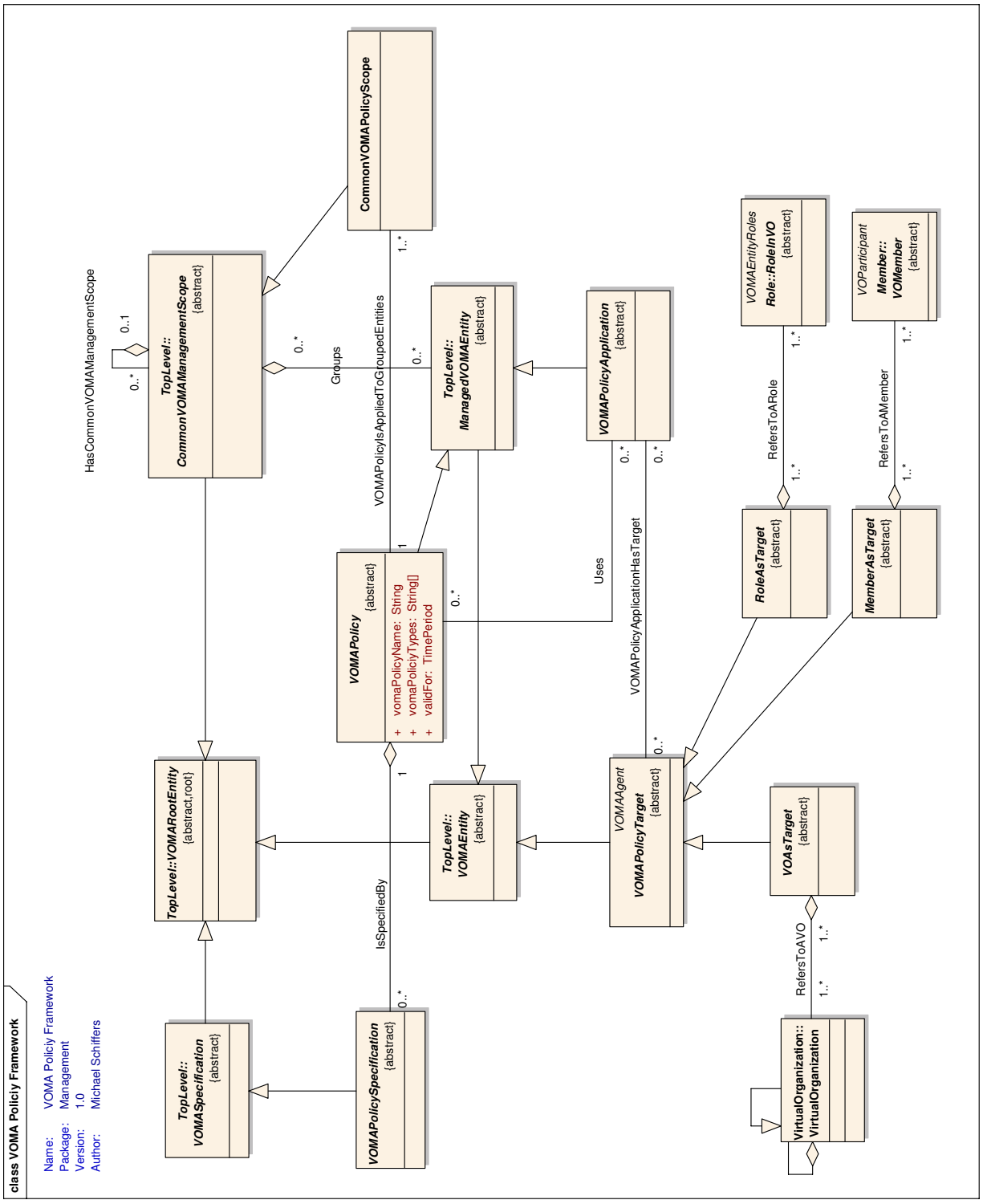


Abbildung 5.10: VOMA Policy Framework

und VOMAPolicyApplication ist ein VOMAPolicyTarget jedoch nicht notwendigerweise ein *managed object*, sondern kann auch eine UnManagedVOMAEntity sein.

Die Semantik von Policies steht hier nicht im Vordergrund. Diese wird in den Spezifikationsklassen VOMAPolicySpecification als Spezialisierung einer allgemeinen VOMASpecification-Klasse festgelegt [TMF, 2007]. Als Spezialisierung des allgemeinen CommonVOMAManagementScope werden in der Klasse CommonVOMAPolicyScope nur diejenigen Entitäten gruppiert, die nach gleichen Policy-basierten Managementprinzipien gemanagt werden. Die Assoziation VOMAPolicyIsAppliedToGroupedEntities definiert dabei explizit, auf welche ManagedVOMAEntities in einer CommonVOMAPolicyScope-Klasse Policies anwendbar sind.

Die Beziehung zwischen Policies, Policy-Events und deren Spezifikation wird aus Abbildung 5.11 deutlich. Policy-Events repräsentieren die Ereignisse, die die Ausführung von Policies anstoßen. Diese werden in „Sammlungen“ (*collections*) als Abstraktion von Mengen von Ereignissen oder Folgen von Ereignissen (geordnet oder ungeordnet, mit oder ohne Duplikate, festgelegt durch das Attribut collectionType) hinterlegt, einem aus [TMF, 2007] übernommenen Konzept.

VOMAPolicyEvents können hierarchisch angeordnet werden (Composite-Pattern [Fowler, 1996]) und sind mit einem eventspezifischen Status (evaluationStatus) versehen, der den Rückgabe-Code der letzten Evaluation des Events durch die Methode pullEvents angibt. In VOMA sind primär VO-spezifische, rollenspezifische und Mitglieder-spezifische Events sowie Events im Rahmen von Logging-Verfahren von Bedeutung, die in den korrespondierenden Spezifikationen definiert werden. Letztere sind insbesondere für VO-orientierte Post Mortem-Analysen und Audits relevant.

Policies werden in Policy Repositories hinterlegt, so auch die VOMAPolicies. Anders als im klassischen Fall lokaler Managementsysteme, „besitzt“ eine VO jedoch keine physischen Repositories für die Speicherung seiner Policies. Vielmehr müssen logische Repositories auf physische Ressourcen der UnManagedVOMAEntities abgebildet werden. VOMA unterstützt dies mit dem aus [Andreozzi u. a., 2007] übernommenen VOMASite-Konzept, indem VOMASites – die ja *Hosting-Environments* in UnManagedVOMAEntities umfassen, die Policy-Repositories „hosten“ (siehe Abbildung 5.12).

Das Repository selbst wird als spezielle Collection-Klasse mit einer beschränkten Lebensdauer modelliert, um die für VOs und deren Policies geltende Dynamik zu unterstreichen.

VO-Managementinteraktionen

VOMAPolicies und VO-Workflows dienen im Rahmen von VO-Managementinteraktionen zur Herstellung und Aufrechterhaltung der Betriebsbereitschaft Virtueller Organisationen. Diese Zusammenhänge werden in den Abbildungen 5.13, 5.14 und 5.15 mit den zentralen Klassen VOManagementInteraction, VOMAContract und VOMAWorkflowSpecification dargestellt.

Eine VOManagementInteraction wird nicht von der Wurzel-Entität VOMARootEntity

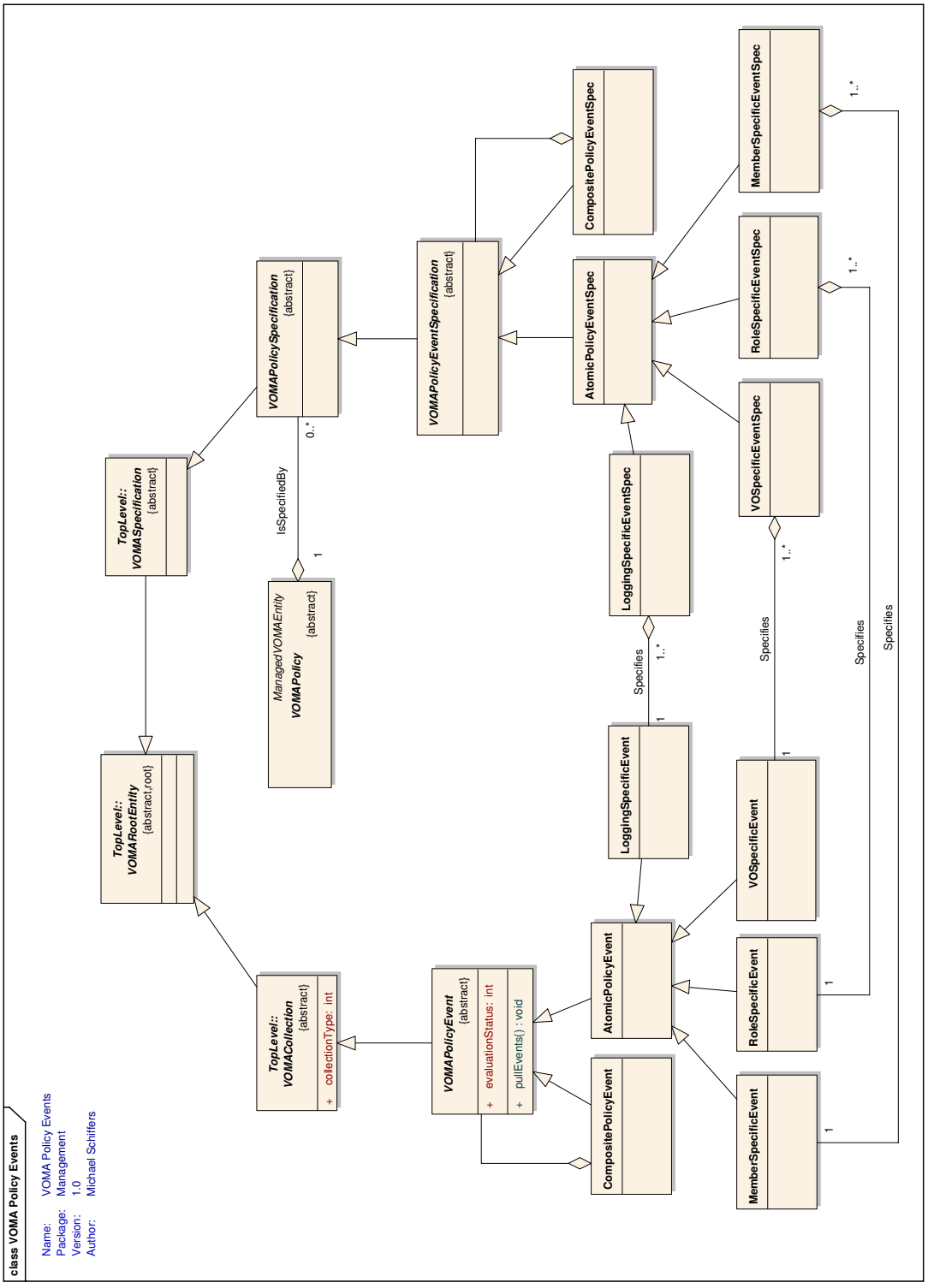


Abbildung 5.11: VOMA Policy Events

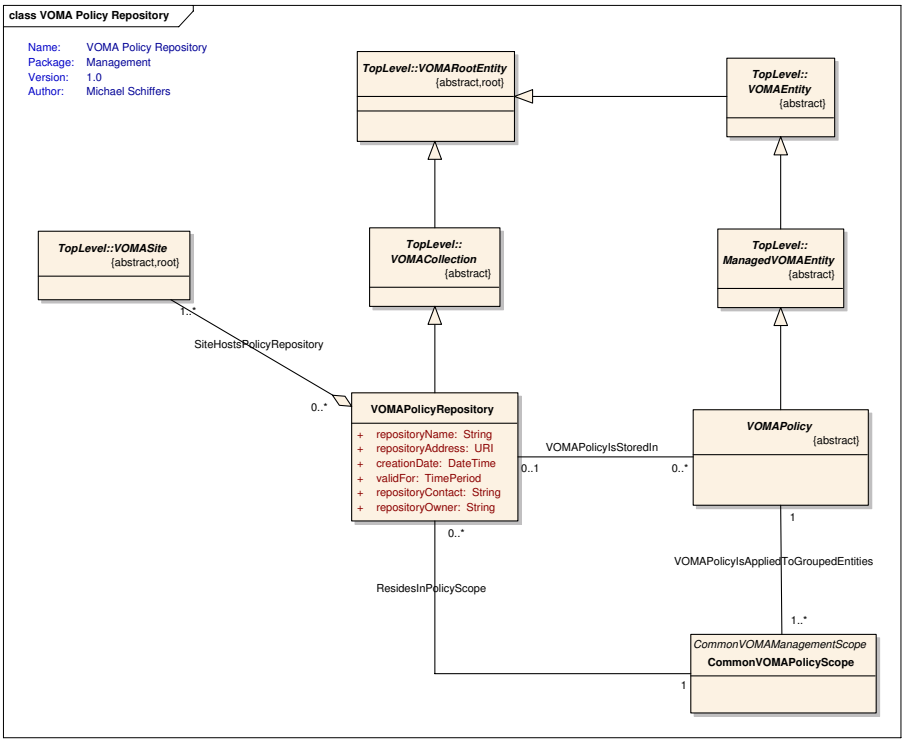


Abbildung 5.12: VOMA Policy Repository

abgeleitet, sondern dient selbst als Wurzel-Entität für managementspezifische Geschäftsprozesse. `VOManagementInteraction` wird neben den generischen Identifizierungs- und Beschreibungsattributen durch typische Prozessattribute (`initiationDate`, `plannedCompletionDate`, `projectedCompletionDate`), den aktuellen Status der Interaktion und den Fokus der Interaktion (VO, Member, Rolle) im Attribut `interactionFocusType` beschrieben.

Spezialfälle einer `VOManagementInteraction` bilden die `VOMAContracts` mit den daraus abgeleiteten `VOMASpecificSLA`, die auch über die Klasse `VOMASpecification` spezifiziert werden. `VOMAContracts` besitzen eine Vertragsnummer, eine Lebensdauer (die mit der Operation `getContractTimeToLive()` überprüft werden kann), ein Ziel (Attribut `contractObjectives`) und den eigentlichen Vertragsbeginn (`contractInceptionDate`). Daneben werden Verträge im Rahmen des VO-Managements immer gegengezeichnet (Attribute `approvalDate` und `approvedBy`). Weitere Beispiele für `VOManagementInteractions` sind in Abbildung 5.14 mit den `VOMAOperationRequest`, `VOMAResponse` und `VOMANotification` zu finden.

Managementinteraktionen involvieren neben `VOMASites` – insbesondere wenn virtuelle Ressourcen und Dienste im Mittelpunkt stehen – Entitäten-Rollen, die klassisch in der Form von Managern (Klasse `VOMAManagingRole`) und Agenten (Klasse `VOMAAgent`) vorliegen. Als `VOMAManagingRole` fungieren die Rollen im `virtualStaff` und die VO selbst,

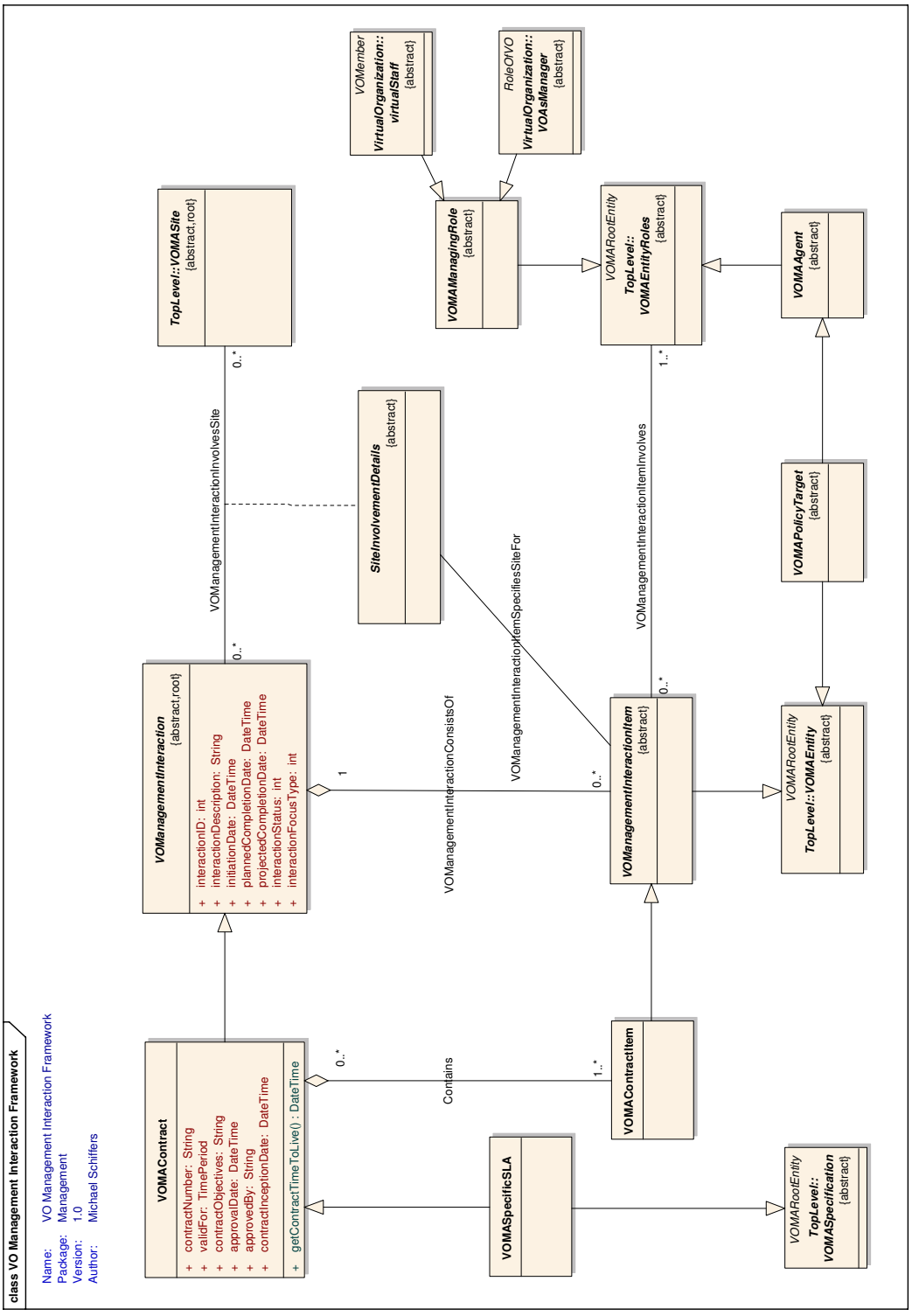


Abbildung 5.13: VO-Managementinteraktionen

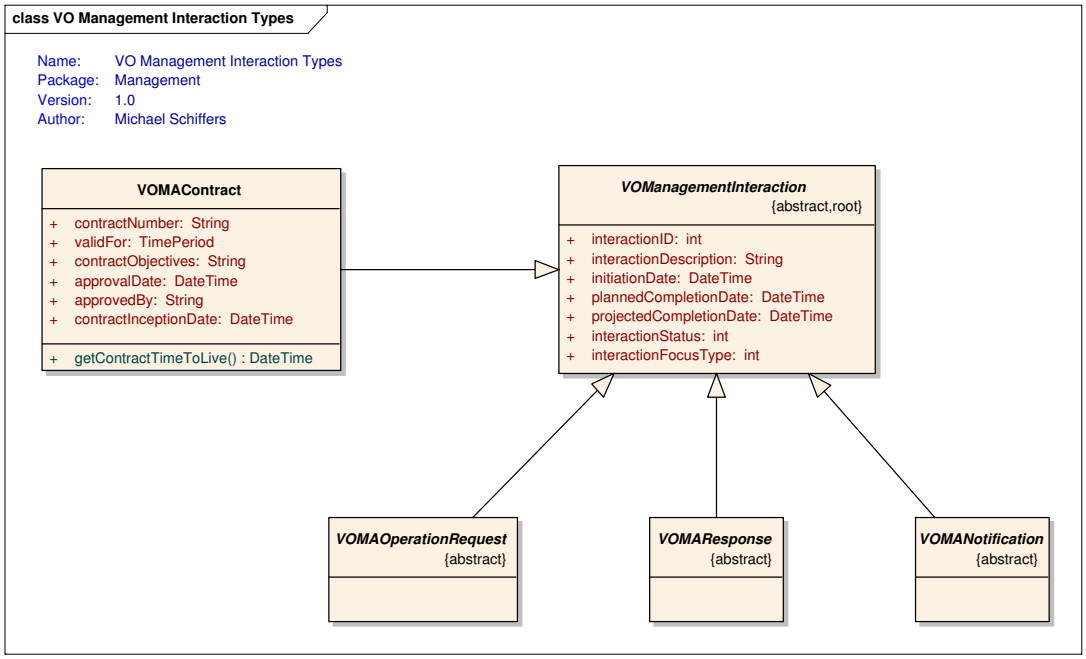


Abbildung 5.14: Typen von VO-Managementinteraktionen

wenn sie die Rolle eines Managers übernimmt, beipielsweise als Initiator einer Sub-VO.

`VOMAWorkflows` werden im Rahmen des VOMA-Informationsmodells als Spezialfall von `VOMAPolicies` verstanden. Sie werden von der allgemeinen Spezifikationsklasse `VOMASpezification` abgeleitet und bestehen aus Workflowknoten (`ControlNode` und `ActivityNode`) und Workflowtransitionen (siehe Abbildung 5.15).

Logging und Archivierung

`LoggingSpecificEvents` bilden einen speziellen Trigger für `VOMAPolicies` und generieren einen entsprechenden Eintrag in das `VOMAArchiv`, ebenfalls eine Ableitung der `Collection`-Klasse (siehe Abbildung 5.16). Das Archiv wird von einer `VOMASite` betrieben und durch die zum `virtualStaff` gehörende Rolle des `VOArchiveManagers` gemanagt.

Interaktionen mit lokalen Managementsystemen

Lokale Managementsysteme (LMS) werden in dem hier verfolgten Konzept in realen Organisationen realisiert und von `VOMASites` genutzt. LMSs selbst sind nicht Gegenstand von VO-seitigen Managementinterventionen, können aber zum Betrieb lokaler Ressourcen und Dienste `VOMAPolicies` in den `VOMARepositories` verwenden (siehe Abbildung 5.17).

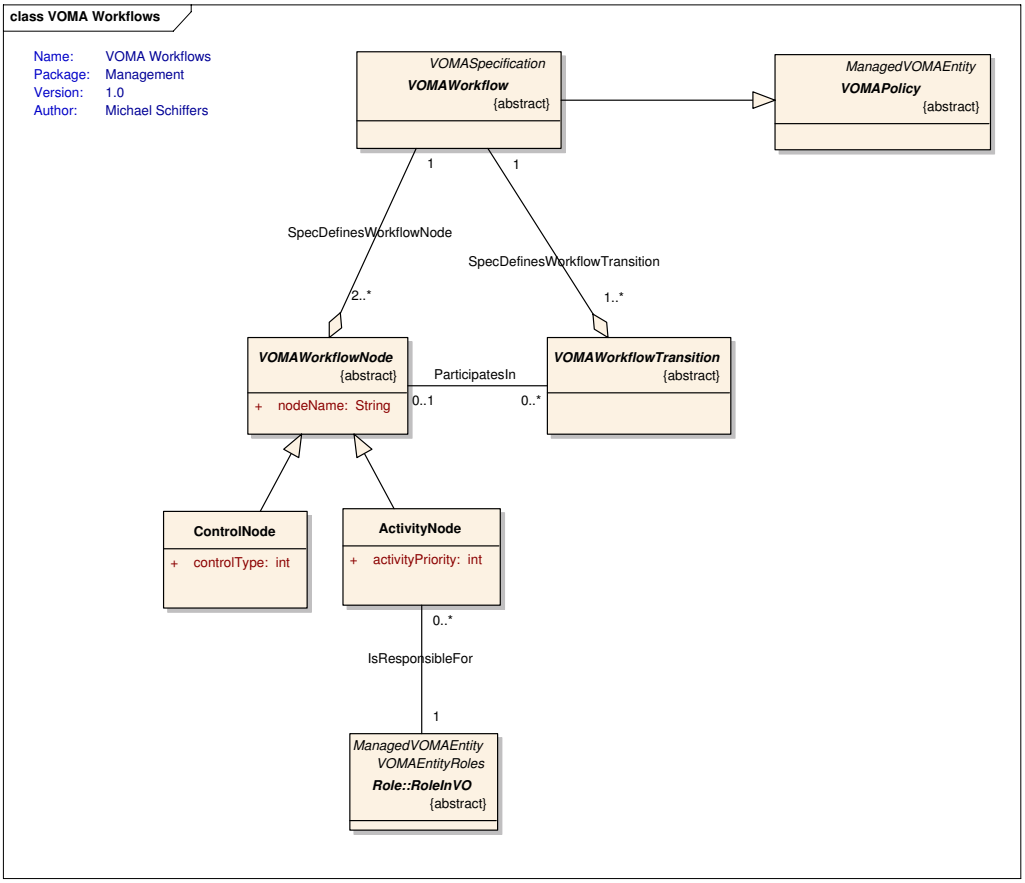


Abbildung 5.15: Workflows

5.2.5 Die Domäne `Member`

Namen und Identifizierung von Individuen

VOs sind Kollektionen von Organisationen, Gruppen und Individuen, die – ähnlich wie Organisationen – über Namen adressiert werden und sich über unterschiedliche Verfahren authentifizieren. Allerdings sind diese Individuen aus dem VO-Management heraus nicht direkt zu managen (siehe auch Abschnitt 5.2.2), sie gehören zur Klasse `UnManagedVOMAEntity`. Die Identifizierung von Individuen ist aber ebenso Bestandteil des VO-Managements wie die Gruppierung von Individuen zu Managementbereichen (siehe auch Abschnitt 5.2.4)

Abbildung 5.18 zeigt das allgemeine Modell zur Benennung und Identifizierung von Individuen im Rahmen des VO-Managements, das zu den Spezifikationen der DFN-AAI [Gietz u. a., 2006] kompatibel ist.

Die Authentifizierung von Individuen selbst kann über verschiedene Verfahren geschehen, einige sind in Abbildung 5.18 angegeben.

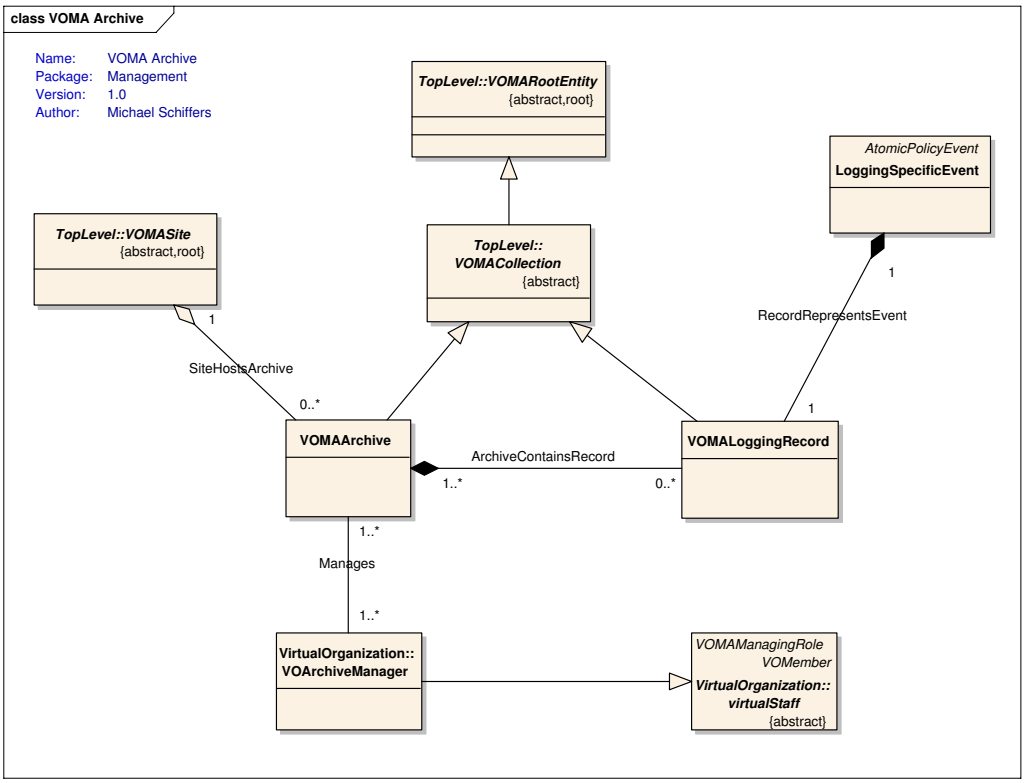


Abbildung 5.16: Logging und Archivierung

Individuelle VO-Mitgliedschaft

Individuen beantragen die Mitgliedschaft zu einer VO oder werden dazu eingeladen. Konzeptionell sind sie in beiden Fällen `VOApplicants`, deren „Bewerbung“ in der Klasse `IndividualAppliesForMembershipDetails` modelliert wird. Der Antrag zur Aufnahme ist mit einem Zeitstempel versehen (Attribut `applicationRequestDate`) und beinhaltet im Attribut `applicationRequestJustification` eine Begründung des Antrages zum Zeitpunkt `membershipRequestDate`. Im Attribut `applicationRequestScope` werden schablonenartig (siehe auch die Darstellung der VOMA Base Types im Abschnitt 5.2.11) die Erwartungen an die VO dargestellt. Die Beantragung der Aufnahme stellt ein VOMA-Policy-Event dar und wird entsprechend behandelt. Abbildung 5.19 zeigt die Beziehungen im Überblick.

Mit der Genehmigung der Aufnahme wird der `VOApplicant` zum `VOParticipant` ab einem definierten Zeitpunkt (Attribut `startDate`). `VOParticipants` werden hier als abstraktes Konzept eingeführt, um die eigentlichen Mitglieder einer VO und die externen Partner in einigen VO-Managementfragen gleich behandeln zu können. Wenn keine Missverständnisse zu befürchten sind, werden wir im Folgenden dennoch häufig den Begriff „Mitglied“ auch für diese erweiterte Sicht verwenden.

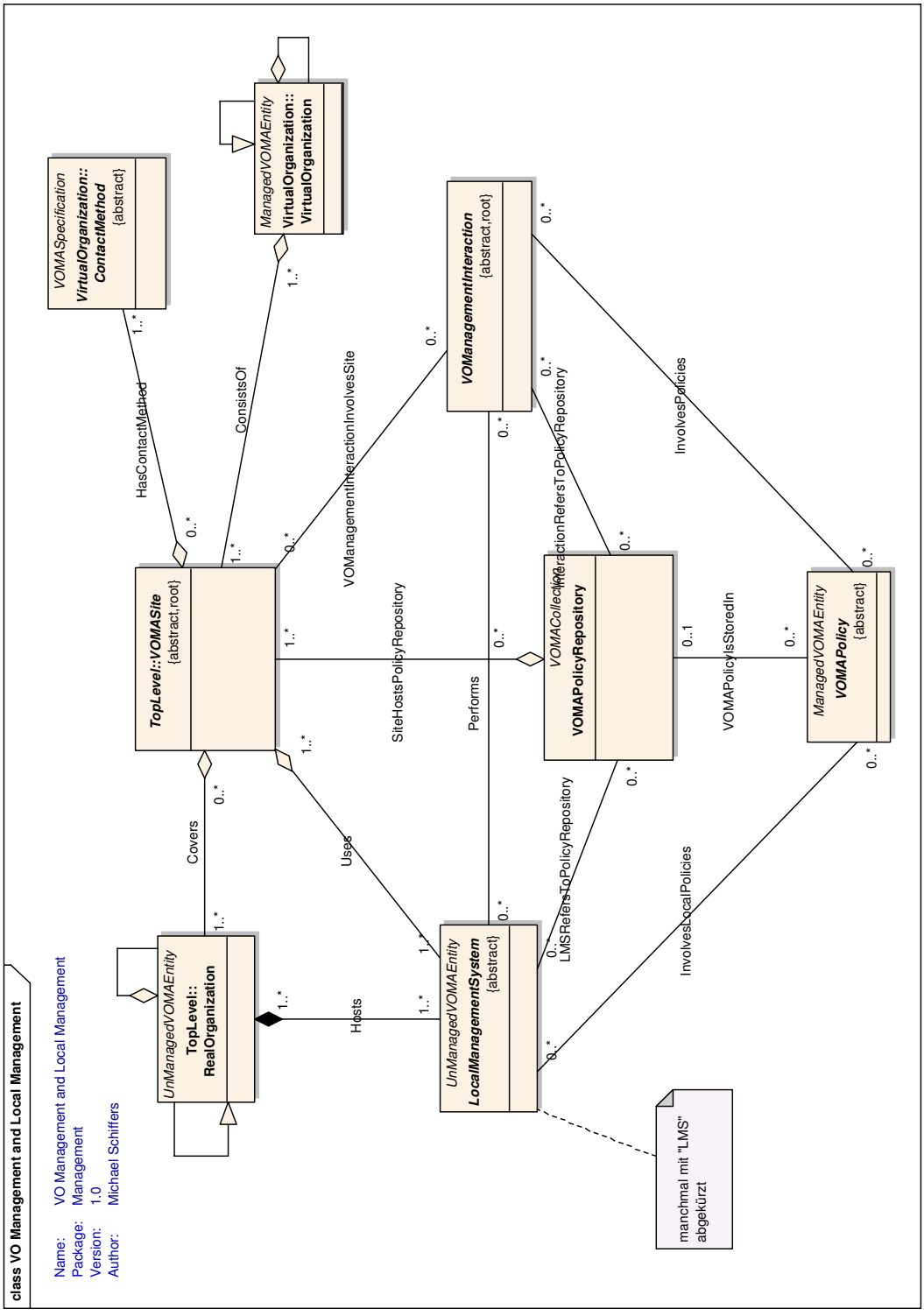


Abbildung 5.17: VO-Management und lokales Management

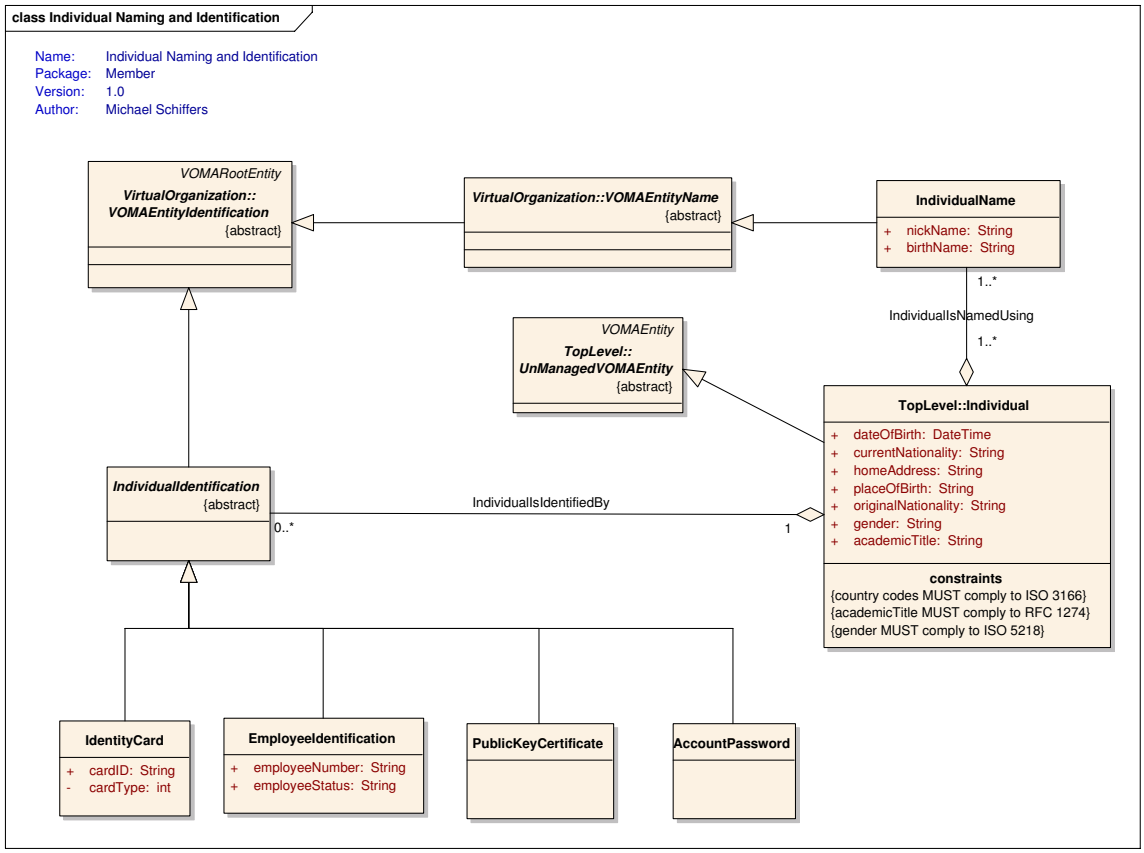


Abbildung 5.18: Namen und Identifizierung von Individuen

Namen und Identifizierung von Organisationen

Ähnlich wie Individuen können auch Organisationen über Namen angesprochen werden. Organisationsnamen erben die Eigenschaften der `VOMAEntityIdentification`-Klasse, protokollieren jedoch in einem zusätzlichen Attribut den Urheber der letzten Namensänderung (Attribut `nameChangedBy`). Abbildung 5.20 zeigt neben diesen Beziehungen auch, dass Organisationen über Kombinationen von Vollmachten (Klasse `CertificateOfAuthority`) und – speziell bei Unternehmen – Handelsregisterauszüge (Klasse `TradeRegisterCertificate`) authentifiziert werden können.

Beide Verfahren müssen jedoch insofern konsistent sein, als Handelsregisterauszüge und Vollmachten übereinstimmen müssen. Dies wird in der Klasse `AuthorizationDetails` modelliert. Die Vollmacht selbst muss von einer `TrustedEntity` ausgestellt sein, so dass die `RealOrganizationIdentification` letztlich von einer vertrauenswürdigen Entität bestätigt wird. Eine solche Autorität stellt im übrigen das Handelsregister selbst dar, da es öffentlichen Glauben genießt und Eintragungen als richtig und vertrauenswürdig gelten. Die eigentliche Identifizierung von Organisationen beruht damit auf einer öffentlichen Identi-

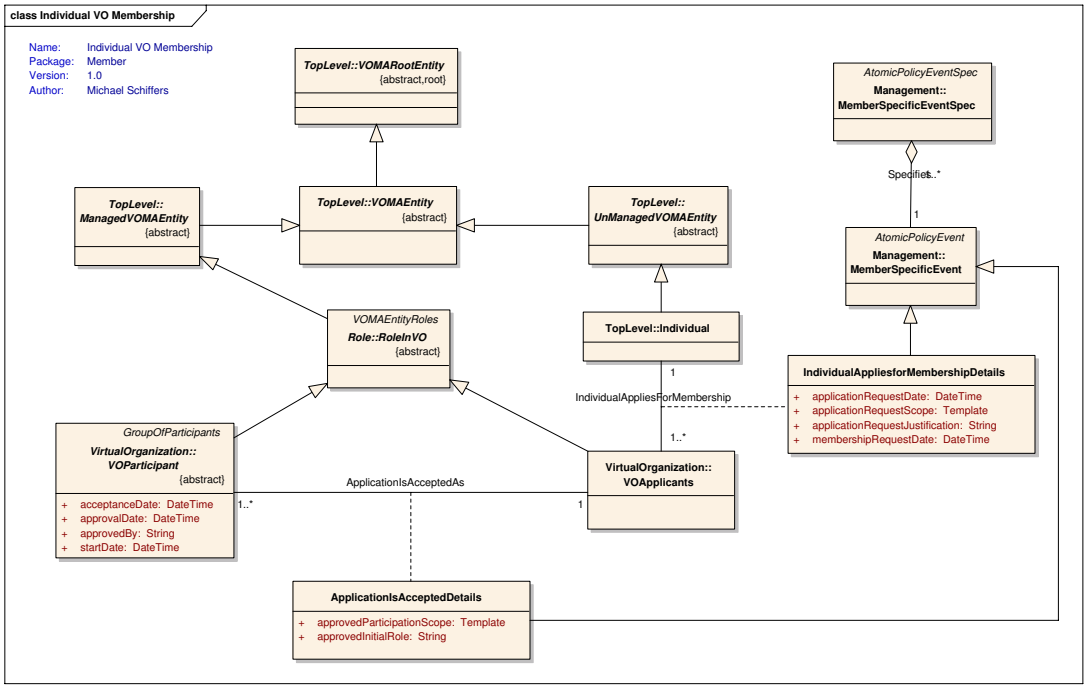


Abbildung 5.19: Individuelle VO-Migliedschaft

fizierungsnummer (z.B. der Handelsregisternummer), der Angabe der Registrierungsstelle, der ursprünglichen vertretungsberechtigten Personen (juristischer und natürlicher Art) und der aktuell vertretungsberechtigten Personen.

Organisationen werden außer durch die geerbten Attribute zusätzlich durch deren Gründungszeitpunkt, die aktuelle Adresse, mögliche Konzernzugehörigkeiten und die ursprüngliche und aktuelle Nationalität (im Sinne des Registrierungslandes) beschrieben. Gerade die letzten beiden Attribute haben durch ein verstärktes Sicherheitsbedürfnis an Bedeutung gewonnen, da nicht allen Nationalitäten gleiche Rechte eingeräumt werden, wie auch ein Hinweis zum „*Safe-Harbor*-Übereinkommen“ des DFN-Vereins unter `http://www.grid.lrz.de/res/globus/Anfrage_LRZ.doc` verdeutlicht.

Organisationale VO-Mitgliedschaft

Wie Individuen können auch Organisationen als solche die Mitgliedschaft zu einer VO beantragen oder sie werden dazu eingeladen. Konzeptionell werden beide Mitgliedschaften analog behandelt, wie Abbildung 5.21 zeigt.

Gruppierung von Mitgliedern

Wie die Szenarios im Abschnitt 3.1.1 zeigen, können Participants einer VO für bestimmte Zwecke Gruppen bilden (z.B. zur Durchführung einzelner Projekte in der D-Grid InGrid-

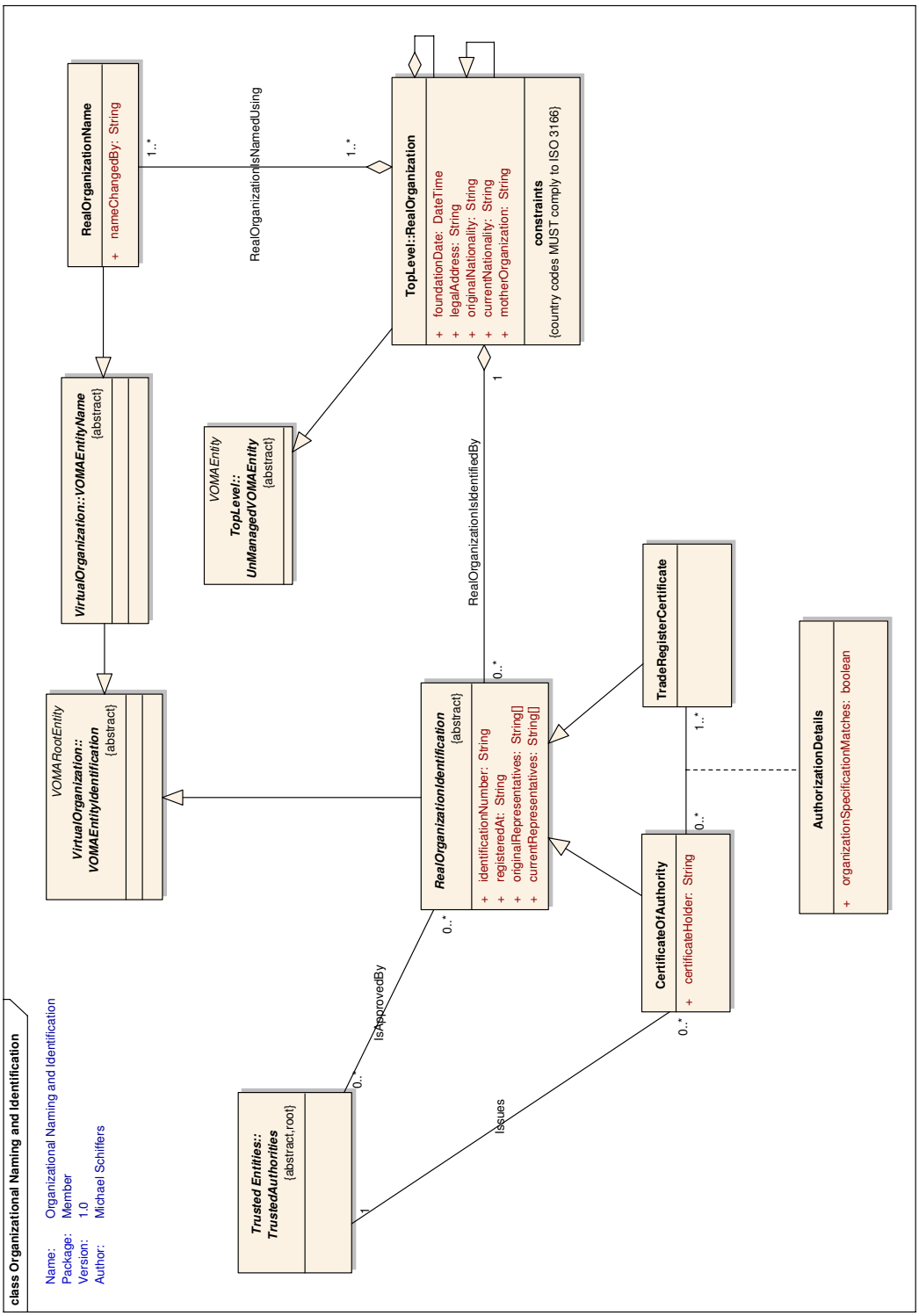


Abbildung 5.20: Namen und Identifizierung von Organisationen

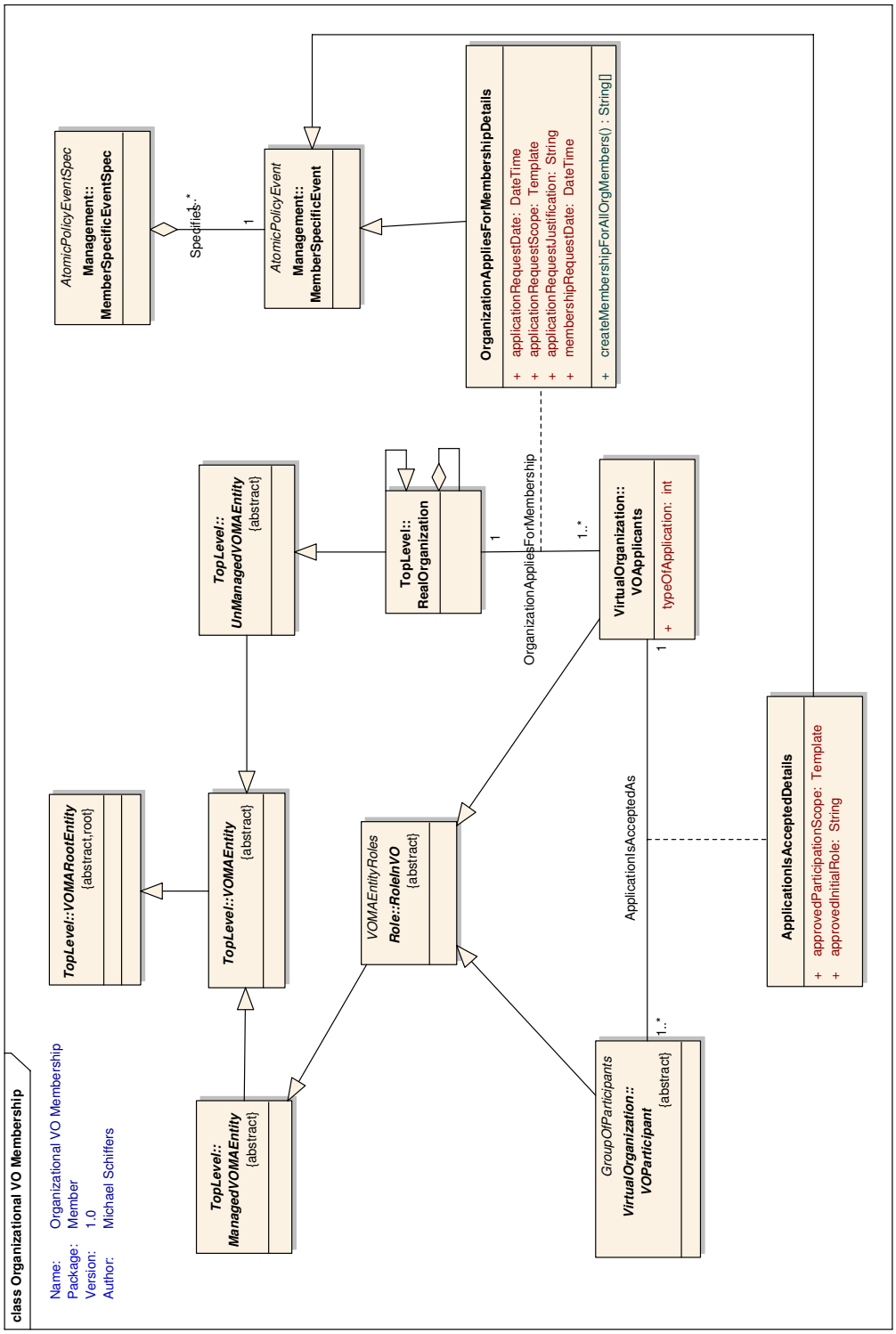


Abbildung 5.21: Organisationale VO-Mitgliedschaft

Community). Abbildung 5.22 zeigt die mit Gruppenbildungen zusammenhängenden Konzepte im Detail.

Wie Individuen und Organisationen werden auch Gruppen über Namen identifiziert werden, sie besitzen eine begrenzte Lebensdauer (Attribute `creationDate`, `validFor`). Gruppen können geschachtelt sein, wobei jede Komponente eine eigene Lebensdauer besitzt, die in die Berechnung der Gesamtlebensdauer über die Methoden `calculateMinTimeToLive()` und `calculateMinTimeToLive()` einfließt.

5.2.6 Die Domäne `Role`

Rollen werden im VO-Managemenmt zur Regelung von Zugriffen auf VO-Ressourcen und Dienste im Rahmen von Role Based Access Control (RBAC)-Verfahren [Ferraiolo u. Kuhn, 1992] genutzt, eine Anforderung, die aus den Szenarios abgeleitet wird. In VOs werden `Participants` (selbst wieder eine Rolle) auf Rollen abgebildet, wobei jedem `Participant` mehrere Rollen zugeordnet werden können und jede Rolle durch mehrere `Participants` wahrgenommen werden kann. Insofern können Rollen auch mit Gruppen assoziiert werden. Für die Verwaltung dieser Zuordnungen werden in der Regel Identity Management Systeme (IdM) eingesetzt [Hommel, 2007].

Namen und Identifizierung von Rollen

Rollen sind wie Individuen, Gruppen und Organisationen über Namen und spezielle VO-individuelle Verfahren identifizierbar, die in Abbildung 5.23 jedoch nicht separat aufgeführt sind. Die einfachste Art, Rollen zu identifizieren, ist der Nachweis deren Existenz über die Klasse `VOMAEntityIdentification`.

Einen interessanten Ansatz zur automatischen Identifizierung von Rollen über Hidden Markov Chains im Rahmen linguistischer Textanalysen schlägt im übrigen [Sigletos u. a., 2002] vor.

Rollen in VOs

Rollen besitzen in VOMA *Capabilities*, einem an CIM angelehnten Konzept zur Spezifikation von Berechtigungen und Verantwortlichkeiten [DMTF, 2006a], das hier auf Rollen erweitert wird. Rollen und Capabilities werden im Rahmen allgemeiner `VOMASpecifications` in den Klassen `VOMARoleSpecification` bzw. `VOMACapabilitiesSpecifications` spezialisiert (siehe Abbildung 5.24).

Die Lebensdauer einer Rolle innerhalb einer VO wird beschrieben durch deren Gründungszeitpunkt (Attribut `roleCreationDate`) und den Gültigkeitszeitraum (`validFor`). Ebenso wird in der Klasse `RoleInVo`, die die Rollen repräsentiert, hinterlegt, ob die Rolle zur Zeit besetzt ist (`isAssigned`) und wer der Rolle zugeordnet ist (`assignedMembers`).

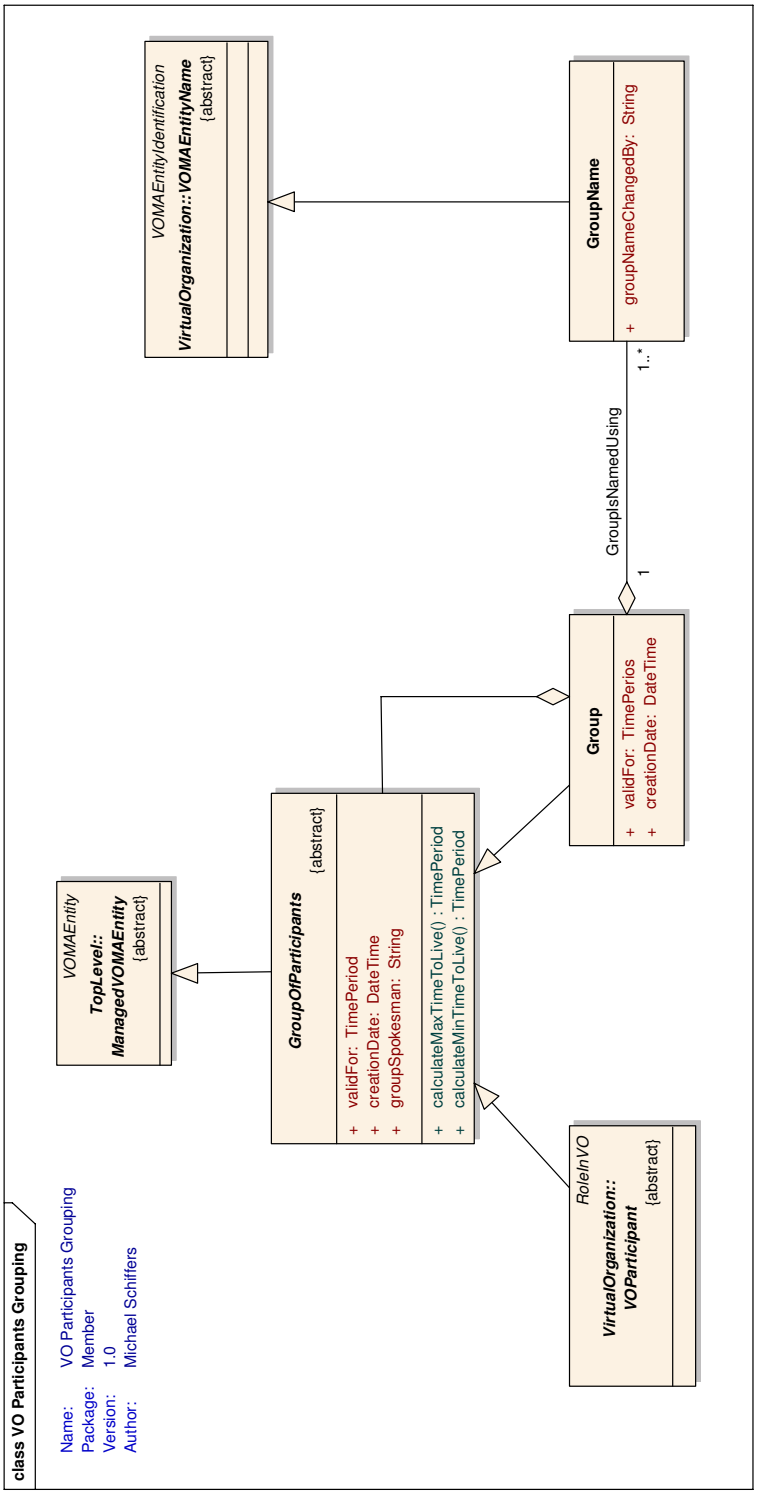


Abbildung 5.22: Gruppierung von Mitgliedern

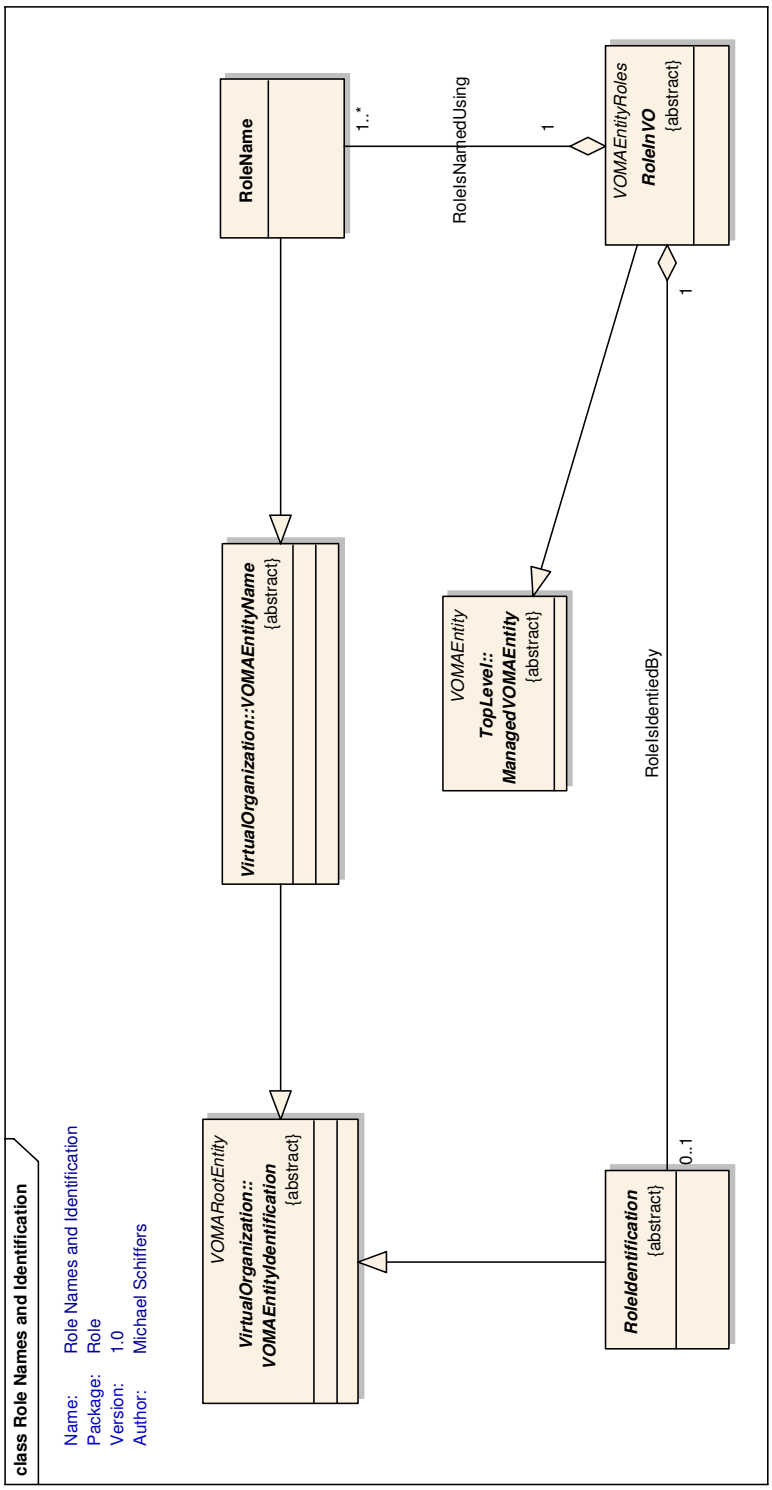


Abbildung 5.23: Namen und Identifizierung von Rollen

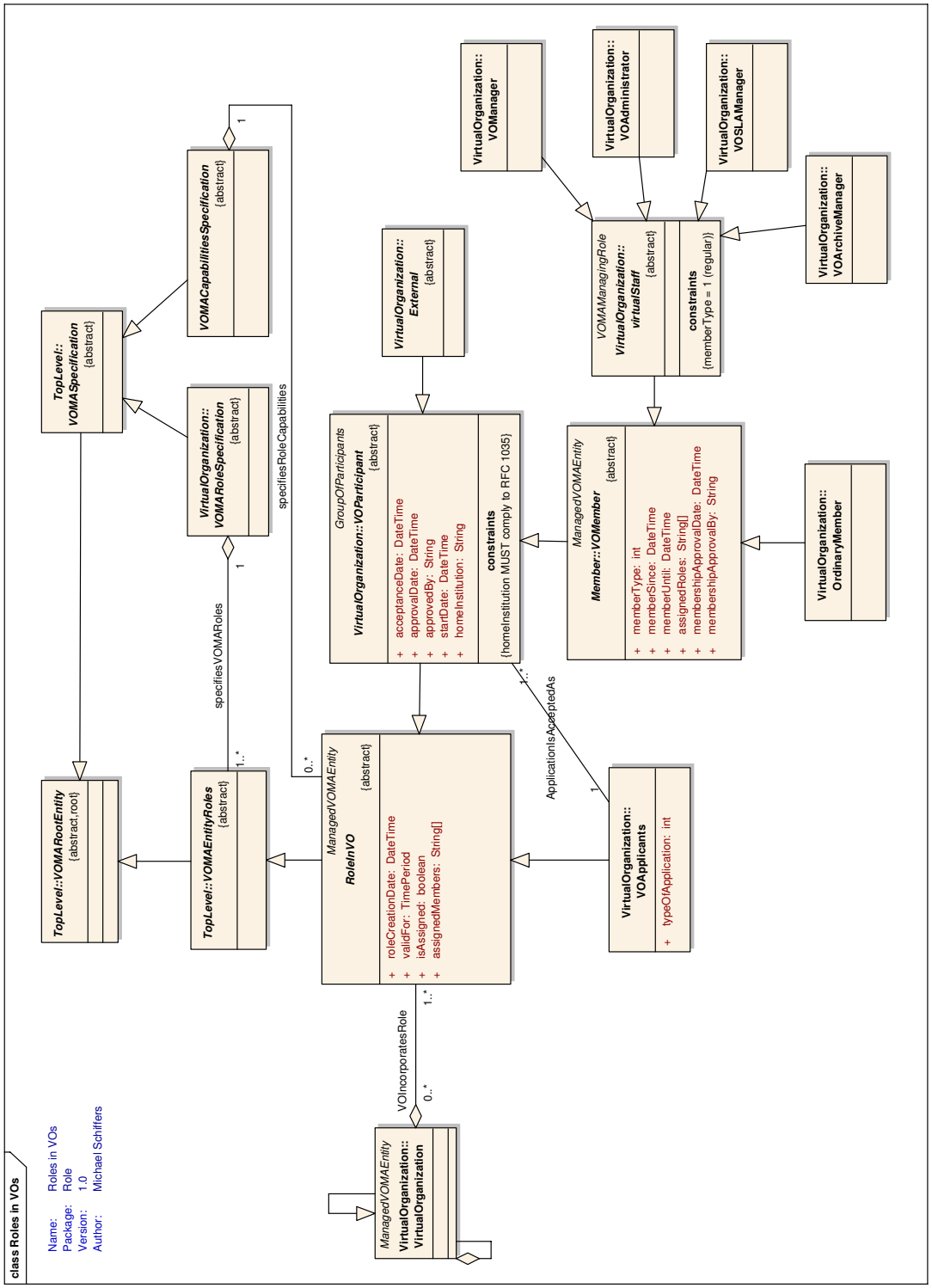


Abbildung 5.24: Rollen in VOs

Für das VO-Management relevante Rollen können in die grundsätzlichen Kategorien `VOApplicants` (will an der VO teilnehmen) und `VOParticipants` (darf an der VO teilnehmen) eingeordnet werden. `VOApplicants` können um die reguläre Teilnahme nachsuchen oder um einen Gast-Zugang (zum Beispiel die Test-User oder Gäste in DEISA). Diese Auswahl wird im Attribut `typeOfApplication` repräsentiert. Für einen akzeptierte Applicant wird in der Klasse `VOParticipant` die Genehmigung zur Teilnahme in den entsprechenden Attributen zusammen mit der Angabe der Heimat-Organisation des neuen Mitgliedes (Attribut `homeInstitution`) hinterlegt. `VOParticipants` können reguläre Mitglieder (Klasse `VOMember`) oder externe Teilnehmer (Klasse `External`) sein, beispielsweise in Form von Beratern oder Drittanbietern. Jedes Mitglied einer VO, repräsentiert in der Klasse `VOMember`, kann ein reguläres Mitglied sein (`ordinaryMember`) oder eine für das VO-Management relevant Rolle spielen (`virtualStaff`). Die in Abbildung 5.24) dargestellten Rollen des `virtualStaff` sind nur beispielhaft und sind nicht identisch mit den im Abschnitt 5.5 spezifizierten Rollen.

Rollen von VOs

Im Kontext des VO-Managements zeigen die beteiligten Entitäten ein komplexes Verhalten, indem sie verschiedene Rollen einnehmen können. Dies gilt auch für VOs selbst in Multi-VO-Umgebungen, wie sie beispielsweise im EmerGrid-Szenario, in IPY und im D-Grid angedacht sind. VOs können dabei als Provider virtueller Dienste oder virtueller Ressourcen (Klasse `VOAsProvider`) und als Verbraucher solcher Dienste bzw. Ressourcen auftreten (Klasse `VOAsConsumer`), aber auch Managementrollen übernehmen (Klasse `VOAsManager`) und als `VOParticipant` teilnehmen (Klasse `VOAsMember`). Abbildung 5.25 macht dies deutlich.

Beispiel: Im D-Grid nutzt eine VO *A* der AstroGrid-D-Community virtuelle Dienste oder virtuelle Ressourcen einer VO *B* der C3-Community. *A* tritt dann als Verbraucher der Ressourcen bzw. Dienste des Providers *B* auf. Gleichzeitig kann aber auch *A* Speicherkapazitäten für VOs in derselben Community bereitstellen, *A* fungiert damit auch als Provider und kann in der Rolle eines VO-Managers die Gründung einer Sub-VO initiieren.

Durch die Modellierung von Rollen als separate Entitäten (siehe Abbildung 5.25) können derartige Gegebenheiten formal sauber unter Verwendung des Role-Object-Patterns (ROP) [Fowler, 1996] repräsentiert werden. Der Gewinn ist dabei eine Trennung der Information über VOs von Information über Rollen und deren Beziehungen. VO-Rollen werden in der abstrakten Klasse `RoleOfVO` spezifiziert, die direkt von der `TopLevel`-Klasse `VOMAEntityRoles` abgeleitet ist. Rollen können in VOMA nach ihrem `roleType` kategorisiert werden, um intern orientierte Rollen (beispielsweise `VOAsMember`) von extern orientierten Rollen (beispielsweise `VOAsProvider`) zu unterscheiden. Zusätzlich wird im Attribut

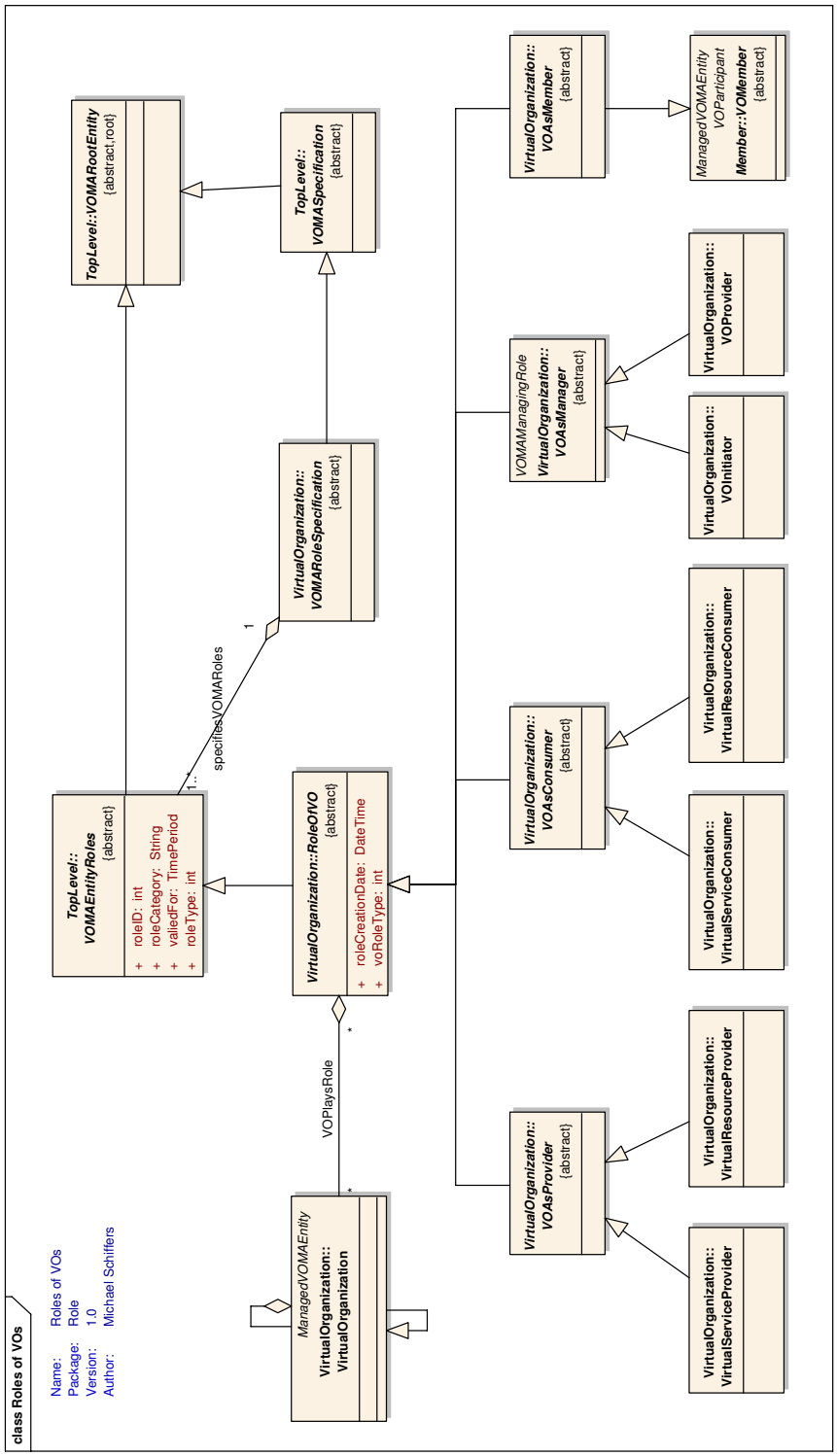


Abbildung 5.25: Rollen von VOs

`roleCategory` eine Gruppierungsmöglichkeit angeboten, VO-Rollen zu höherwertigen Rollen zu aggregieren (um beispielsweise sporadisch besetzte Rollen von kritischen Rollen zu trennen). Rollentypen besitzen wie VOs selbst eine begrenzte Lebensdauer – ausgedrückt im Attribut `validFor`. Der Typ der spezialisierten VO-Rolle wird analog zu [TMF, 2007] im Attribut `voRoleType` kodiert. Im übrigen sei an dieser Stelle noch angemerkt, dass – anders als in realen Organisationen – Virtuelle Organisationen kein Linienmanagement kennen und daher auch keine Klassen für die Modellierung von Positionen notwendig sind (siehe auch Abschnitt 2.1).

5.2.7 Die Domäne `VirtualResource`

VOs stellen virtuelle Ressourcen bereit (siehe Abschnitt 3.1.2), die von den Mitgliedern (`VOMember`) und anderen VOs genutzt werden können. Virtuelle Ressourcen werden aus realen Ressourcen, die von realen Organisationen (als Quota-Provider im Abschnitt 3.2.1) der VO entweder komplett oder in Kontingenten zur Verfügung gestellt werden, aggregiert. Nur virtuelle Ressourcen werden von der VO verwaltet, nicht jedoch reale Ressourcen. Reale Ressourcen können dabei physische Ressourcen sein (Rechner, Printer, Speicherkapazitäten) oder logische Ressourcen (Lizenzen, Software). Abbildung 5.26 zeigt diese Zusammenhänge genauer.

Eine virtuelle Ressource (Klasse `VirtualResource`) kann als *managed object* über einen Namen (Attribut `virtualResourceName`) und eine Beschreibung (`resourceDescription`) identifiziert werden. Als virtuelle Ressource besitzt auch sie eine beschränkte Lebensdauer, ausgedrückt durch die Attribute `validFor` und `creationDate`. Die Lebensdauer der Ressource kann jedoch die Lebensdauer der VO nicht überschreiten. Im `informationServiceLink` wird zusätzlich über einen Uniform Resource Identifier (URI), einem VOMA Basistyp, spezifiziert, welcher Managementdienst die für die Ressource relevanten Statusinformationen liefert. Dies kann in einer Globus GT4-spezifischen Instanziierung beispielsweise die URI der Globus MDS-Komponente sein.

Administrativ wird eine virtuelle Ressource von einer oder mehreren `VOMASites` bereitgestellt und im Konzert mit lokalen Managementsystemen betrieben. Der Zugang und die Verwendung von virtuellen Ressourcen wird über VO-weite und lokale Policies geschützt, ausgedrückt in den Klassen `VRSpecificEvent` und `VRSpecificEventSpec`.

Virtuelle Ressourcen sind, wie Abbildung 5.27 zeigt, entweder rechnende Ressourcen (Klasse `ComputingResource`) oder Speicherressourcen (Klasse `StorageResource`), die typspezifische Attribute besitzen und über die Assoziation `ResourceBinding` verbunden sind.

5.2.8 Die Domäne `VirtualService`

Neben virtuellen Ressourcen stellen VOs auch virtuelle Dienste bereit (siehe Abschnitt 3.1.2), die von den Mitgliedern und anderen VOs im Rahmen VO-spezifischer Policies

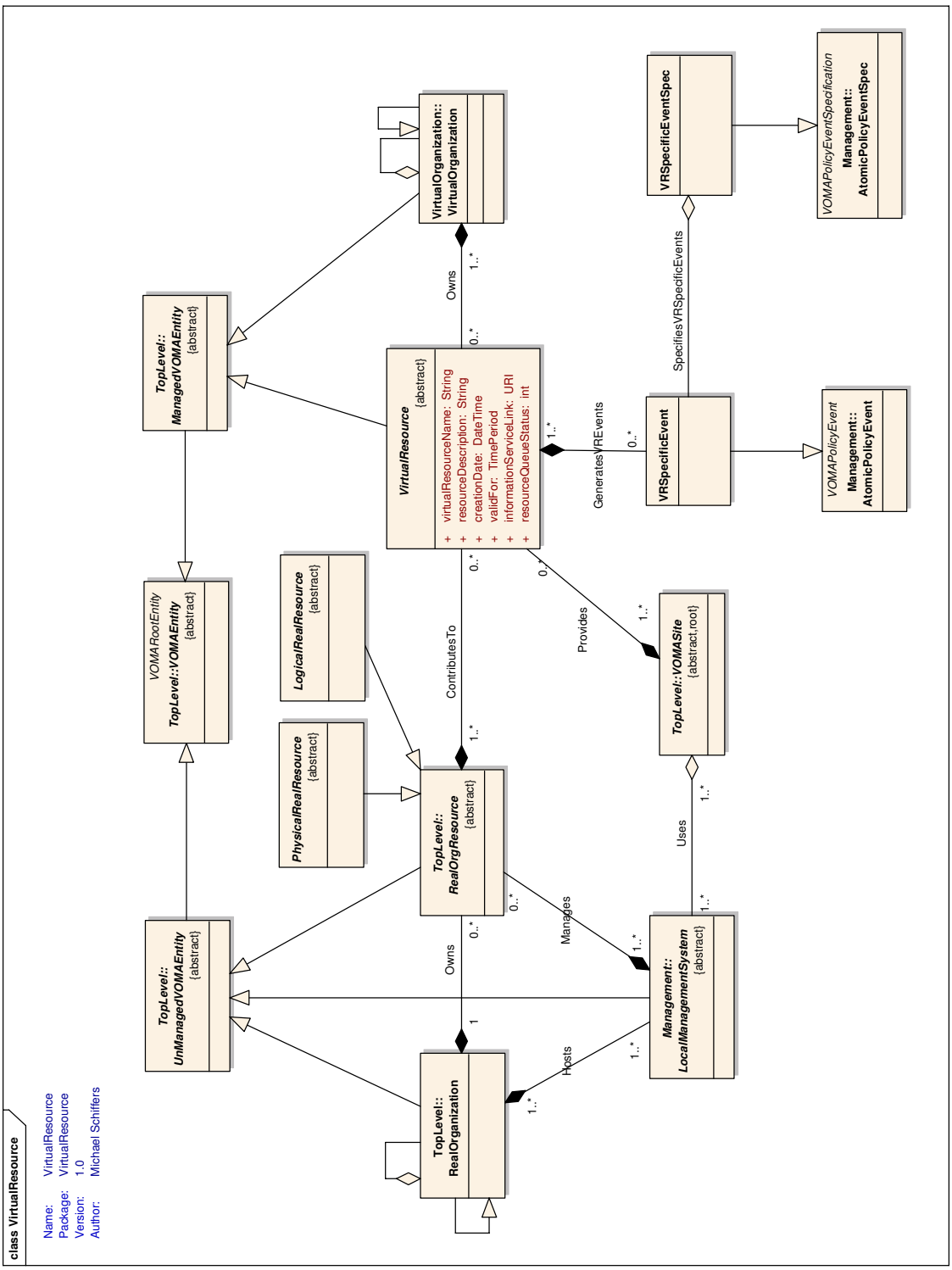


Abbildung 5.26: Virtuelle Ressourcen

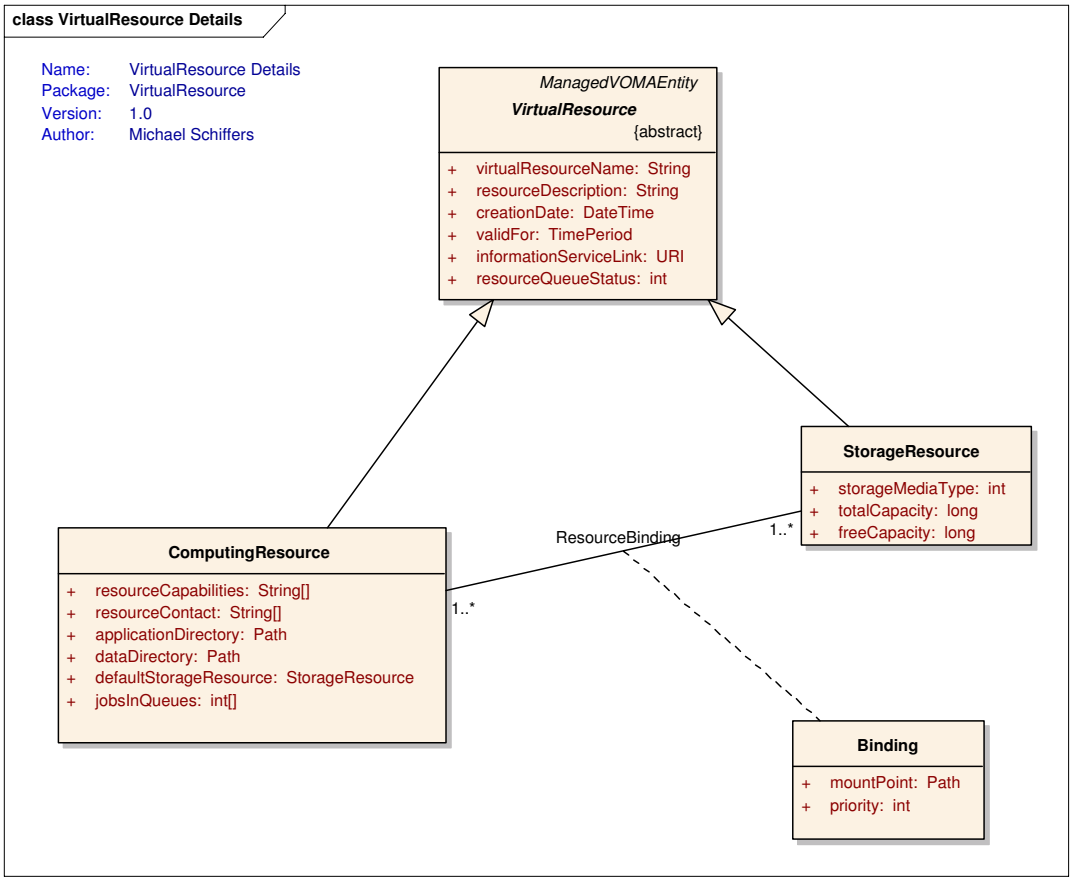


Abbildung 5.27: Typen virtueller Ressourcen

genutzt werden können. Anders als bei virtuellen Ressourcen werden virtuelle Dienste nicht von realen Organisationen an die VO delegiert, sie werden allerdings aus realen Diensten komponiert oder aggregiert. Abbildung 5.28 zeigt diese Zusammenhänge genauer.

Neben virtuellen Ressourcen stellen VOs virtuelle Dienste bereit (siehe Abschnitt 3.1.2), die von den Mitgliedern (`VOMember`) und anderen VOs genutzt werden können. Virtuelle Dienste werden aus realen Diensten (Klasse `RealOrgService`) und Ketten virtueller Dienste komponiert. Abbildung 5.28 zeigt diese Zusammenhänge genauer. Ein virtueller Dienst (Klasse `VirtualService`) kann als *managed object* über einen Namen (Attribut `serviceName`) und eine Beschreibung (`serviceDescription`) identifiziert werden. Als virtueller Dienst besitzt er eine beschränkte Lebensdauer, ausgedrückt durch die Attribute `validFor` und `creationDate`. Die Lebensdauer des Dienstes kann jedoch die Lebensdauer der VO nicht überschreiten.

Virtuelle Dienste können von der VO als `VOAsProvider` im Attribut `serviceType` typisiert und im Attribut `serviceVersion` versioniert werden. Sie sind zudem Status-behaftet, ausgedrückt in den Belegungen „OK“, „Warning“, „Critical“, „Unknown“ und „Other“

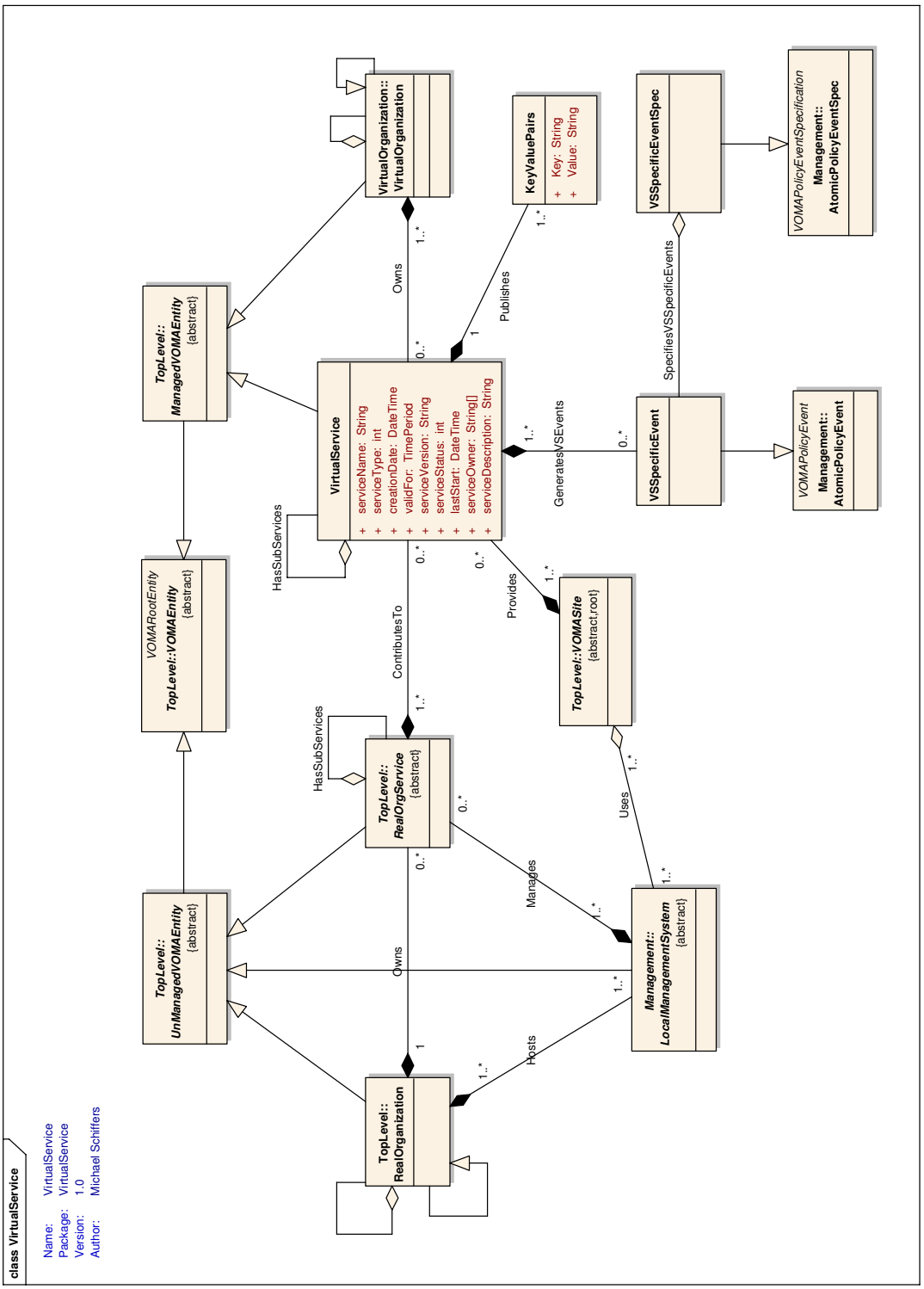


Abbildung 5.28: Virtuelle Dienste

des Attributes `serviceStatus`. Virtuelle Dienste publizieren dienstspezifische Informationen als Schlüssel-Wert-Paare (Klasse `KeyValuePairs`). Administrativ wird ein virtueller Dienst wie eine virtuelle Ressource von einer oder mehreren `VOMASites` bereitgestellt und im Konzert mit lokalen Managementsystemen betrieben. Der Zugang und die Verwendung von virtuellen Diensten wird ebenso über VO-weite und lokale Policies geschützt. Dies wird in den Klassen `VSSpecificEvent` und `VSSpecificEventSpec` ausgedrückt.

5.2.9 Die Domäne `Trusted Entities`

Hauptsächlich im Rahmen der Authentifizierung von Individuen und Organisationen werden neutrale, vertrauenswürdige Autoritäten zur Beglaubigung von Identitäten genutzt. Diese werden gemäß Abbildung 5.29 in der Klasse `TrustedAuthorities` und ihren Spezialisierungen `Notary` und `Escrow` repräsentiert.

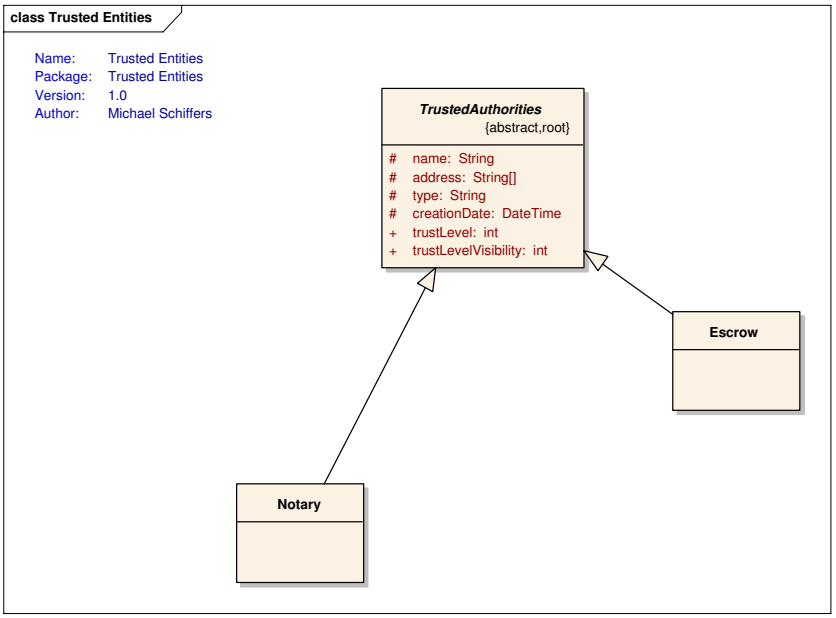


Abbildung 5.29: Trusted Entities

5.2.10 Interoperabilität mit verwandten Schemata

Das VOMA-Informationsmodell (die VO-MIB) ist auf das Management Virtueller Organisationen in Grids ausgerichtet und spezifiziert daher die Entitäten und deren Beziehungen, die in diesem Umfeld von Bedeutung sind und mit lokalen Managementarchitekturen und -prozessen assoziiert werden müssen. Dies wird nicht zuletzt durch Abbildung 5.17 verdeutlicht. Eine wesentliche Voraussetzung für eine Grid-weite Verwendbarkeit und Verwendung von VOMA ist daher deren problemlose Interoperabilität mit bereits verwendeten Schemata bzw. existierenden Standards. Eine Analyse bestehender Grid-Projekte in [Saeki

u. a., 2006] zeigt die Relevanz von CIM und GLUE in der ersten Kategorie, die Roadmap des OGF-Referenzmodells [Strong, 2007] unterstreicht zusätzlich die Bedeutung des SID-Modells des Telemanagement Forums für die zweite Kategorie. Im Folgenden wird deshalb exemplarisch dargestellt, wie das VOMA-Informationsmodell auf diese drei Schemata abgebildet werden kann.

Interoperabilität von VOMA und CIM

CIM [DMTF, 2007] stellt ein objektorientiertes Managementinformationsmodell mit dem Ziel dar, einen plattform- und technologieunabhängigen Austausch von Managementinformationen Netzwerke, Systeme und Dienste betreffend zu unterstützen und damit die Kooperation von Managementsystemen zu gewährleisten (siehe auch Kapitel 4). Diesem Zweck dienen im CIM-Standard das Core Model (siehe Abbildung 4.2), dessen Erweiterung im Common Model und technologiespezifische Erweiterungschemata.

Einer Darstellung der Integration des VOMA-Schemas in das CIM-Schema sind zunächst einige grundsätzliche Bemerkungen voran zustellen:

- CIM verwendet einen strikten Spezialisierungs- oder Subklassenansatz, dem sich auch eine Erweiterung des Schemas um VOMA-Konstrukte unterzuordnen hat. Dieser Ansatz führt zu einem sehr komplexen und aufwändigen Erweiterungsmechanismus [Keller u. a., 2001] und ist ohne die Basis eines streng angewandte Regelwerkes nicht leicht durchzuführen [Kempter, 2004]. Im VOMA-Modell wird dagegen ein Spezifikationsorientiertes Modellierungsparadigma favorisiert, das wesentlich leichter zu erweitern ist, damit aber den Nachteil in Kauf nimmt, bestehende Klassen nach Spezifikationsänderungen unter Umständen neu definieren zu müssen..
- VOMA verwendet in vielen Bereichen das UML-Konstrukt einer Assoziationsklasse. Dies ist in CIM unbekannt.
- VOMA basiert vollständig auf UML-Konstrukten, CIM verwendet dagegen in einigen Bereichen einen eigenen „UML-Dialekt“ und zusätzliche Beschreibungen im Managed Object Format (MOF).
- Im VOMA-Modell wird explizit eine „Ownership“-Assoziation vorgesehen, CIM kennt diese nicht.
- CIM definiert zwar Klassen zur Modellierung von Organisationen innerhalb seines User Models, diese werden jedoch weder als *managed objects* betrachtet noch gestatten sie die Behandlung der speziellen Aspekte Virtueller Organisationen.
- Der Fokus des CIM-Schemas liegt auf der Betriebsphase. Insofern sind Lebenszyklusunterstützende und Managementinteraktion-orientierte Konstrukte – wenn überhaupt – nur rudimentär vorhanden.

Eine Integration des VOMA-Schemas in CIM würde vor diesem Hintergrund neben anderen Maßnahmen erfordern:

1. VOs als CIM `ManagedElement` aufzufassen,
2. VOs und die CIM `OrganizationalEntity` zu assoziieren,
3. die CIM `UserEntity` zu spezialisieren und mit CIM `Roles` zu assoziieren,
4. `VOMACollections` und `VOMASpecifications` als CIM `ManagedElement` zu betrachten,
5. die VOMA `UnManagedEntities` und deren Assoziationen in CIM zu integrieren,
6. die VOMA Assoziationsklassen auf andere CIM-kompatible UML-Konstrukte abzubilden.

Während Anforderung 1 leicht durch entsprechendes *Sub-Classing* zu erfüllen ist, sind alle anderen Anforderungen nur mit manuellem Aufwand zu realisieren, der im Rahmen dieser Arbeit allerdings nicht geleistet werden kann.

Interoperabilität von VOMA und SID

SID stellt ein Informationsmodell dar, das auf die Unterstützung von Managementprozessen fokussiert, allerdings wegen seiner Provenienz primär auf die Anforderungen des Telekommunikationsbereiches zugeschnitten ist [Brenner u. a., 2006]. Auch wenn SID auf den Grundprinzipien des CIM-Schemas basiert, weicht es doch nicht unerheblich davon ab, um eine klare Trennung zwischen System- und Geschäftsprozess-orientierten Managementinformationen zu treffen. Diese Unterscheidbarkeit liegt auch dem VOMA-Ansatz zu Grunde, indem VO-bezogene Managementinteraktionen separat modelliert werden.

SID kann zwar die Schwächen von CIM durch eine mächtigere Ausdrucksfähigkeit kompensieren (z.B. mit der Verwendung von Zustandsautomaten zur Spezifikation von Lebenszyklen), kennt aber auch weder den Begriff einer Virtuellen Organisation noch das Beziehungsgeflecht zwischen VOs und anderen Entitäten wie Rollen, Mitgliedern und VO-Policies. Dennoch ist eine Einordnung des VOMA-Schemas in SID sehr viel einfacher zu gestalten als in CIM, da beide Ansätze auf den gleichen Modellierungsparadigmen fußen. Abbildung 5.30 zeigt am Beispiel der SID Policy Domäne, wo SID angepasst werden muss (die gestrichelt umrandeten Bereiche) bzw. erweitert werden muss (die grau hinterlegten Bereiche). Auch hier kann im Rahmen dieser Arbeit diese Abbildung nicht geleistet werden.

Interoperabilität von VOMA und GLUE

Das GLUE-Schema stellt kein allgemeines Informationsmodell im Sinne einer Managementarchitektur dar (siehe auch Kapitel 4). Der Fokus liegt auf der Modellierung von Grid-Ressourcen und -diensten. Insofern sind in GLUE auch keine Konstrukte zur Modellierung von VOs und deren Beziehungen zu Rollen, Mitgliedern, Policies, etc. vorhanden. Mit dem Site-Konzept kennt GLUE allerdings administrative Domänen für virtuelle Ressourcen und Dienste. Wegen dieser Beschränkung ist die Frage damit nicht, ob und wie VOMA auf GLUE abgebildet wird, sondern wie GLUE-Entitäten in VOMA integriert werden können. Dies zeigt die folgende Tabelle 5.9.

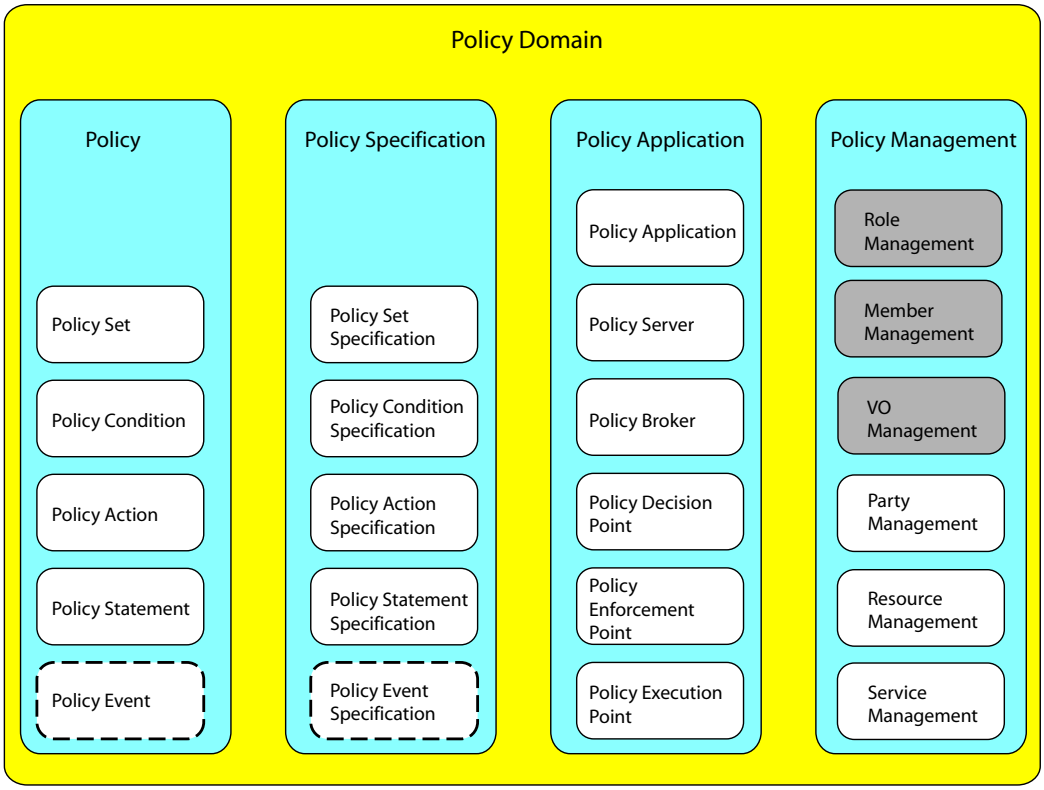


Abbildung 5.30: Erweiterung der SID Policy Domains [TMF, 2007]

Alle anderen GLUE-Klassen wie `Host`, `Cluster`, oder `OperatingSystem` werden in VOMA nicht benötigt und können daher beliebig instanziiert werden. Die in GLUE vorhandenen Klassen `VOView` und `VOInfo` sind mit den Klassen `ComputingElement` respektive `StorageArea` assoziiert. Diese Assoziationen werden in VOMA wegen der VO-Management-Sicht direkt als Attribute virtueller Ressourcen und Dienste modelliert.

GLUE-Klasse	VOMA-Klasse
`CESEBind`	`Binding`
`ComputingElement`	`ComputingResource`
`ServiceData`	`KeyValuePairs`
`Location`	`RealOrganization`
`StorageElement`	`StorageResource`
`VOInfo`	`VirtualResource`
`VOView`	`VirtualResource`
`Service`	`VirtualService`
`Site`	`VOMASite`

Tabelle 5.9: Abbildung der GLUE-Klassen auf VOMA-Klassen

5.2.11 Zusammenfassung des Informationsmodells

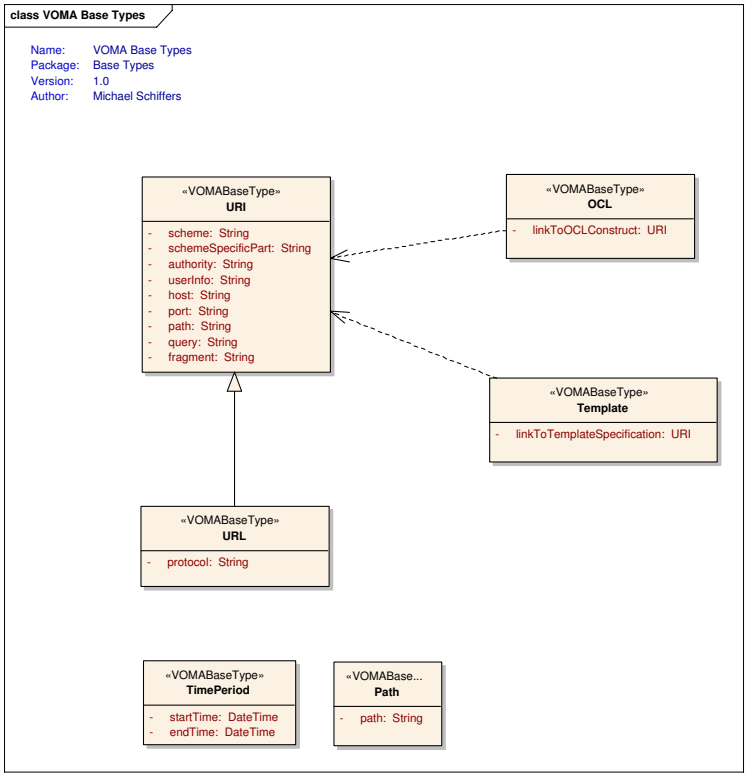


Abbildung 5.31: VOMA Basistypen

In diesem Abschnitt wurde das VOMA-Informationsmodell spezifiziert. Ausgehend von den Anforderungen des Kapitels 3 wurden mit `VirtualOrganization`, `VOMember`, `RoleInVO`, `VirtualResource`, `VirtualService` und `VOMAPolicy` die zentralen Entitäten des Modells und deren Beziehungsgeflecht dargestellt und in Modelldomänen angeordnet. Die Anhänge A und B enthalten die vollständigen Listen der im Informationsmodell verwendeten Domänen und deren Klassen im Detail unter Verwendung der VOMA Basistypen gemäß Abbildung 5.31.

Abbildung 5.32 zeigt einen kompletten Überblick über die verwendeten Klassen und deren Generalisierungsabhängigkeiten. Deutlich ist die Prominenz einiger zentraler Klassen zu erkennen.

Abbildung 5.33 zeigt dagegen die wesentlichen Assoziationsabhängigkeiten der VOMA-Klassen.

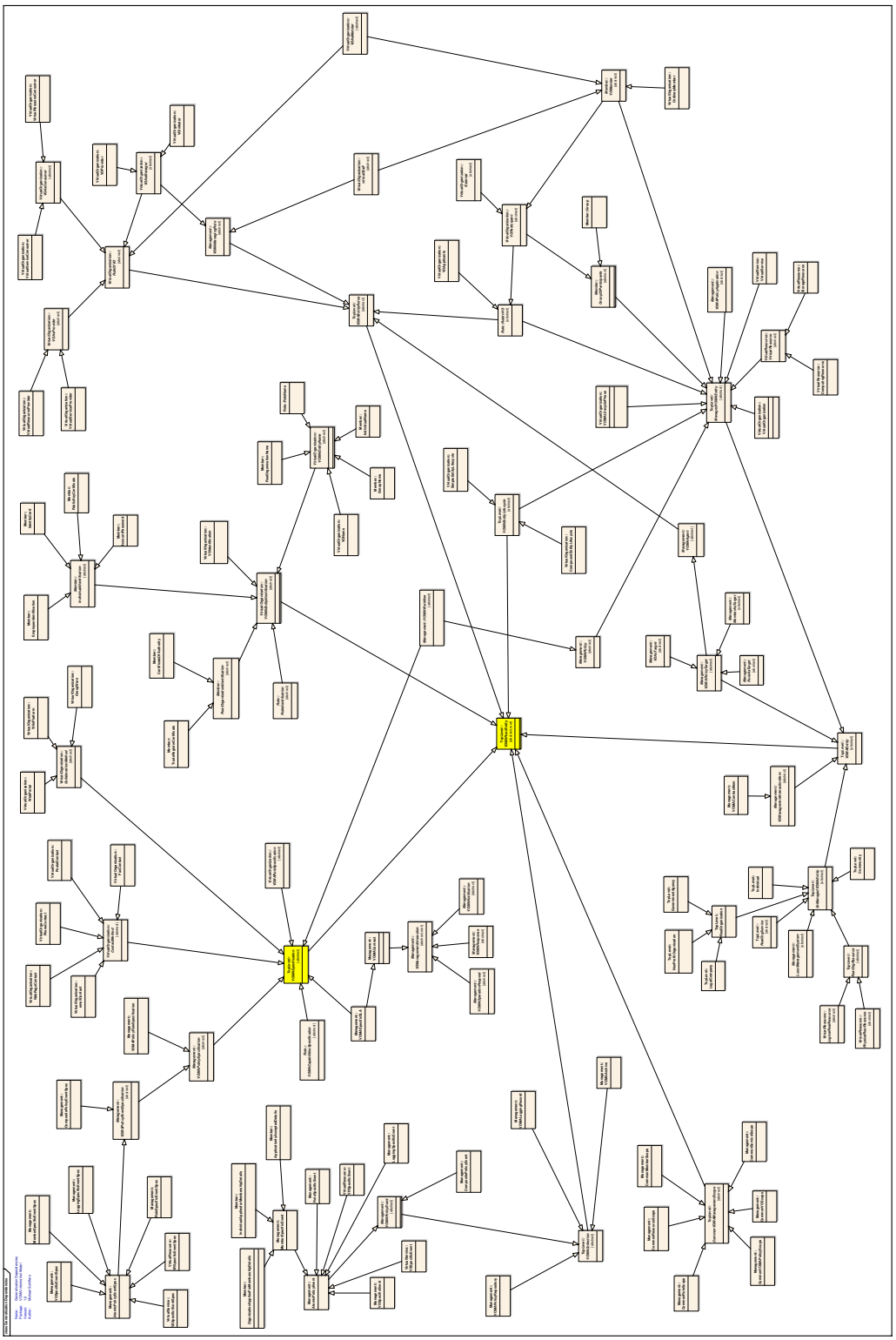


Abbildung 5.32: Generalisierungsnetzwerk der VOMA-Klassen

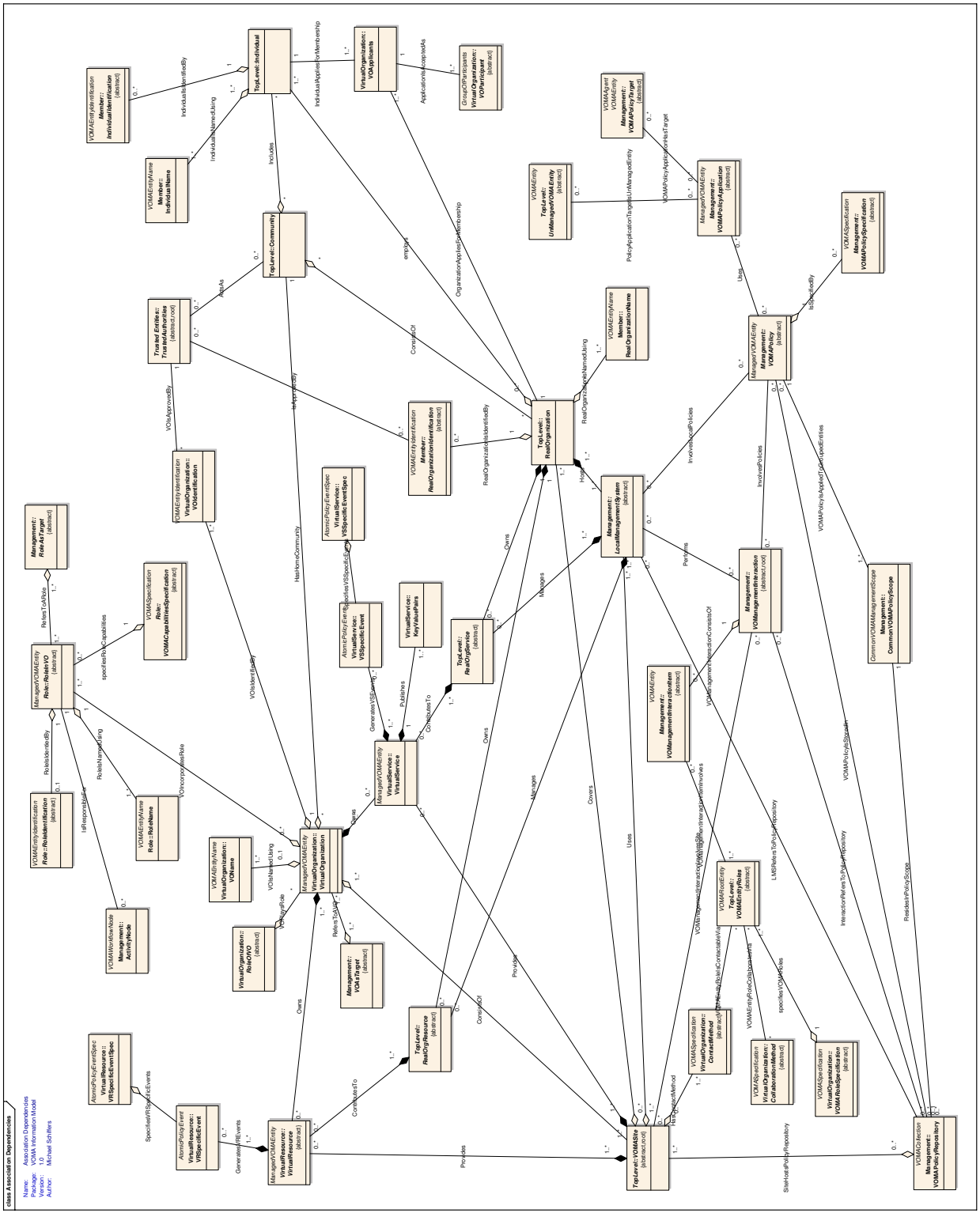


Abbildung 5.33: Assoziationsnetzwerk der VOMA-Klassen

5.3 Entwurf des Organisationsmodells

Im Kontext von Managementarchitekturen adressiert das Organisationsmodell den Teil der Architektur (siehe auch Abschnitt 2.3.1), der die involvierten Akteure, ihr Rollenspiel und die Grundprinzipien ihrer Kooperation im Rahmen konkreter Zielvorgaben (Policies) beschreibt [Hegering u. a., 1999]. Diesem Zweck dient auch das VOMA-Organisationsmodell. Im Folgenden werden daher neben den am VO-Management beteiligten Rollen auch deren Interaktionen festgelegt und einem *Managementdomänen*-Konzept untergeordnet. Obwohl im Abschnitt 3.2.1 wesentliche Aktoren schon informell im Rahmen der Anforderungsspezifikation eingeführt wurden und im Abschnitt 5.2 bereits formal im VOMA-Informationsmodell verankert wurden, erfordert ein auf das Management Virtueller Organisationen ausgerichtetes Organisationsmodell weitergehendere Festlegungen, denen dieser Abschnitt gewidmet ist. Dazu gehören insbesondere

1. die Festlegung eines (administrativ orientierten) Domänenkonzepts für das VO-Management zur Konkretisierung der Klassen `CommonVOMAPolicyScope` und `CommonVOScope` im Informationsmodell (Abschnitt 5.3.1),
2. die Spezifikation der am VO-Management beteiligten Rollen und deren Verteilung auf die Domänen (Abschnitt 5.3.2) und
3. die Beschreibung der Interaktionen zwischen den Rollen sowohl innerhalb einer Domäne als auch domänenübergreifend (Abschnitt 5.3.3).

5.3.1 Definition der Managementdomänen

Die Festlegung der Managementdomänen orientiert sich an dem in Abbildung 3.11 auf Seite 96 dargestellten Ebenenkonzept. Jeder Ebene wird für das Organisationsmodell in kanonischer Weise eine eigene administrative Domäne [Hegering u. a., 1999] zugeordnet, da sich die auf diesen Ebenen befindenden Managementobjekte und -prozesse jeweils einer gemeinsamen Verwaltungshoheit unterordnen. Der Community-Ebene wird dabei die Domäne `CommunityDomain` zugeordnet, der Ebene der realen Organisationen die Domäne `RealOrganizationDomain` und der Ebene der Virtuellen Organisationen die Domäne `VirtualOrganizationDomain`. Abbildung 5.34 macht dies deutlich.

5.3.2 Definition der VO-Managementrollen

Unter einer *Rolle* wird im Kontext des VOMA-Organisationsmodells ein definierter Aufgabenumfang verstanden, der entsprechend dem zuvor beschriebenen Informationsmodell von den an der VO beteiligten Organisationen, den die VO einbettenden Communities, einer Person, oder ganz allgemein einer Entität, wahrgenommen bzw. bereitgestellt wird (siehe auch Abbildung 3.11). Entitäten sind nicht an feste Rollen gebunden, sondern können gleichzeitig mehrere Rollen einnehmen. Dies wird im Informationsmodell durch die

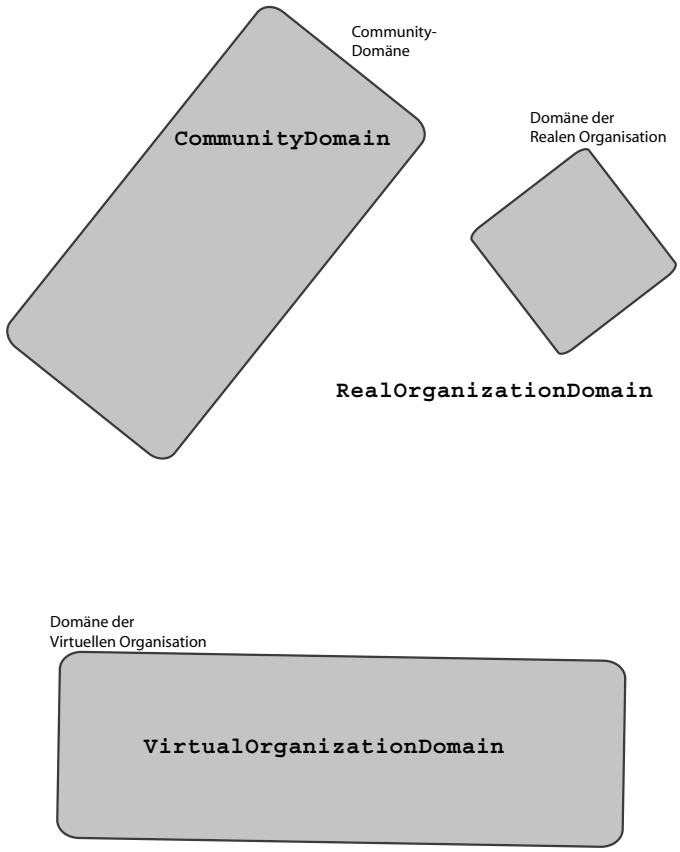


Abbildung 5.34: Domänen des Organisationsmodells

Klasse `VOMAEntityRoles` und deren Assoziationen und Subklassen ausgedrückt. Der Zweck des VOMA-Rollenkonzeptes liegt in der Abstraktion von der konkreten Durchführung einer VO-Managementinteraktion, indem die Art und Weise, wie eine Entität den Funktionsumfang erbringt, verschattet wird. Im VOMA-Informationsmodell werden Rollen über entsprechende Spezifikationen beschrieben (siehe Abbildung 5.35 für einen entsprechenden Auszug aus dem VOMA-Informationsmodell), in denen durch die Angabe der rollenspezifischen *Capabilities* die Aufgaben bzw. Funktionen bestimmt werden, die die Rollen wahrnehmen bzw. bereitstellen. Damit wird eine der Grundanforderungen an VOMA erfüllt, nämlich die logische Trennung der VO-spezifischen Aspekte von den lokalen Managementaspekten.

Die im Rahmen des VO-Managements involvierten Rollen können direkt aus den Anforderungen im Kapitel 3 und den dort identifizierten Aktoren (Abschnitt 3.2.1) bzw. Anwendungsfällen (Abschnitt 3.2.2) abgeleitet werden. Da VOMA als Bindeglied zwischen den VOs initiierenden Communities, an VOs beteiligten realen Organisationen und den eigentlichen VOs dient, stellt VOMA diesen Entitäten eine managementorientierte Sichtweise auf VOs bereit. Insofern können die am VO-Management beteiligten Rollen dem Managementbereich der Communities, den Managementbereichen der realen Organisationen (im

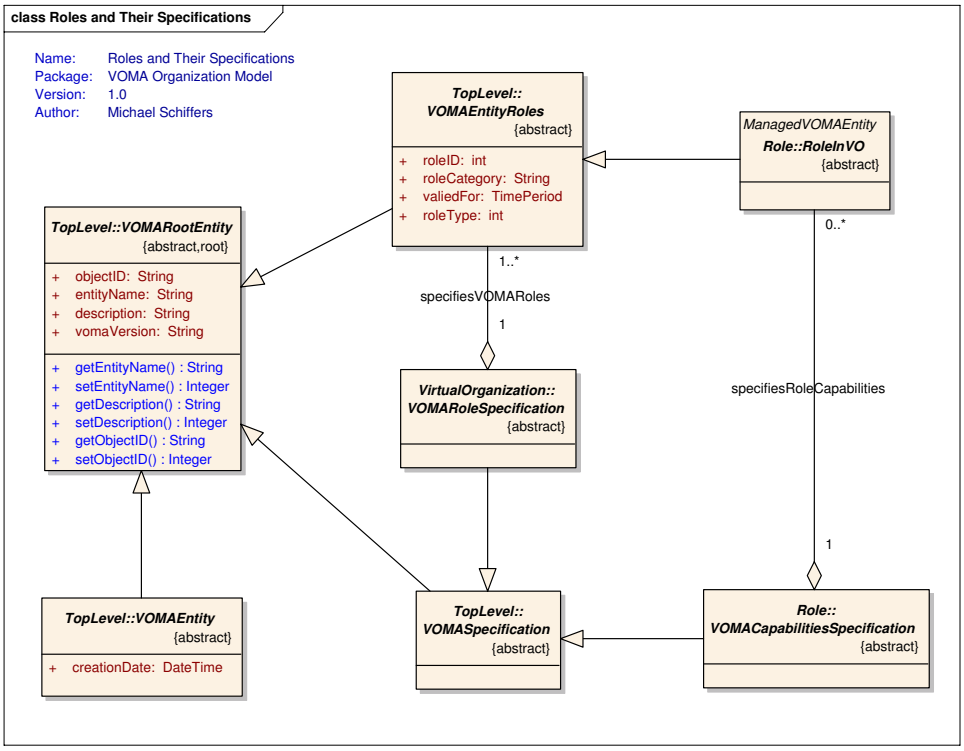


Abbildung 5.35: Rollen und Rollenspezifikation

erweiterten Sinn) und den VOs selbst zugeordnet werden. In den folgenden Teilabschnitten wird jeweils einer dieser Aspekte näher diskutiert.

Rollen der Communities

Die auf Seiten der Communities (als *breeding environments*) am VO-Management beteiligten Rollen haben die Aufgabe, das Gründungsumfeld einer VO so aufzubereiten, dass die VO initiiert, initialisiert und gestartet werden kann, da der Impuls zur Formation von VOs stets aus den Communities erfolgt (siehe die Szenarios im Abschnitt 3.1.1 und die Beschreibung der Akteure VO-Initiator bzw. VO-Provider im Abschnitt 3.2.1). Sie orientieren sich daher im wesentlichen an den Managementinteraktionen, die zu Beginn und Ende jeden VO-Lebenszyklus von Bedeutung sind. Eine Beschreibung von VO-Lebenszyklusphasen erfolgte bereits in den Abschnitten 2.1.2 und 3.1.2, so dass an dieser Stelle darauf verzichten werden kann. Lebenszyklen selbst – nicht nur die von VOS – werden im Informationsmodell in der Klasse `VOMAEntityLifecycle` (siehe Abbildung 5.8) beschrieben.

Die Identifikation der VO-Managementrollen einer Community orientiert sich an den Managementinteraktionen direkter und indirekter Art, die die Community-Ebene mit der RO-Ebene und der VO-Ebene im Kontext des Lebenszyklus einer VO verbindet. Dies ist in Abbildung 5.36 als Abstraktion der Abbildung 3.11 dargestellt.

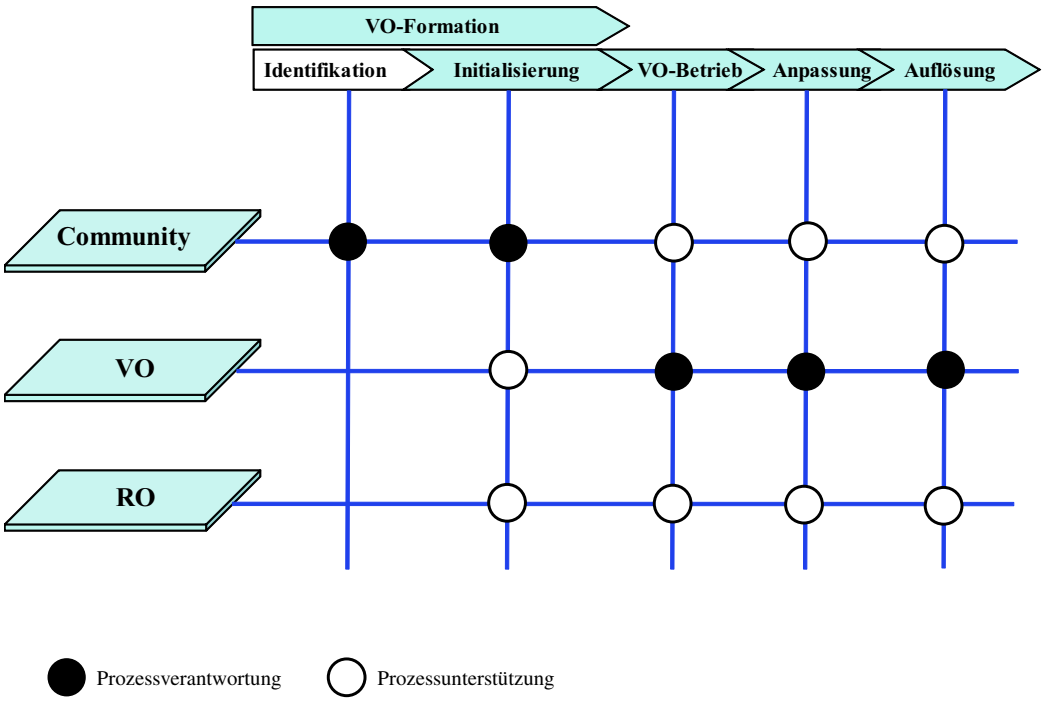


Abbildung 5.36: Prozessverantwortung und Prozessunterstützung der Rollen des VO-Managements

Die Hoheit über das Formationsverfahren einer VO resultiert auf der Community-Ebene deshalb in Interaktionsklassen zur Initiierung und zur Initialisierung von VOs und fällt damit dem klassischen Funktionalbereich des aus dem OSI-Management bekannten Configuration Managements zu, auch wenn Fragen des Abrechnungsmanagements, des Sicherheitsmanagements, des Fehlermanagements und des Leistungsmanagements ebenfalls berührt werden. Der VO-Initiator auf der Community-Ebene ist jedoch einzig und allein daran interessiert, dass die von ihm geforderte VO installiert wird und betriebsbereit ist. Die Betriebsphase der VO wird – siehe auch die Überlegungen im Kapitel 3 – von der VO virtuell „gehostet“. Insofern reduzieren sich VO-Managementinteraktionen dieser Phase auf Seiten der Community auf rein informierende Interaktionen oder direkte Eingriffe in VO-Konfigurationen im Rahmen einer „Änderung der Geschäftsgrundlage“. Die Auflösungsphase involviert die Community wieder mehr, da von ihr in der Regel der Impuls zur Auflösung der VO ausgeht, dieser jedoch wieder als Trigger einer Konfigurationsinteraktion verstanden werden kann.

Damit ergeben sich für die Community-Seite die nachfolgend aufgeführten VO-Managementrollen. Anzumerken ist dabei, dass die hier beschriebenen Rollen zum Teil nur in Ansätzen mit den Aktoren des Abschnitts 3.2.1 übereinstimmen, die dort skizzierten Aktoren jedoch vollständig durch die hier vorgeschlagenen Rollen überdeckt werden. Dies wird in den Tabellen jeweils dargestellt.

Community Configuration Manager. Der Community Configuration Manager (COM-

CM) zeichnet für sämtliche Fragen im Zusammenhang mit der Konfiguration Virtueller Organisationen auf der Community-Ebene verantwortlich (Tabelle 5.10).

Rolle *Community Configuration Manager*	
Bezeichner	COM-CM
Ebene	Community
Beschreibung	verantwortlich für die Konfiguration und Konfigurierung Virtueller Organisationen auf der Community-Ebene
Assoziierte Aktoren der Anforderungsanalyse	VO-Initiator, VO-Provider

Tabelle 5.10: Zusammenfassung der Rolle *Community Configuration Manager*

Dem Community Configuration Manager obliegt damit die „Erstinstallation“ einer VO und deren Bootstrapping. Er ist zudem in die im Rahmen der Auflösung einer VO erforderlichen Maßnahmen eingebunden. Der Community Configuration Manager übernimmt schließlich alle Aktivitäten, die im Zusammenhang mit Community-seitig initiierten Re-Konfigurationen einer VO stehen. Veränderungen im Bereich von Rollen-, Mitgliedschafts- und Ressourcenkonstellationen gehören nicht zum Spektrum des COM-CM, diese werden konzeptionell sauber auf der VO-Ebene selbst wahrgenommen.

Der Community Configuration Manager trägt im übrigen zu einer Art Mandantenfähigkeit bei, indem die Auswirkungen der Initialisierung nur für die jeweils beteiligten realen Organisationen sichtbar sind und für alle anderen Organisationen transparent bleiben, selbst wenn deren Ressourcen durch die VO genutzt werden.

Community VO-Provider. Der Community VO-Provider (COM-VP) ist für den Lebenszyklus einer VO verantwortlich (Tabelle 5.11). Im Rahmen der Formation einer VO beinhaltet dies u.a. die Verhandlungen und Vereinbarungen mit realen Organisationen zur Bereitstellung lokaler Ressourcen, Dienste und Individuen.

Der Community VO-Provider kapselt damit alle mit dem Lebenszyklus einer VO zusammenhängenden „logistischen“ und administrativen Aufgaben. Diese können den folgenden Kategorien zugeordnet werden:

Asset Management. Im VO-bezogenen Asset Management werden auf der Community-Ebene vom COM-VP Charakteristika und *Capabilities* der in der Community versammelten Entitäten gepflegt. Die verfügbaren Ressourcen, Dienste oder Entitäten anderer Art können dann in der Form von Yellow Pages (YP) Brokern oder dem Configuration Management zur Verfügung gestellt werden (wie zum Beispiel im IPY-Szenario).

Rolle *Community VO-Provider*	
Bezeichner	COM-VP
Ebene	Community
Beschreibung	verantwortlich für den Lebenszyklus von VOs
Assoziierte Aktoren der Anforderungs-analyse	VO-Initiator, VO-Provider, VO-Archive Manager

Tabelle 5.11: Zusammenfassung der Rolle *Community VO-Provider*

Wenn die Anzahl der VOs einer Community wächst oder deren Lebenszyklen beschleunigt werden, steigt auf der Community-Ebene die Notwendigkeit einer Community Information Base (CIB), in der die aktiven, aufgelösten und beantragten VOs verwaltet werden (und sei es nur zu Audit-Zwecken). Die Inhalte der Community Information Base können aus der VO-MIB abgeleitet bzw. übernommen werden, Struktur und Operationen sind allerdings nicht Gegenstand dieser Arbeit.

Contract Management. Die Grundlage jeder VO bilden Vereinbarungen zwischen den an der VO beteiligten Institutionen und den „Repräsentanten“ der VO. Diese Vereinbarungen können in Form von komplexen SLAs oder einfachen Policies vorliegen. Die Verhandlung, Festschreibung und Überwachung solcher Vereinbarungen auf der Community-Ebene wird vom Contract Management des Community VO-Providers durchgeführt.

Community Accounting Manager. Der Community Accounting Manager (COM-AM) ist dafür verantwortlich, dass auf der Community-Ebene VOs betreffende abrechnungsspezifische Fragen adressiert werden (Tabelle 5.12).

Rolle *Community Accounting Manager*	
Bezeichner	COM-AM
Ebene	Community
Beschreibung	verantwortlich für VO-orientiertes Abrechnungsmanagement auf der Community-Ebene
Assoziierte Aktoren der Anforderungs-analyse	VO-Provider

Tabelle 5.12: Zusammenfassung der Rolle *Community Accounting Manager*

> Insbesondere stellt der Community Accounting Manager Community-spezifische Verfahren zur Verfügung, die das Erfassen von VO-Beteiligungen und -Nutzungen, das Festlegen entsprechender Abrechnungseinheiten, das Führen von VO-Konten, das Ausstellen von Rechnungen oder das Führen von abrechnungsorientierten Statistiken unterstützen.

Unabhängig von diesen Rollen, können – je nach Granularität der Betrachtung – weitere Rollen identifiziert werden, die sich an den klassischen FCAPS-Funktionsbereichen orientieren. Beispielsweise könnte eine separate Rolle eines Problem-Managers betrachtet werden. Wir haben hier jedoch die Rollen des Configuration Managers und des Accounting Managers hervorgehoben, weil einerseits deren zentrale Bedeutung unbestritten ist und weil andererseits Funktionen des Fehlermanagements und des Sicherheitsmanagements in anderen Rollen und Betrachtungebenen gekapselt werden können (siehe auch Abschnitt 5.4). Funktionen des Leistungsmanagements sind auf der Community-Ebenen unkritisch und werden hier nicht betrachtet. Gleichzeitig soll mit der Diskussion des Accounting Managers auch dargestellt werden, wie weitere Rollen in konkreten Anwendungsfällen hinzugefügt werden können.

Rollen in den beteiligten realen Organisationen

Entitäten Virtueller Organisationen werden stets auf Entitäten realer Organisationen abgebildet. Insofern „delegieren“ reale Organisationen (im weiteren Sinn) Ressourcen, Dienste und Individuen für einen limitierten Zeitraum in VOs und unterstellen sie damit auch deren Management. Die Integration von VOMA in die bestehenden lokalen Managementprozesse realer Organisationen ist – was die Verankerung im Informationsmodelle angeht – in Abbildung 5.17 und in der Klasse `LocalManagementSystem` dargestellt.

Analog zur Vorgehensweise bei der Identifizierung der Rollen in den Communities lassen sich auch die am VO-Management beteiligten Rollen auf der Ebene der realen Organisationen prinzipiell durch eine Spiegelung am VO-Lebenszyklus feststellen. Am VO-Managements sind reale Organisationen in jeder Phase beteiligt (siehe auch Abbildung 5.36). In der Initialisierungs- und Betriebsphase einer VO haben sie die Aufgabe, die notwendigen Kontingente an Ressourcen und Diensten bereitzustellen sowie die RO-spezifischen Teilnehmer an der VO zu authentifizieren. Diese unterstützende Sicht resultiert im Informationsmodell in einer konzeptionellen Trennung von `ManagedVOMAEntities` und `UnManagedVOMAEntities` und in entsprechenden Interaktionsklassen. Die RO ist dabei einzig und allein daran interessiert, dass die von ihr bereitgestellten Ressourcen und Dienste den Vereinbarungen gemäß von der VO genutzt werden und ihnen zu diesem Zweck ständig ein aktuelles Nutzungsprofil vorliegt. Zu berücksichtigen ist ferner, dass die Nutzung von Ressourcen und Diensten kostenpflichtig gestaltet sein kann, wie einige Szenarios (z.B. DEISA) zeigen.

Da reale Organisationen und Individuen hier als `UnManagedVOMAEntities` aufgefasst werden, ist eine detaillierte Analyse der Rollen, die die VOMA-Schnittstelle auf der Seite einer

realen Organisation nutzen, nicht notwendig. Das VOMA-Organisationsmodell kann sich vielmehr darauf beschränken, die Nutzung von VOMA ausschließlich über die Interaktionen der RO-spezifischen lokalen Manager zu modellieren. Die Integration von VOMA-Informationen und -Funktionalitäten in bestehende lokale Managementprozesse ist *of local concern* und nicht im Fokus dieser Arbeit.

Rolle *RO Local Manager*	
Bezeichner	RO-LM
Ebene	Reale Organisation
Beschreibung	verantwortlich für die Aktivitäten der RO-Seite im Zusammenhang mit dem VO-Management
Assoziierte Aktoren der Anforderungsanalyse	Quota-Provider, Local Manager

Tabelle 5.13: Zusammenfassung der Rolle *RO Local Manager*

Der RO Local Manager (RO-LM) (Tabelle 5.13) ist in sämtliche VO-Lebenszyklusphasen involviert, indem er Ressourcen, Dienste und Individuen für die VO bereitstellt und deren Nutzung aus seiner Perspektive überwacht. Dementsprechend definiert er RO-spezifische SLAs und überwacht deren Einhaltung. Nutzungsspezifische Abrechnungen werden dem VO-Management durch den Local Manager für ein entsprechendes Accounting/Billing übermittelt. Der RO-LM erkennt Verletzungen von SLAs und hat Policies als Gegenmaßnahmen installiert. Insofern realisiert das RO Local Management ein Customer Service Management bzgl. einer VO im Sinne von [Langer, 2001; Nerb, 2001].

Rollen in der Virtuellen Organisation

Virtuelle Organisationen werden auf der Community-Ebene gegründet und im Rahmen der Initialisierung mit Entitäten der RO-Ebene bestückt (siehe Abbildung 3.11). Nach der Herstellung der Betriebsbereitschaft im Rahmen des Formationsprozesses übernehmen die VO-Rollen eigenverantwortlich die weiteren Aktivitäten zur Aufrechterhaltung der Betriebsbereitschaft der VO (siehe auch die Beschreibung der Szenarios im Abschnitt 3.1.1). Sie umfassen damit solche Managementinteraktionen, die an den eigentlichen Betrieb einer VO gekoppelt sind und auf das Management von Rollen, Mitgliedern, virtuellen Ressourcen und virtuellen Diensten ausgerichtet sind (siehe Abbildung 5.36).

Die Aufrechterhaltung der Betriebsbereitschaft einer VO resultiert auf der VO-Ebene in Interaktionsklassen zum Management von Mitgliedschaften, zum Management von Rollen, zur Bereitstellung virtueller Ressourcen und zur Bereitstellung virtueller Dienste. Jede dieser Interaktionsklassen kann weiter in folgende Subklassen unterteilt werden:

Fehlermanagement. In diese Subklasse fallen Interaktionen, die der Behandlung von Störungs- und Fehlermeldungen im jeweiligen Kontext dienen (beispielsweise Fehlverhalten von Mitgliedern oder betriebliche Einschränkungen durch Ausfall virtueller Ressourcen).

Konfigurationsmanagement. Diese Subklasse umfasst die Interaktionen, die zur Konfiguration und Re-Konfiguration einer VO erforderlich sind (beispielsweise die Aufnahme neuer Mitglieder oder das Löschen von Rollen).

Abrechnungsmanagement. Diese Subklasse deckt alle Interaktionen im Rahmen von VO-spezifischen Abrechnungs- und Rechnungsstellungsverfahren ab, die beispielsweise VO-Mitglieder betreffen.

Vertragsmonitoring. Auf der VO-Seite ist die Notwendigkeit vom Management von Leistungsaspekten im engeren Sinn während der Betriebsphase eher gering. Da die VO die eigentlichen Ressourcen nicht besitzt, ist für sie nur interessant, ob den Mitgliedern in ihren Rollen die vereinbarten Ressourcen und Dienste zur Verfügung stehen oder nicht, unabhängig von deren Güte. Analog kann für den Aspekt des Sicherheitsmanagements argumentiert werden, dass eine VO ausschließlich an der Einhaltung von Sicherheitsvereinbarungen interessiert ist. Kombiniert definieren beide Aspekte eine Klasse von Interaktionen, die aktuelle und historische Informationen über die Leistung der VO im Kontext vereinbarter Nutzungs- und Teilnahmebedingungen bereitstellt. Dies betrifft beispielsweise die Überwachung von Mitgliedschaftsrelationen, das Monitoring von Rollenverhalten oder die Quantifizierung der Verfügbarkeit virtueller Dienste.

Aus diesen Überlegungen ergeben sich für die VO-Seite die folgenden Rollen im Rahmen des VO-Managements:

VO Member Manager. Der VO Member Manager (VO-MM) ist für sämtliche Fragen im Zusammenhang mit Mitgliedschaften zur VO und deren Rollen zuständig (Tabelle 5.14).

Rolle *VO Member Manager*	
Bezeichner	VO-MM
Ebene	Virtuelle Organisation
Beschreibung	verantwortlich für Mitgliedschaften und Rollen
Assoziierte Aktoren der Anforderungsanalyse	VO-Manager, VO-Administrator

Tabelle 5.14: Zusammenfassung der Rolle *VO Member Manager*

Dem VO Member Manager obliegt damit die Annahme von *Applicants*, die Aufnahme von *Participants* (siehe Abbildung 5.24), die Vergabe von Rechten und die Beendigung von Mitgliedschaften. Er ist zudem verantwortlich für die Abbildung von Mitgliedern auf Rollen.

Virtual Resource/Service Provider. Der Virtual Resource/Service Provider (VO-VRS) ist für sämtliche Fragen im Zusammenhang mit der Bereitstellung und Nutzung virtueller Ressourcen und Dienste einer VO zuständig (Tabelle 5.15).

Rolle *Virtual Resource/Service Provider*	
Bezeichner	VO-VRS
Ebene	Virtuelle Organisation
Beschreibung	verantwortlich für die Bereitstellung und Nutzung virtueller Ressourcen und Dienste
Assoziierte Aktoren der Anforderungsanalyse	VO-Manager, VO-Administrator

Tabelle 5.15: Zusammenfassung der Rolle *Virtual Resource/Service Provider*

Er sorgt dafür, dass für die Lebensdauer einer VO – oder einen anderen vereinbarten Zeitraum – die realen Komponenten der virtuellen Ressourcen und Dienste verfügbar und nutzbar bleiben. Verletzungen der Verfügbarkeit werden an die entsprechenden Institutionen eskaliert und VO-weit bekannt gemacht.

VO Manager. Der VO Manager (VO-MG) ist für den ordnungsgemäßen Betrieb der VO verantwortlich (Tabelle 5.16).

Rolle *VO Manager*	
Bezeichner	VO-MG
Ebene	Virtuelle Organisation
Beschreibung	verantwortlich für den ordnungsgemäßen Betrieb der VO
Assoziierte Aktoren der Anforderungsanalyse	VO-Manager, VO-Administrator

Tabelle 5.16: Zusammenfassung der Rolle *VO Manager*

In dieser Rolle werden analog zum Community VO-Provider alle koordinierenden Aufgaben zum Betrieb einer VO gekapselt, seien sie logistischer, administrativer oder funktionaler Art.

Es sei an dieser Stelle der Vollständigkeit halber angemerkt, dass die hier gewählte Rollenaufteilung nicht die einzige Möglichkeit darstellt, das VO-Management organisatorisch zu unterstützen. Eine Alternative bildet zum Beispiel eine Orientierung an den im Rahmen der ITIL-Prozesse definierten Rollen [Brenner, 2007; Köhler, 2004; Office of Government Commerce, 2001]. Leider fehlt dem ITIL-Kontext eine exakte Beschreibung der Vorgehensweise, wie die spezifizierten Prozesse und die verwendeten Rollen konkret gewonnen wurden. Insofern stellt ITIL (und ähnliche Ansätze) zur Zeit keine geeignete Alternative zu der hier durchgeführten systematischen Ableitung der VO-Managementrollen dar. Stattdessen kann ITIL aber durchaus dazu verwendet werden, im Rahmen eines Tragfähigkeitsnachweises die Relevanz der hier identifizierten Rollen zu bestätigen. Die Abbildbarkeit der VOMA-Rollen auf ITIL-Rollen ist grundsätzlich möglich, würde aber den Rahmen dieser Arbeit sprengen.

5.3.3 Interaktionen der VO-Managementrollen

Nach der Identifizierung der am VO-Management beteiligten Kernrollen auf der Community-Ebene, der RO-Ebene und der VO-Ebene, ist es nun das nächste Ziel dieses Abschnitts, das Zusammenspiel der Rollen im Kontext des VO-Managements zu beschreiben. Die primären Kooperationsbeziehungen zwischen den Rollen werden konzeptionell über Interaktionskanäle in Anlehnung an [Wedig, 2004] modelliert (siehe Abbildung 5.37).

Ein Interaktionskanal beschreibt dabei eine Schnittstelle, an der zwei Rollen in einer definierten Art und Weise miteinander kommunizieren. Interaktionskanäle haben Ähnlichkeiten mit den aus der TINA-Servicearchitektur bekannten Referenzpunkten [Abarca u. a., 1997; Langer, 2001], besitzen aber eine ausgeprägtere Sitzungs- oder Transaktionssemantik, um die zu Grunde liegende Prozessorientierung der Rolleninteraktionen zu unterstreichen. Die an einem spezifischen Interaktionskanal angebotenen Funktionalitäten werden bei der Festlegung des Funktionsmodells im Abschnitt 5.5 diskutiert.

Aus Abbildung 5.37 wird ersichtlich, dass Interaktionskanäle bereichsintern (die Kanäle CC1 und CC2 bzw. VV1 bis VV2) oder bereichsübergreifend ausgerichtet sein können. Eine solche Unterscheidung ist prinzipiell notwendig, um die strengeren Kommunikationspolicies (z.B. Sicherheitsaspekte oder Vertraulichkeitsaspekte) bei bereichsübergreifenden Interaktionen adressieren zu können.

Interaktionskanal CC1. Über den Interaktionskanal CC1 stellt der Community Configuration Manager dem Community VO-Provider den Zugriff auf die geplante und aktuelle Konfiguration der VO in Form von Konfigurationstemplates zur Verfügung. Gleichzeitig erlaubt der Kanal dem Community Configuration Manager, sich einen Überblick über die aktuelle Konfiguration der VO zu verschaffen, da VO-

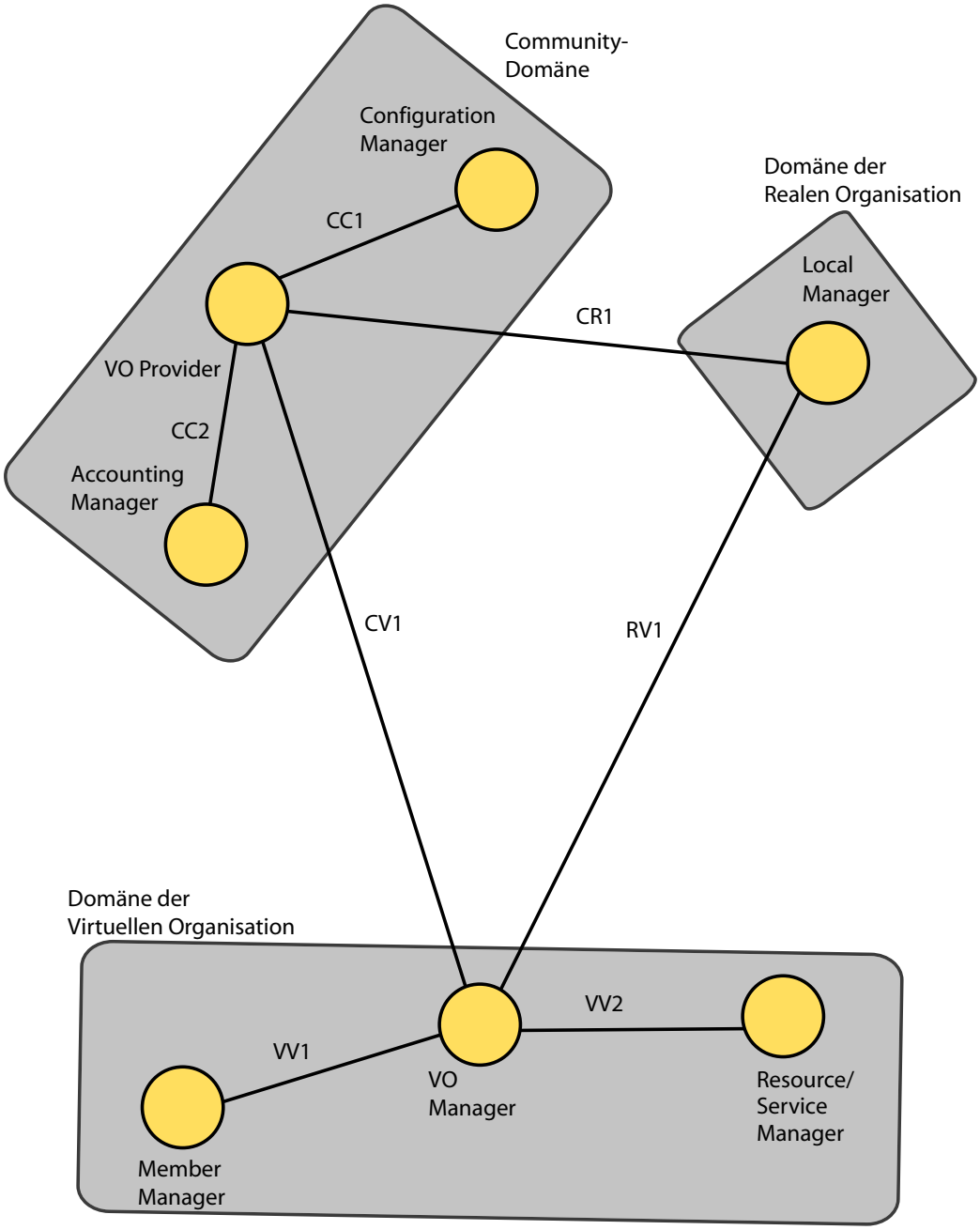


Abbildung 5.37: Interaktionskanäle zwischen VOMA-Rollen

Konfigurationsänderungen dem VO-Provider vom VO-Manager über den Kanal CV1 mitgeteilt werden.

Interaktionskanal CC2. Über den Interaktionskanal CC2 stellt der VO-Provider dem Community Accounting Manager spezifische Abrechnungsinformationen zur VO-Nutzung bereit. Dies beinhaltet neben Informationen zum Verbrauch der VO-

Ressourcen und -Dienste auch Informationen zur Beteiligung und Nutzung einer VO. Der Community Accounting Manager verwaltet umgekehrt Schwellwerte zur VO-Nutzung, deren augenblicklichen Status er dem VO-Provider über diesen Kanal zukommen lässt.

Interaktionskanal CR1. Der Interaktionskanal CR1 dient dem VO-Provider der Community-Ebene zur Kommunikation der zur Gründung einer VO erforderlichen *Capabilities* in Form von Ressourcen, Diensten und Individuen an das lokale Management. Dies kann im Rahmen einer direkten gezielten Anfrage geschehen (*unicast*) oder im Rahmen eines Einladungsprotokolls (*multicast*). Umgekehrt werden über diesen Kanal auch die Angebote des lokalen Managements kommuniziert. Sämtliche Interaktionen zwischen der Community und den realen Organisationen werden über diesen Kanal abgewickelt.

Interaktionskanal CV1. Der Interaktionskanal CV1 dient dem VO-Provider der Community-Ebene zur Kommunikation der für den Betrieb der VO relevanten Randbedingungen in Form übergeordneter Managementinformationen, etwa die Konfiguration der VO betreffend, an den VO-Manager der VO-Ebene. Umgekehrt stellt der VO-Manager dem VO-Provider über diese Schnittstelle die aktuelle VO-Situation dar. Sämtliche Interaktionen zum Management einer VO zwischen der Community und der VO werden über diesen Kanal abgewickelt.

Interaktionskanal RV1. Der VO Manager und das lokale Managementsystem nutzen den Interaktionskanal RV1, um Betriebsphasen-spezifische Managementinformationen auszutauschen.

Interaktionskanäle VV1 und VV2. Der VO-Manager stellt über die Interaktionskanäle VV1 und VV2 dem Member Manager bzw. dem Resource/Service Provider rollenspezifische Sichten auf die VO bereit, während er selbst von beiden Rollen fokussierte Managementinformationen zu Mitgliedern und deren Rollen bzw. virtuellen Ressourcen und Diensten erhält.

Unabhängig von den hier betrachteten Interaktionskanälen lassen sich weitere Kooperationsbeziehungen zwischen den am VO-Management beteiligten Rollen identifizieren. So kann beispielsweise der VO-Manager den Local Manager im Rahmen der Herstellung der Betriebsbereitschaft der VO beauftragen, bestimmte Managementwerkzeuge zu installieren.

5.3.4 Spezifikation des Organisationsmodells

Vor dem Hintergrund dieser Beschreibungen lassen sich nun die Entitäten und Domänen des VOMA-Organisationsmodells formal darstellen. Dazu wird – nicht zuletzt wegen einer angestrebten Konsistenz mit der Spezifikation des Informationsmodells – auch

hier von UML als Spezifikationssprache für die Modellierung des statischen Aufbaus des Organisationsmodells Gebrauch gemacht. Insbesondere eignet sich der UML-Erweiterungsmechanismus des Stereotyps, um die Bedeutung von Domänen, Rollen und die sie verbindenden Interaktionskanälen in der Modellierung zu unterstreichen (siehe auch Kapitel 5.1.1).

Abbildung 5.38 zeigt das VOMA-Organisationsmodell bestehend aus den Domänen `CommunityDomain`, `RealOrganizationDomain` und `VirtualOrganizationDomain`, die jeweils als Stereotyp der Klasse «`VOMADomain`» modelliert sind. Jeder Domäne sind statisch die entsprechenden Rollen gemäß Abbildung 5.37 als Klasse vom Stereotyp «`VOMARole`» zugeordnet. Wenn zwischen Rollen ein Interaktionskanal besteht, wird dieser über eine Klasse des Stereotyps «`InteractionChannel`» modelliert. Der besseren Übersichtlichkeit halber sind in Abbildung 5.38 die Klassen der Interaktionskanäle grau hinterlegt und Multiplizitäten der Aggregationen weggelassen worden.

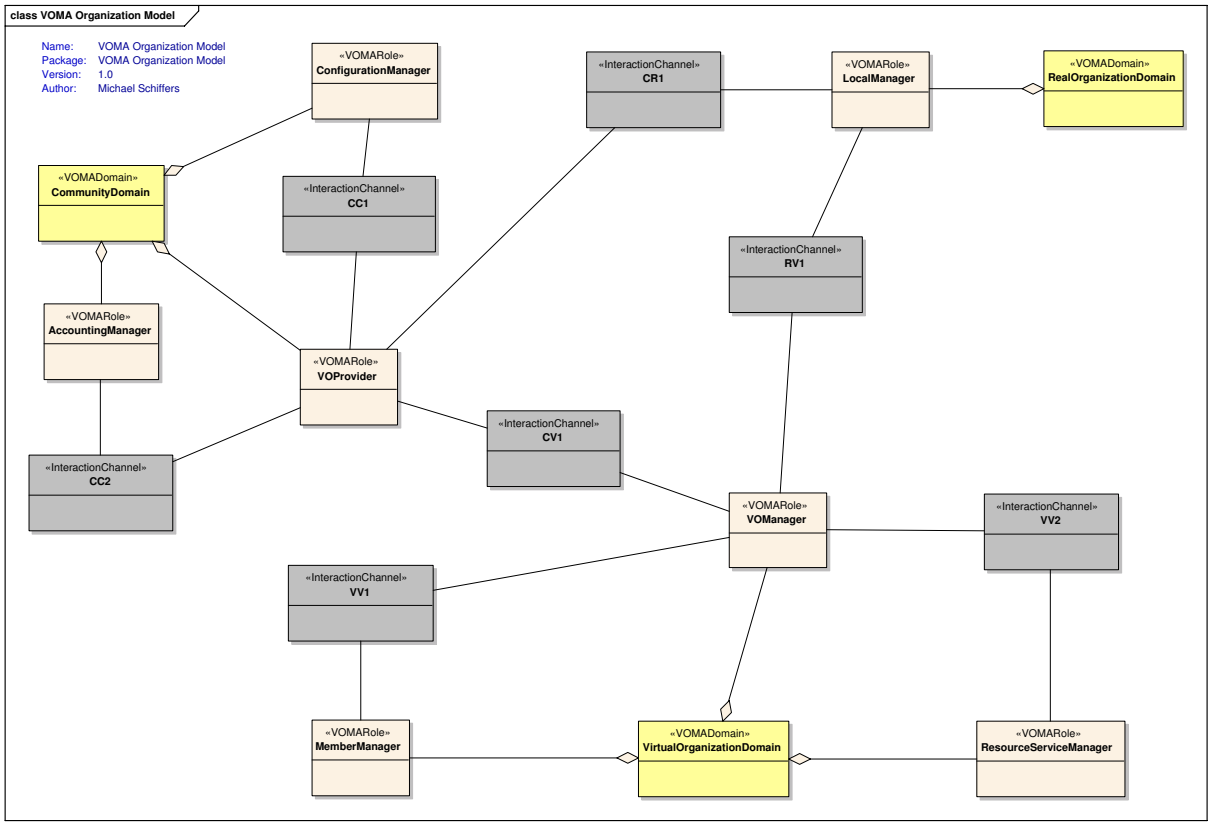


Abbildung 5.38: VOMA Organisationsmodell

Zur Beschreibung der VOMA-Rollen wird der Stereotyp «`VOMARole`» als Erweiterung von UML-Klassen verwendet. Ihm liegt die folgende Semantik zu Grunde: Eine «`VOMARole`» definiert einen Funktionsumfang, der von einer Organisation, einer Person, oder allgemein im VOMA-Kontext einer `VOMAEntity` bereitgestellt bzw. aus-

geführt wird. Jede `VOMAEntity` kann mehrere Rollen unterschiedlicher Art gleichzeitig ausfüllen. Ebenso werden die VOMA-Domänen als Stereotyp «`VOMADomain`» modelliert. Eine «`VOMADomain`» setzt sich aus einer oder mehreren Rollen («`VOMARole`») zusammen, die sich über einen Interaktionskanal austauschen. Auch dieser Kanal wird als Stereotyp modelliert («`InteractionChannel`»). Ein Interaktionskanal bindet zwei Rollen und kann innerhalb einer Domäne oder zwischen zwei Domänen „geschaltet“ sein. Er beschreibt damit eine Schnittstelle zwischen Rollen. Abbildung 5.39 macht dies deutlich.

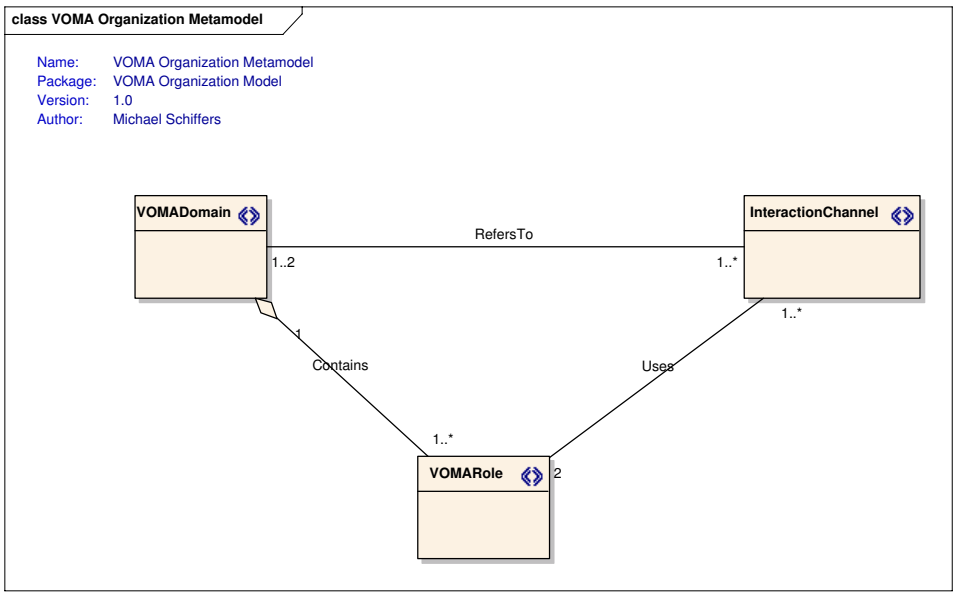


Abbildung 5.39: Metamodell des VOMA Organisationsmodells

Die funktionalen Ausprägungen der Schnittstellen werden in den folgenden Abschnitten erläutert.

5.3.5 Zusammenfassung des Organisationsmodells

In diesem Abschnitt wurde das VOMA-Organisationsmodell spezifiziert. Ausgehend von den Anforderungen des Kapitels 3 wurden mit der `CommunityDomain`, der `RealOrganizationDomain` und der `VirtualOrganizationDomain` die für das VO-Management relevanten Domänen mit ihren Rollen dargestellt. Ebenso wurden zwischen den Rollen Interaktionskanäle identifiziert. Domänen, Rollen und Interaktionskanäle wurden als Stereotype eines VOMA-UML-Profils definiert.

Mit der Spezifikation des Informationsmodells und des Organisationsmodells sind nun die Grundlagen gelegt, um im nächsten Schritt die Kommunikationsmechanismen zwischen den Rollen festzulegen. Dies geschieht nun im Abschnitt 5.4.

5.4 Entwurf des Kommunikationsmodells

Nach der Festlegung des VOMA-Informationsmodells und des VOMA-Organisationsmodells wird nun im Kommunikationsmodell beschrieben, welche Mechanismen zum Austausch von VO-spezifischen Managementinformationen notwendig sind. Dabei sind die folgenden Aspekte von besonderem Interesse (siehe auch [Hegering u. a., 1999]):

- Welche Entitäten kommunizieren Managementinformationen?
- Wie sehen die Kommunikationsmechanismen für Managementinterventionen, für das Monitoring von VOMA-Entitäten und für asynchrone Notifikationen aus?
- Wie sehen die Austauschformate für das Managementprotokoll aus?
- Wie werden die VO-Managementprotokolle in die Dienstarchitektur bzw. Protokollhierarchie der zu Grunde liegenden Kommunikationsinfrastruktur eingebettet?
- Welche auf den Kommunikationsmechanismen aufsetzenden spezifischen Basisdienste werden zusätzlich angeboten?

Die ersten vier Fragen werden im Abschnitt 5.4.1 adressiert, die letzte Frage im Abschnitt 5.4.2.

5.4.1 Kommunikationsmechanismen

Das VOMA-Kommunikationsmodell basiert auf dem klassischen hierarchischen Manager/Agenten-Ansatz [Hegering u. a., 1999]. Dementsprechend können die an einer VO-Managementinteraktion beteiligten Rollen – je nach Ausprägung des Interaktionskanals – sowohl die Rolle eines Agenten als auch die eines Managers einnehmen. Die aktuelle Ausprägung wird für eine Managementanwendung im Informationsmodell in den Klassen `VOMAManagingRole` und `VOMAAgent` hinterlegt, die beide spezialisierte `VOMAEntityRoles` darstellen.

Der Fokus der VOMA-Architektur liegt auf dem Management Virtueller Organisationen in Grids. Obwohl VOMA zunächst Plattform-unabhängig spezifiziert ist, ist die Einbindung in SOA-Konzepte, denen das Prinzip der losen Kopplung zu Grunde liegt [Kaye, 2003], deshalb eine wesentliche Anforderung. Folgerichtig basiert das VOMA-Kommunikationsmodell auf Mechanismen, die a priori kein Persistenzkonzept voraussetzen und asynchron arbeiten können. Vor diesem Hintergrund verwenden VOMA-Manager und -Agenten zu VO-Managementzwecken die folgenden Protokollinteraktionen mit entsprechenden Parametrisierungen:

- `discover`, um Managementobjekte zu identifizieren,
- `get`, um Managementobjekte zu lesen,
- `query`, um Managementobjekte zu suchen,

- `put`, um Attribute von Managementobjekten zu setzen,
- `create`, um Managementobjekte zu generieren,
- `delete`, um Managementobjekte zu löschen,
- `subscribe`, um über Ereignisse informiert zu werden und
- `notify`, um über Ereignisse zu informieren.

Protokollinteraktionen werden von der `VOMAManagingRole` als *Request* formuliert und vom `VOMAAgent` mit einer *Response* bedient. Um Plattform- und (weitgehende) Technologie-Unabhängigkeit zu erreichen, werden Request und Response als XML-Dokumente über einer Grid-Infrastruktur – vorzugsweise einer sicheren Web Services-Infrastruktur – übermittelt. Responses können als Fehlermeldungen oder als Erfolgsmeldungen kategorisiert werden. Request- und Response-Formulierungen sind Plattform-spezifisch und werden hier nicht weiter diskutiert. Ein Beispiel einer XML-Formulierung für eine `get`-Nachricht wird in Listing 5.1 im Kontext des WS-Management-Rahmenwerkes [Arora u. a., 2005] gezeigt, in der Informationen zu einem physischen Plattenspeicher abgefragt werden.

Listing 5.1**:** Beispiel einer `get`-Nachricht nach [Arora u. a., 2005]

```
<s:Envelope
        xmlns:s="http://www.w3.org/2003/05/soap-envelope"
        xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
        xmlns:wsman="http://schemas.xmlsoap.org/ws/2005/06/management">
        <s:Header>

        <wsa:To>
                http://1.2.3.4/wsman/
        </wsa:To>
        <wsman:ResourceURI>
                wsman:samples.org/2005/02/physicalDisk
        </wsman:ResourceURI>
        <wsa:ReplyTo>
                <wsa:Address>
                http://schemas.xmlsoap.org/ws/2004/08/addressing
                /role/anonymous
                </wsa:Address>
        </wsa:ReplyTo>
        <wsa:Action>
                http://schemas.xmlsoap.org/ws/2004/09/transfer/Get
        </wsa:Action>
        <wsa:MessageID>
                uuid:d9726315-bc91-430b-9ed8-ce5ffb858a87
        </wsa:MessageID>
        <wsman:SelectorSet>
                <wsman:Selector Name="LUN"> 2 </wsman:Selector>
        </wsman:SelectorSet>
        <wsman:OperationTimeout> PT30S </wsman:OperationTimeout>

        </s:Header>
        <s:Body/>
</s:Envelope>
```

Im selben Kontext kann dann eine Antwort wie im Listing 5.2 formuliert werden.

Listing 5.2: Beispiel einer `get`-Antwort nach [Arora u. a., 2005]

```
<s:Envelope
        xmlns:s="http://www.w3.org/2003/05/soap-envelope"
        xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
        xmlns:wsman="http://schemas.xmlsoap.org/ws/2005/06/management">

<s:Header>
        <wsa:To>
        http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
        </wsa:To>
        <wsa:Action s:mustUnderstand="true">
        http://schemas.xmlsoap.org/ws/2004/09/transfer/GetResponse
        </wsa:Action>
        <wsa:MessageID s:mustUnderstand="true">
                uuid:d9726315-bc91-430b-9ed8-ce5ffb858a88
        </wsa:MessageID>
        <wsa:RelatesTo>
                uuid:d9726315-bc91-430b-9ed8-ce5ffb858a87
        </wsa:RelatesTo>
</s:Header>

<s:Body>
        <PhysicalDisk xmlns="http://schemas.acme.com/2005/02/samples/physDisk">
                <Manufacturer> Acme, Inc. </Manufacturer>
                <Model> 123-SCSI 42 GB Drive </Model>
                <LUN> 2 </LUN>
                <Cylinders> 16384 </Cylinders>
                <Heads> 80 </Heads>
                <Sectors> 63 </Sectors>
                <OctetsPerSector> 512 </OctetsPerSector>
                <BootPartition> 0 </BootPartition>
        </PhysicalDisk>
</s:Body>

</s:Envelope>
```

Ausgehend von diesen Kern-Operationen werden im Rahmen des VOMA-Kommunikationsmodells weitere Basisdienste definiert, die neben den Kern-Operationen von den Funktionen des Funktionsmodells als „Makros“ verwendet werden. Sie stellen sicher, dass die VOMA-Rollen nur autorisierte Interaktionen durchführen und dass Interaktionen nicht nur transienten Charakter haben müssen. Methodisch werden die (Plattform-unabhängigen) Basisdienste aus der Analyse der Interaktionskanäle des Organisationsmodells aus Abschnitt 5.3 gewonnen (siehe auch Abbildung 5.37). Sie sind generisch angelegt, indem sie von den Spezifika einzelner Interaktionskanäle abstrahieren. Sie verwenden dabei die vorher definierten Kern-Operationen `get`, `put`, etc..

5.4.2 Höhere Basisdienste im Kommunikationsmodell

Das VOMA-Kommunikationsmodell orientiert sich in seinen Grundprinzipien am VOMA-Organisationsmodell und dessen Einbettung in ein Grid-Umfeld, das geprägt ist von Verteiltheit, inter-organisationalem Management, Co-Management und loser Kopplung (siehe auch Abschnitt 3.1). Insofern müssen die Basisdienste konzeptionell auch so angelegt werden, dass Grid-typische Anforderungen, wie etwa die Rollenverteilung über autonome

Organisationen, generisch abgedeckt werden. Eine systematische Beurteilung der Interaktionskanäle des Organisationsmodells liefert vor diesem Hintergrund die Notwendigkeit folgender Basisdienste:

- Publizieren und Finden von Schnittstellen der Interaktionskanäle
- Durchsetzen von Zugangsbeschränkungen zu Interaktionskanälen
- Benachrichtigen über asynchrone Ereignisse im Interaktionskanal
- Protokollieren der Aktivitäten auf dem Interaktionskanal

Sind diese Basisdienste vorhanden, kann auf deren Grundlage unter Berücksichtigung der VOMA-Organisations- und Informationsmodelle das Funktionsmodell festgelegt werden (Abschnitt 5.5).

Registration and Discovery Service

Damit VOMA-Rollen sich überhaupt über einen Interaktionskanal austauschen können, müssen sie diesen etablieren können, was allerdings die Kenntnis der jeweiligen Gegenseite und deren Möglichkeiten (*Capabilities*) voraussetzt. Es wird deshalb ein generischer Basisdienst Registration and Discovery Service (RDS) auf der Grundlage der `discover`-Kern-Operation benötigt, der den VOMA-Rollen die Möglichkeit bietet, die eigenen Capabilities zu publizieren – und damit anderen Entitäten zugänglich zu machen – und andere Rollen durch die Angabe erforderlicher Capabilities zu identifizieren.

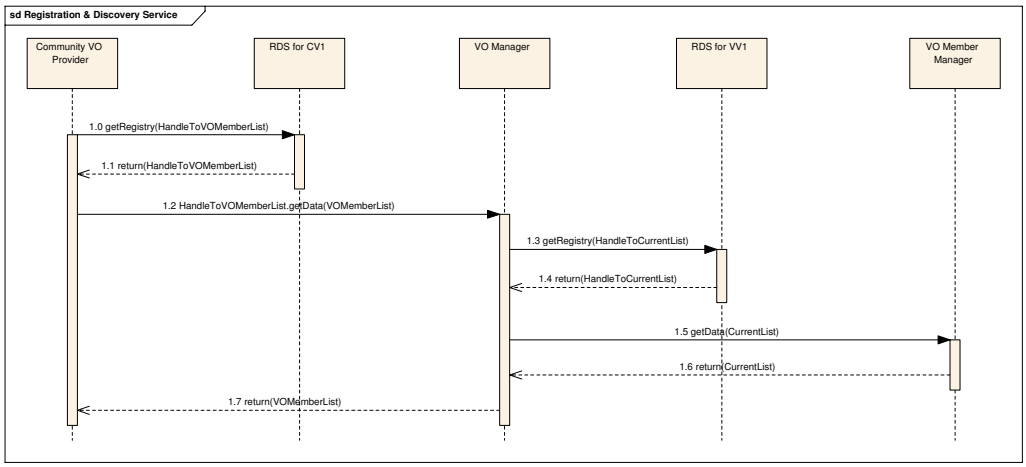


Abbildung 5.40: Beispiel für die Verwendung von RDS

Da das VOMA-Kommunikationsmodell sich wegen seiner lose gekoppelten Entitäten prinzipiell am Publish/Subscribe-Paradigma orientiert, wird RDS für jeden im VOMA-Organisationsmodell identifizierten Interaktionskanal erforderlich. Nur so kann sichergestellt werden, dass die an den Interaktionskanälen beteiligten Rollen die angebotenen

Schnittstellen auch referenzieren können. Zur Verwaltung der Schnittstellen verwendet RDS eine *Registry* mit entsprechenden Einträgen zur Identifizierung von Capabilities. Typische Beispiele für Registry-Realisierungen sind OASIS's Universal Description, Discovery and Integration (UDDI) [OASIS, 2004] und der Index Service des Globus Toolkits [Sotomayor u. Childers, 2006]. Die Einträge in die RDS-Repositories erfolgen durch die Rollen jeweils selbst, was ein global bekanntes und verfügbares, im Informationsmodell in der Klasse `CommonVOScope` repräsentiertes, Repository voraussetzt. Die Aufgabe von RDS besteht ausschließlich darin, Referenzen beim Aufruf der VOMA-Funktionalität aufzulösen.

Abbildung 5.40 zeigt an einem einfachen Beispiel, wie RDS durch den `VO Provider` der `CommunityDomain` genutzt wird, um am Interaktionskanal CV1 über den `VO Manager` der `VirtualOrganizationDomain` eine Referenz auf die aktuelle Memberliste der VO zu erhalten, wobei der `VO Manager` selbst den Kanal VV1 für die Kommunikation mit dem `Member Manager` nutzt.

Security and Restriction Service

Aus der losen Kopplung und der lokalen Autonomie des VOMA-Umfeldes leiten sich einerseits die Anforderung nach einem gesichertem Transport von Managementinformationen ab und andererseits die Erfordernis, den Zugang zu Managementinformationen nur autorisierten Entitäten nach vorheriger Authentifizierung zu gewähren. Für den entarteten Fall, dass eine VO mit einer realen Organisation übereinstimmt oder komplett von ihr abgedeckt wird, kann diese Anforderung allerdings auch abgeschwächt werden.

Der Basisdienst Security and Restriction Service (SRS) widmet sich diesem Aspekt. Auch hier ist wieder anzumerken, dass SRS keine konkrete Realisierung antizipieren will, sondern lediglich Realisierungsanforderungen darstellt. Beispiele möglicher Realisierungen sind in [Hommel, 2007] und für das Grid-Umfeld in [Gietz u. a., 2007] zu finden. Anders als in ähnlichen Ansätzen (z.B. der CSM-Architektur in [Langer, 2001]) wird SRS als Basisdienst jeder Interaktion verwendet. Dadurch wird sichergestellt, dass im dynamischen VO-Umfeld die beteiligten Rollen (genauer: die Rolleninhaber) – nachdem sie ihre Identität gegenüber VOMA im Rahmen von Authentifizierungsverfahren nachgewiesen haben – nur für den Zugang zu für sie vorgesehene Informationen autorisiert sind. SRS muss Plattform-spezifisch auf der Basis der dort zur Verfügung stehenden Sicherheitsprotokolle (z.B. HyperText Transfer Protocol Secure (HTTPS)) realisiert werden. Abbildung 5.41 zeigt dies exemplarisch für den Interaktionskanal RV1 zwischen dem `Local Manager` und dem `VO Manager`, der einen Zugang zu spezifischen SLA-Informationen anfordert.

Bevor eine VOMA-Rolle einen Interaktionskanal nutzen kann, muss sie sich gegenüber der Gegenseite authentifizieren. Dies geschieht über die Operation `authenticate()`. Insofern sendet in Abbildung 5.41 der `VO Manager` eine entsprechende Authentifizierungsanfrage mit seiner eigenen User ID an den `Local Manager`. Der `Local Manager` leitet die Anfrage mit der `authentify()`-Operation an den SRS weiter, der nach entsprechenden Verfahren zunächst überprüft, ob die angegebene `UserID` gültig ist, ob die angegebene Rolle gültig

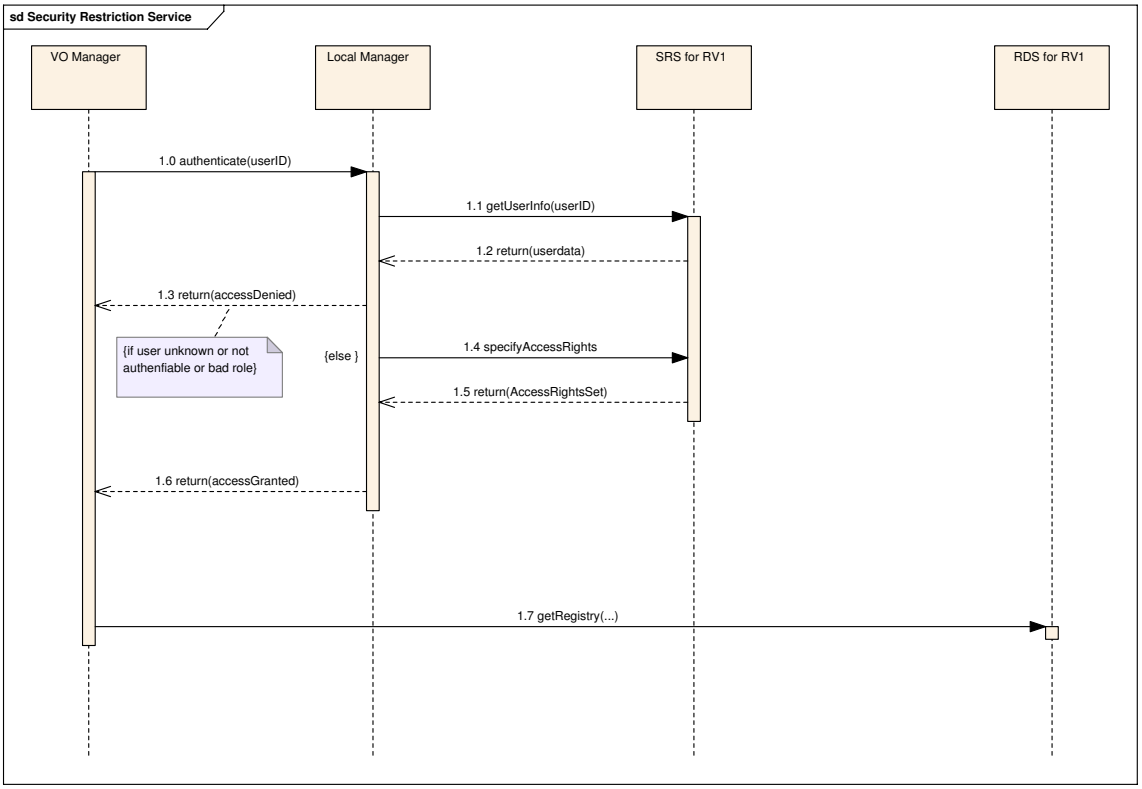


Abbildung 5.41: Beispiel für die Verwendung von SRS

ist und ob die Rollenzuordnung gültig ist. Schlägt eine dieser Prüfungen fehl, wird ein `authentificationFailed`-Fehler ausgelöst. Andernfalls bestimmt der `Local Manager` die Zugriffsrechte des `VO Managers` für diesen Interaktionskanal. Ob für den weiteren Verlauf der Interaktion zwischen `VO Manager` und `Local Manager` eine eigene Sitzung eingerichtet werden muss oder nicht, ist an dieser Stelle von nachrangiger Bedeutung. In jedem Fall basiert der anschließende Zugang zu den Informationen des `Local Managers` auf den aus der Authentifizierung und der Rollenbelegung abgeleiteten oder explizit gesetzten Berechtigungen, die die Autorisierungsentscheidungen steuern. Werden Sessions verwendet, können sie wie in Abbildung 5.42 modelliert werden.

Authentifizierungs- und Autorisierungsmechanismen sind für Grids und allgemeine Föderationen nicht trivial. Auf eine weiterführende Erörterung wird an dieser verzichtet und stattdessen auf die einschlägige Literatur verwiesen, zum Beispiel [Hommel, 2007].

Notification Service

Asynchrone Ereignisse (z.B. die Bereitstellung oder der Ausfall von Ressourcen oder die Aufnahme neuer Mitglieder) machen es notwendig, dass die am VO-Management beteiligten Rollen zeitnah benachrichtigt werden, primär bei domänenübergreifenden Interakti-

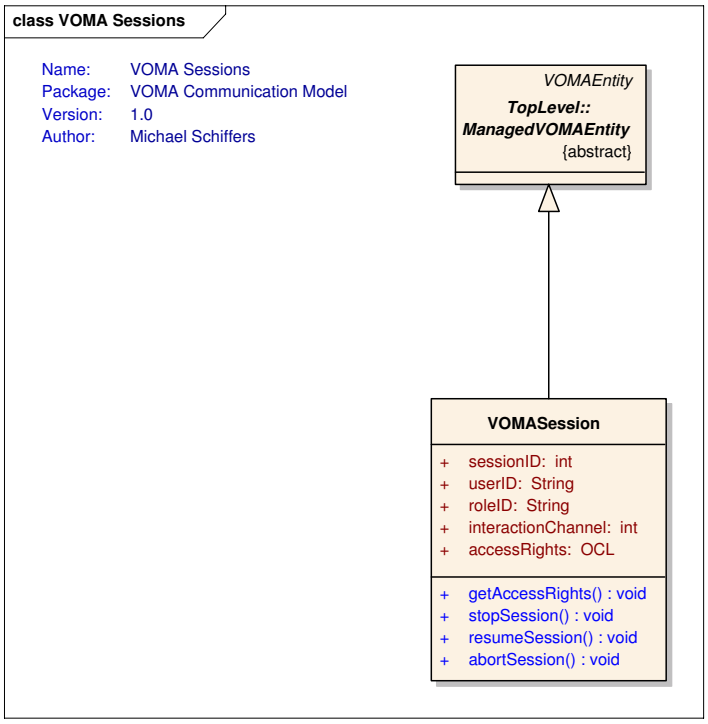


Abbildung 5.42: Modellierung von Sessions im VOMA-Informationsmodell

onskanälen. Dies geschieht mit dem Notification Service (NOS), der einen synchronen und asynchronen Benachrichtigungsmechanismus darstellt.

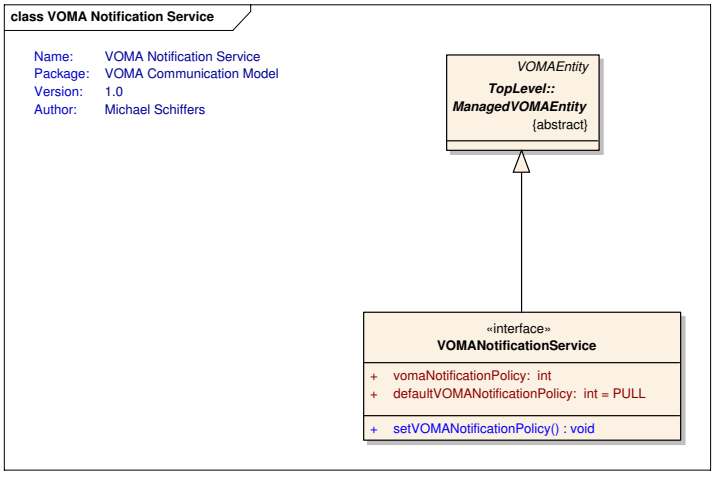


Abbildung 5.43: Einbindung des VOMA Notification Service in das VOMA-Informationsmodell

Wie die eigentliche Benachrichtigung über NOS zugestellt wird (über email, akustische Signale, Popup-Fenster), ist in diesem Kapitel irrelevant und ist Gegenstand einer konkreten Instanziierung der Architektur. Sie wird in den Attributen `vomaNotificationPolicy` und `defaultVOMANotificationPolicy` der Klasse `VOMANotificationService` des Informationsmodells hinterlegt (siehe Abbildung 5.43).

Erlaubte Werte der `VOMANotificationPolicy` sind `PULL` für synchrone Notifikationen und `PUSH` für asynchrone Benachrichtigungen. Zur Realisierung des Dienstes werden die Kern-Operationen `notify` bzw. `subscribe` verwendet.

5.4.3 Zusammenfassung des Kommunikationsmodells

In diesem Abschnitt wurden die Grundelemente des VOMA-Kommunikationsmodells vorgestellt. Ausgehend von den Spezifikationen des Organisations- und des Informationsmodells wurden die für die Interaktionskanäle erforderlichen Kern-Operationen identifiziert und darauf aufbauend zusätzliche Basisdienste in ihren wesentlichen Ausprägungen vorgeschlagen. Weitere Basisdienste erscheinen an dieser Stelle für das VOMA-Kommunikationsmodell nicht notwendig, spezielle Implementierungen der VOMA-Schnittstellen können jedoch weitergehende Dienste erforderlichen machen, die dann allerdings „einsatzspezifisch“ sind.

Nach der Spezifikation des Kommunikationsmodells können nun unter Verwendung der VOMA-Informations- und Organisationsmodelle im nächsten Schritt zum Abschluss der Modellierung der VO-Managementarchitektur die funktionalen Bereiche analysiert und festgelegt werden. Dies geschieht im Abschnitt 5.5.

5.5 Entwurf des Funktionsmodells

Nach der Spezifikation des VOMA-Informationsmodells, des Organisationsmodells und des Kommunikationsmodells wird in diesem Abschnitt nun der Gesamtkomplex „VO-Management“ in adäquate Funktionsbereiche gegliedert. Das Ziel ist dabei, einen Satz generischer Funktionsbausteine für Anwendungen des Managements Virtueller Organisationen in Grids festzulegen [Hegering u. a., 1999]. Es ist nicht Aufgabe des VOMA-Funktionsmodells, einen *vollständigen* Satz von VO-Managementanwendungen zu identifizieren, sondern vielmehr deren Komposition aus allgemeinen Funktionsbausteinen zu unterstützen. Im VOMA-Funktionsmodell werden daher für die einzelnen Funktionsbereiche die erwarteten Funktionalitäten und die VO-Managementobjekte zur Erbringung dieser Funktionalität zu definieren sein. Als Teilmodell der VO-Managementarchitektur ist das VOMA-Funktionsmodell im Konzert aller Modelle der Gesamtarchitektur zu sehen und adressiert die folgenden Fragen:

- In welche Funktionsbereiche kann der Gesamtkomplex des Managements Virtueller Organisationen gegliedert werden?

- Welche generischen VO-Managementfunktionen können dann für die einzelnen Funktionsbereiche festgelegt werden?
- Welche Aufrufkonventionen gelten für die so identifizierten Funktionen?

Diesen Fragen widmet sich der vorliegende Abschnitt. Zur Beantwortung der ersten Frage werden zunächst im Unterabschnitt 5.5.1 die VOMA-Funktionsbereiche festgelegt. Es schließt sich dann für jeden dieser Funktionsbereiche im Unterabschnitt 5.5.2 eine Diskussion der erforderlichen VO-Managementfunktionen an, womit die beiden anderen Fragen adressiert werden.

Diese Diskussion wird für alle Bereiche nach dem gleichen Schema durchgeführt, indem zunächst der Funktionsbereich mit seinem generellen Fokus adressiert wird und anschließend im Kontext des VOMA-Informationsmodells die erforderlichen Objekte und Methoden eingeführt bzw. – wenn nötig – erweitert werden, die dann die Schnittstellen zu den Managementfunktionen bilden.

5.5.1 Festlegung der Funktionsbereiche

Die Strukturierung des Gesamtkomplexes „VO-Management“ kann prinzipiell nach einer Vielzahl von Kriterien erfolgen. Die klassische Technik zur Festlegung von Funktionsbereichen besteht in der Orientierung an den FCAPS-Bereichen des OSI-Managements [Hegering u. a., 1999], eine andere Orientierung bietet das CORBA-Schichtenmodell [Keller, 1998; Siegel, 1996], eine wieder andere Möglichkeit zeigt das Web Services Distributed Management (WSDM)-Framework [OASIS, 2006g, h] mit dem Fokus auf einen Basissatz generischer `get-`, `set-`, `discover-` und `notify`-Funktionen. Im Kontext dieser Arbeit wird eine Strukturierung der VO-Managementfunktionalität auf der Basis der in Abbildung 3.11 (Seite 96) dargestellten Ebenen und der auf diesen Ebenen identifizierten Rollen (siehe Abbildung 5.37) favorisiert. Die konsequent aus den Anforderungen bestimmten Rollen und deren Interaktionskanäle induzieren nämlich bereits kanonisch die VOMA-Funktionsbereiche. Folglich kann zu jeder VOMA-Rolle ein korrespondierender Funktionsbereich mit einem spezifischen Fokus identifiziert werden (siehe auch Abbildung 5.37):

- auf der Community-Ebene (`CommunityDomain`) die Bereiche `Configuration Management`, `VO-Provisioning` und `Accounting Management`
- auf der Ebene der realen Organisationen (`RealOrganizationDomain`) der Bereich `Local Management`
- auf der Ebene der Virtuellen Organisationen (`VirtualOrganizationDomain`) die Bereiche `Member Management`, `VO-Management` und `Resource/Service Management`

Es wird an dieser Stelle jedoch angemerkt, dass – dem Fokus dieser Arbeit entsprechend – Funktionsbereiche, die primär „anwendungsspezifisch“ sind, hier nur *of minor*

concern sind. Dies betrifft beispielsweise den Funktionsbereich eines VO-Initiators (siehe Abschnitt 3.2.1) oder allgemein eines VOMA-Users, der die hier definierten VOMA-Funktionsbausteine im Rahmen komplexerer VO-Managementanwendungen nutzt. Insofern kann die Vollständigkeit des Funktionsmodells nur indirekt nachgewiesen werden, indem die in den Anforderungen dargestellten Anwendungsfälle mit den hier definierten Bausteinen realisierbar sind. Um jedoch auch nicht antizipierte VO-Managementanwendungen behandeln zu können, lässt sich das im Folgenden entwickelte Modell leicht erweitern und an konkrete Gegebenheiten anpassen. Abbildung 5.44 stellt die Funktionsbereiche noch einmal als Paketdiagramm übersichtlich zusammen.

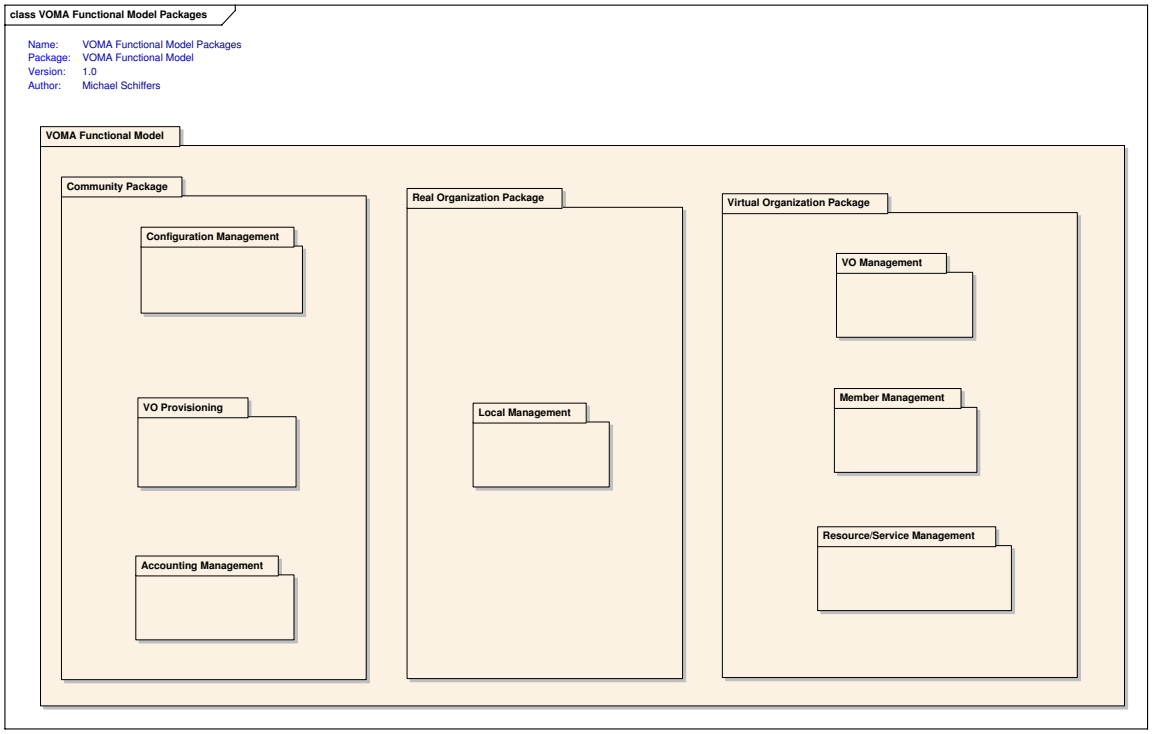


Abbildung 5.44: Überblick über das VOMA-Funktionsmodell

5.5.2 Festlegung der Funktionen

In den folgenden Abschnitten werden die in Abbildung 5.44 dargestellten Funktionsbereiche detailliert betrachtet. Jeder Funktionsbereich wird zunächst allgemein eingeordnet, bevor die spezifischen Objektmodelle und die darauf aufsetzenden Operationen festgelegt werden. Gemeinsam bilden sie die in plattformspezifischen VOMA-Modellen zu realisierenden Schnittstellen (siehe auch Tabelle 5.17 auf Seite 268).

5.5.3 Funktionsbereich *Configuration Management*

Der Funktionsbereich *Configuration Management* ist in der `CommunityDomain` angesiedelt und fokussiert primär auf VO-Konfigurationen als Repräsentationen von historischen, aktiven und geplanten Virtuellen Organisationen. Der Gegenstand der Betrachtung des Configuration Managements sind damit VO-Konfigurationen und Konfigurierungsaufträge.

Allgemeine Einordnung

Die Konfigurierung[6] einer VO umfasst sämtliche Tätigkeiten, die notwendig sind, diese gemäß der in einer Konfigurierungsvorschrift festgelegten Ausprägung bereitstellen zu können. Eine Konfigurierungsvorschrift beschreibt dabei Sollwerte unterschiedlicher Kategorien, wie sie typischerweise VOs zweckspezifisch attributieren. Attribute aktiver VO-Konfigurationen werden im Laufe des Lebenszyklus der sie repräsentierenden VO im VOMA-Informationsmodell reflektiert. VO-Konfigurationen und deren assoziierte Objekte bilden die initiale „Geschäftsgrundlage“ einer VO und dienen als zentrales Bindeglied aller VO-Managementprozesse. Konzeptionell wird daher das Configuration Management über Aktivitätenanforderungen gesteuert, die im konkreten Fall in Workflows oder allgemeineren Prozessen verankert sein können (siehe auch [Danciu, 2007]). Typische Beispiele sind Anforderungen zur Installation einer Konfiguration als VO oder die Archivierung von „abgearbeiteten“ Konfigurationen. Das Configuration Management wird deshalb auftragsorientiert konzipiert. Die stillschweigend getroffene Annahme ist dabei, dass das Configuration Management die primäre Benutzer-Schnittstelle implementiert, zumindest für einen VO-Initiator gemäß Abschnitt 3.2.1.

Mit dem Fokus auf VO-Konfigurationen können damit für das VO-Configuration Management die folgenden Interaktionen identifiziert werden:

Abfrage von VO-Konfigurationen. Die Schnittstelle zum Configuration Management muss die Möglichkeit bieten, historische, aktuelle und – soweit verfügbar – geplante VO-Konfigurationen abzurufen. Insbesondere zu Auditierungszwecken sind Soll/Ist-Vergleiche zwischen ursprünglich geplanten und aktuell vorhandenen Konfigurationen von Bedeutung. Diese Interaktion wird in erster Linie der VO-Provider nutzen, um den Lebenszyklus einer VO einzurichten und zu überwachen.

Festlegen neuer VO-Konfigurationen. Neben der reinen Abfragemöglichkeit können über die Schnittstelle zum Configuration Management VO-Konfigurationen festgelegt werden, die dann zur weiteren Umsetzung im Sinne einer Initialisierung an den VO-Provider weitergereicht werden.

Löschen von VO-Konfigurationen. Komplementär zur Festlegung neuer VO-Konfigurationen, können diese auch gelöscht oder zurückgezogen (*withdraw*) werden.

[6] Wir verwenden hier den Begriff „Konfigurierung“, um den *Prozess* der Bereitstellung einer *Konfiguration* von der eigentlichen Konfiguration als Zustand abzugrenzen.

Funktional sind beide Aspekte aus der Sicht des Configuration Managements identisch, auf der Ebene von Managementanwendungen allerdings nicht, da gelöschte Konfigurationen eine Historie besitzen. Das Löschen einer Konfiguration initiiert gleichzeitig das „Stornieren“ der entsprechenden VO beim VO-Provider.

Ändern von VO-Konfigurationen. Den am VO-Management beteiligten und berechtigten Rollen wird über diese Interaktion die Möglichkeit gegeben, VO-Konfigurationen zu ändern. Historische Konfigurationen können nicht mehr verändert werden, sie können nur noch nachträglich annotiert werden.

Vergleich von VO-Konfigurationen. Nicht nur zu Auditierungszwecken (Soll/Ist-Vergleiche), auch für die Ermittlung von VO-Duplikaten, von Super- und Sub-VOs und von VO-Überlappungen, ist eine Vergleichsmöglichkeit von Konfigurationen erforderlich.

Notifikationen. Das Configuration Management muss über Notifikationen über den aktuellen Stand von Konfigurierungsaufträgen informieren.

Objektmodell zum Configuration Management

Der Gegenstand der Betrachtung des Configuration Managements sind VO-Konfigurationen und Konfigurierungsaufträge. VO-Konfigurationen werden in der abstrakten Klasse `VOConfiguration`, Konfigurierungen über die abstrakte Klasse `VOConfigRequest` modelliert (siehe Abbildung 5.45), die beide von der im Informationsmodell verankerten Klasse `ManagedVOMAEntity` spezialisiert werden. `VOConfigRequest` referenziert `VOConfiguration` über das Attribut `requestConfiguration`.

Die Klasse `VOConfigRequest` beschreibt allgemeine Aspekte von Konfigurierungsaufträgen. Für konkrete Ausprägungen muss sie durch Vererbungsmechanismen entsprechend verfeinert werden, damit spezifische Community- oder Vertragsgegebenheiten berücksichtigt werden können. Ein Auftrag zur Konfigurierung einer VO wird typischerweise durch einen VO-Initiator ausgelöst und ist durch folgende Attribute gekennzeichnet:

`requestType`**.** Dieses Attribut kennzeichnet den Typ des Konfigurierungsauftrags. Dabei kann es sich um einen Installationsrequest, eine Stornierung eines bestehenden Installationsrequests, eine Kündigung einer Konfiguration (entspricht einer Terminierungsanforderung einer VO), eine Änderungsanforderung oder einen Archivierungsauftrag handeln. Entsprechend wird dieses Attribut mit den Werten `VOInstallationRequest`, `VOWithdrawRequest`, `VOCancellationRequest`, `VOChangeRequest`, `VOArchiveRequest` belegt.

`requestState`**.** Der operative Zustand eines Konfigurierungsauftrages wird im Attribut `requestState` beschrieben. Abbildung 5.46 stellt die möglichen Zustände dar.

Wenn ein neuer Konfigurierungsauftrag an das VO-Configuration Management gestellt wird, befindet dieser sich zunächst im Zustand `RequestReceived`, in dem eine

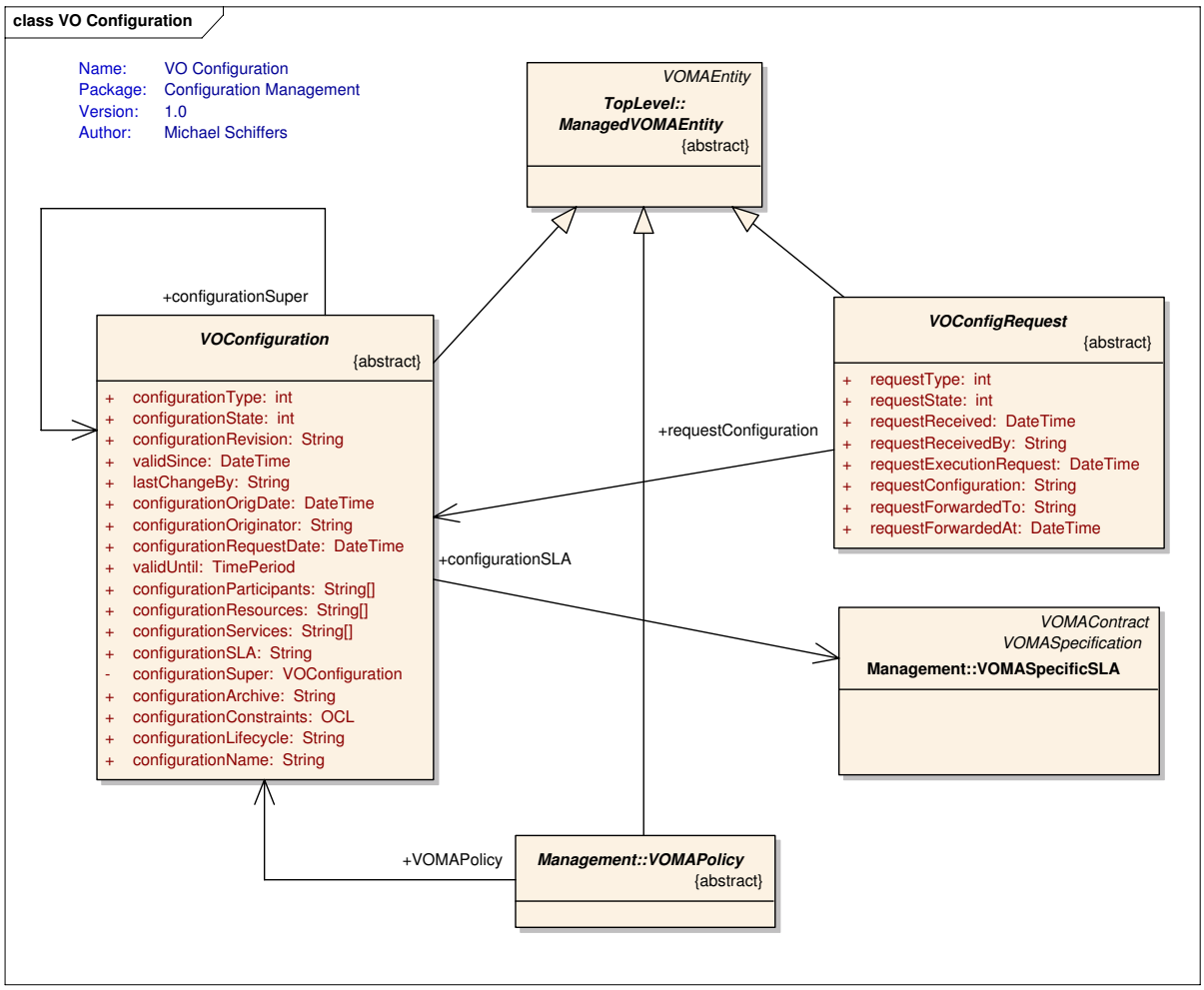


Abbildung 5.45: Konfiguration und Konfigurierung von VOs

Evaluation des Requests durchgeführt wird. Diese umfasst eine formale Prüfung der Berechtigung zur Durchführung des Requests, eine Überprüfung der Konformität des Requests mit aktuellen Policies der Community und eine Analyse historischer Daten. Beim Übergang von `RequestReceived` zu `RequestEvaluated` wird die auftraggebende Rolle über die Notifikation `notifyReceived` über den Eingang des Requests benachrichtigt. Fällt die Evaluation negativ aus, wird der Auftrag mit der Nachricht `notifyRejected` zurückgewiesen. Andernfalls ist der Request angenommen (Zustand `RequestAccepted` und Nachricht `notifyAccepted`). Er kann nun bearbeitet und nach Fertigstellung abgeschlossen werden.

Die Durchführung eines Requests kann allerdings hin und wieder unterbrochen werden. In diesem Fall nimmt das Attribut `requestState` den Wert `RequestSuspended` an. Aus diesem Zustand kann entweder in den Zustand `RequestAccepted` zur Wiederaufnahme der Bearbeitung zurück gewechselt werden oder der Auftrag kann

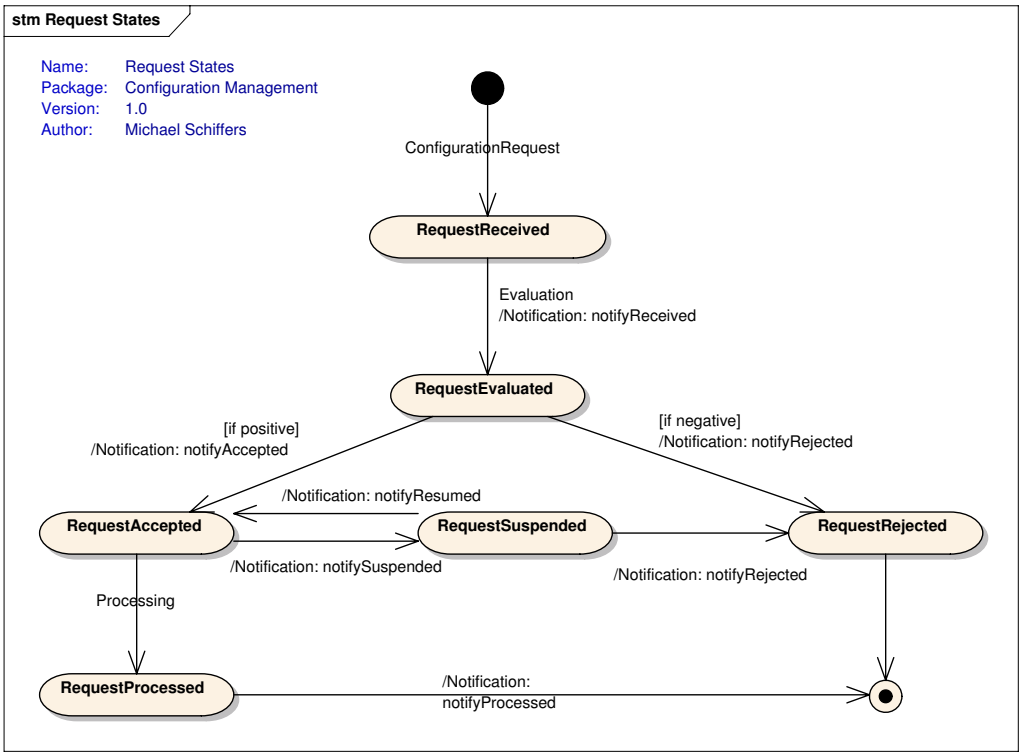


Abbildung 5.46: Zustandsübergänge `requestState`

zurückgewiesen werden. Bei den Zustandsübergängen werden entsprechende Notifikationen gemeldet. Typische Beispiele von Aussetzungen treten bei der Einrichtung von Super-VOs oder der Berücksichtigung neuer Community-Konstellationen auf.

`requestReceived`. Dieses Attribut dokumentiert den Zeitpunkt, an dem das Configuration Management die Konfigurierungsanforderung erhalten hat.

`requestReceivedBy`. In diesem Attribut ist der Auftraggeber der Konfigurierungsanforderung hinterlegt.

`requestExecutionRequest`. Mit diesem Attribut gibt der Auftraggeber einen Wunschtermin an, zu dem die VO-Konfigurierung gestartet werden soll.

`requestConfiguration`. In diesem Attribut wird auf die mit diesem Konfigurierungsrequest assoziierte VO-Konfiguration verwiesen.

`requestForwardedTo` und `requestForwardedAt`. Das Configuration Management managt VO-Konfigurationen, nicht jedoch deren Lebenszyklus. Insofern wird das eigentliche Bearbeiten des eingegangenen Requests delegiert. Wann und an wen delegiert wurde, wird in den Attributen `requestForwardedTo` bzw. `requestForwardedAt` hinterlegt.

Mit dem Attribut `requestConfiguration` verweist ein Konfigurierungsrequest auf eine bestimmt VO-Konfiguration. VO-Konfigurationen werden durch folgende Attribute gekennzeichnet:

`configurationType.` Dieses Attribut repräsentiert den Typ einer Konfiguration. Erlaubte Werte sind `private` und `public`.

`configurationState.` In diesem Attribut ist der aktuelle Status der Konfiguration hinterlegt. Gültige Werte sind `active`, `planned`, `withdrawn` und `finished`.

`configurationRevision.` Konfigurationen können sich im Laufe ihrer Lebensdauer ändern. In diesem Attribut ist deshalb eine Revisionsnummer hinterlegt.

`validSince.` Dieses Attribut spezifiziert, ab wann die in `configurationRevision` hinterlegte Version gültig ist.

`lastChangeBy.` Hier wird der Urheber der in `configurationRevision` hinterlegten Version eingetragen.

`configurationOrigDate` **und** `configurationOriginator.` In diesen Attributen werden der Zeitpunkt der Anforderung der ursprünglichen Konfiguration und deren Autor hinterlegt.

`configurationRequestDate.` Der Zeitpunkt einer VO-Konfiguration muss nicht notwendigerweise mit dem Zeitpunkt des Konfigurierungsrequests übereinstimmen. In diesem Attribut wird deshalb der geplante Konfigurationszeitpunkt hinterlegt.

`validUntil.` Dieses Attribut enthält den projektierten Gültigkeitszeitraum der Konfiguration.

`configurationParticipants.` In diesem Attribut werden die Teilnehmer (als Individuen) an der VO repräsentiert.

`configurationResources.` In diesem Attribut werden die virtuellen Ressourcen der VO hinterlegt.

`configurationServices.` In diesem Attribut werden die virtuellen Dienste der VO hinterlegt.

`configurationSLA.` Dieses Attribut verweist auf eine mit dieser Konfiguration assoziierte VO-spezifische Dienstvereinbarung, die im Informationsmodell hinterlegt ist (Klasse `VOMASpecificSLA`).

`configurationSuper.` Hier wird spezifiziert, mit welchen anderen VO-Konfigurationen diese Konfiguration verlinkt ist. Dieses Attribut wird benötigt, um Hierarchien von VOs darstellen zu können (Super-VOs).

`configurationArchive.` Dieses Attribut repräsentiert einen Link auf ein Archiv, in dem die VO-Konfiguration hinterlegt ist bzw. wird.

`configurationConstraints`. Dieses Attribut repräsentiert Policies, die direkt einer VO-Konfiguration zuzuordnen sind. Sie werden als OCL-Konstrukte ausgedrückt (siehe auch [Kempter, 2004]) und sind mit der Klasse `VOMAPolicy` des Informationsmodells assoziiert.

`configurationLifecycle`. Dieses Attribut repräsentiert die Verbindung der Konfiguration mit einem VO-Lebenszyklus, der im Informationmodell in der Klasse `VOMAEntityLifecycle` (Abbildung 5.8) modelliert wird.

`configurationName`. Dieses Attribut enthält den Namen der Konfiguration, der typischerweise auch der Name der repräsentierten VO ist, wie er auch in der Klasse `VirtualOrganization` als geerbtes Attribut zu finden ist.

Funktionen des Configuration Managements

Basierend auf diesen Objekten bietet der Funktionsbereich des Configuration Managements an seinen Schnittstellen die folgenden Funktionen an:

Abfrage von VO-Konfigurationen. Zur Abfrage von VO-Konfigurationen wird vom Configuration Management die Methode `queryVOConfiguration(searchpattern)` angeboten. Mit Hilfe dieser Funktion können die mit dem Configuration Management interagierenden Rollen – je nach Autorisierung – Teile einer VO-Konfiguration (z.B. die Abfrage des Attributs `configurationRevision`) abfragen, eine komplette VO-Konfiguration lesen oder alle VO-Konfigurationen suchen, die den in `searchpattern` spezifizierten Kriterien genügen. Die Präsentation der Suchergebnisse erfolgt als Sequenz von Attribut-Wert-Paaren, die Plattform-spezifisch aufbereitet werden können. Abbildung 5.47 spezifiziert diese Funktion[7].

Festlegen neuer VO-Konfigurationen. Neue VO-Konfigurationen werden durch die Methode `requestVOConfiguration()` bereitgestellt. Dazu müssen die folgenden Attribute der Klasse `VOConfigRequest` belegt werden: `requestType` mit dem Wert `VOInstallationRequest`, `description` (ein von der Klasse `ManagedVOMAEntity` geerbtes Attribut) mit einer Beschreibung der Konfiguration, `requestConfiguration` mit einem Zeiger auf eine Instanz der Klasse `VOConfiguration` (die Soll-Konfiguration) und `requestExecutionRequest` mit dem gewünschten Installationszeitpunkt. Nach Überprüfen der erforderlichen Autorisierung leitet der Configuration Manager den Auftrag zur Installation einer VO gemäß der angegebenen Konfigurationsspezifikation an den VO-Provider weiter, ergänzt um die Attribute `requestReceived`, `requestReceivedBy`, `requestForwardedTo` und `requestForwardedAt`. Abbildung 5.48 zeigt einen Ausschnitt aus dem Installations-Workflow.

[7] In diesen und folgenden Sequenzdiagrammen wird die Rolle eines „Users“ verwendet. Diese ist generisch zu interpretieren und wird benötigt, um ein vollständiges Bild der Interaktionen zu gewinnen.

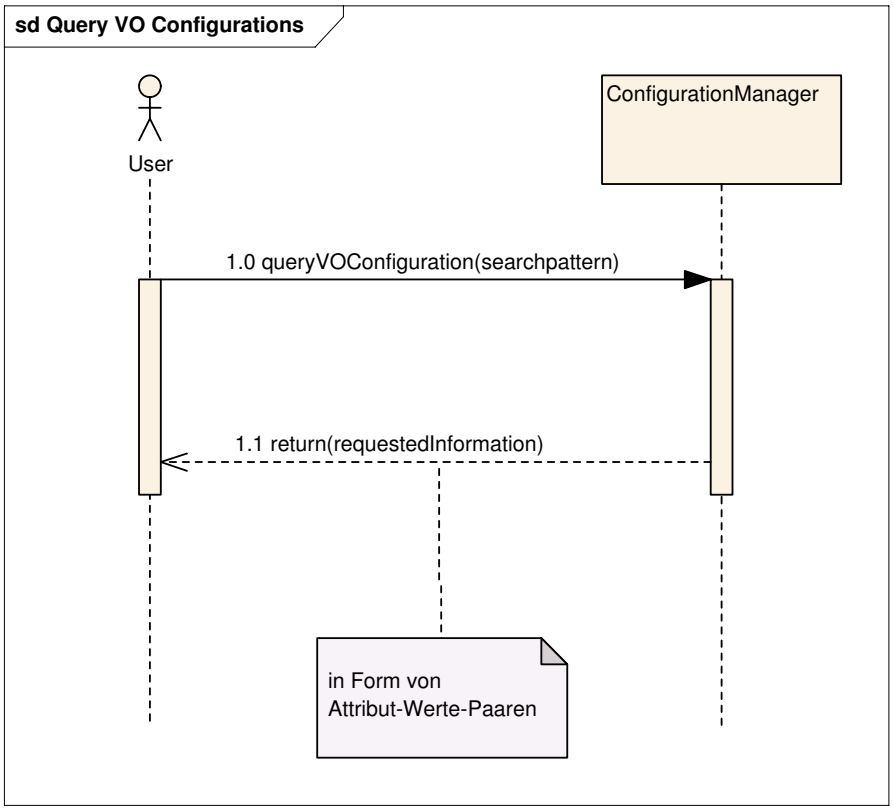


Abbildung 5.47: Sequenzdiagramm zur Abfrage von VO-Konfigurationen

Löschen von VO-Konfigurationen. Zum Löschen einer Konfiguration dient die Funktion `requestVOConfiguration()`, wobei das Attribut `requestType` mit dem Wert `VOCancellationRequest` belegt wird. Der Workflow dieser Funktionsausprägung verläuft prinzipiell analog zu dem in Abbildung 5.48 angegebenen, auf eine gesonderte Darstellung wird deshalb hier verzichtet. Eine Sonderstellung nimmt der Fall ein, eine Konfiguration zu löschen, die noch nicht oder nicht mehr aktiv ist (repräsentiert durch das Attribut `configurationState` der Klasse `VOConfiguration`). In diesem Fall ist eine Benachrichtigung des VO-Providers nicht erforderlich. Eine zusätzliche Fehlerquelle besteht allerdings darin, eine historische Konfiguration (`configurationState == finished`) löschen zu wollen, sollte diese noch nicht archiviert worden sein (Attribute `configurationArchive == NULL` und `configurationState == finished`). Die Rücknahme eines Konfigurierungsrequests wird über den `requestType VOWithdrawRequest` ausgelöst.

Ändern von VO-Konfigurationen. Zum Ändern einer Konfiguration wird die Funktion `requestVOConfiguration()` mit dem `requestType = VOChangeRequest` genutzt. Auch dieser Workflow verläuft prinzipiell analog zu dem in Abbildung 5.48 angegebenen. Eine Sonderstellung nimmt auch hier der Fall ein, eine Konfigura-

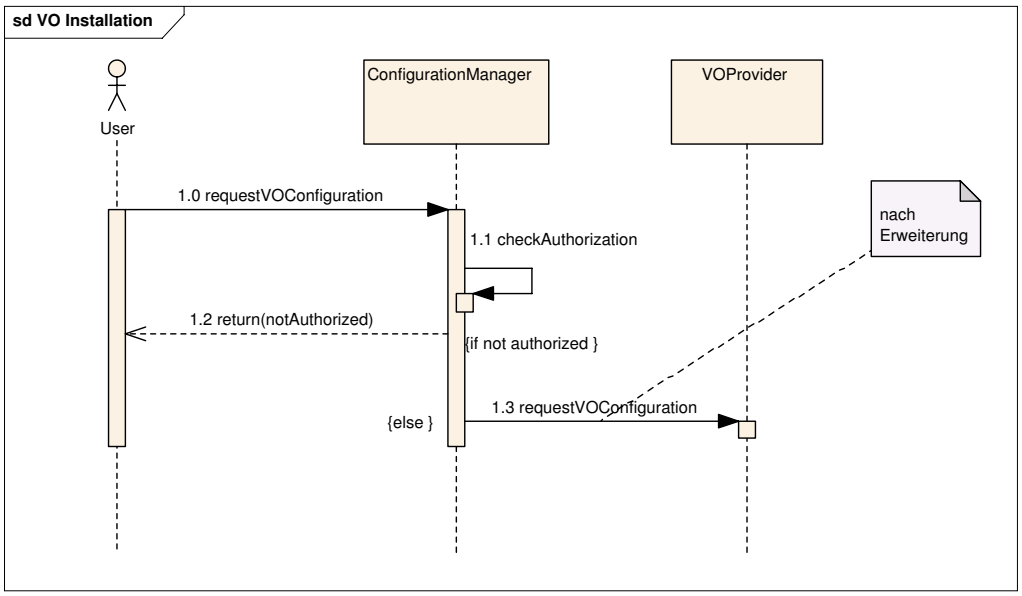


Abbildung 5.48: Sequenzdiagramm zur Installation neuer VO-Konfigurationen

tion ändern zu wollen, die nicht mehr aktiv ist (repräsentiert durch das Attribut `configurationState` der Klasse `VOConfiguration`). In diesem Fall ist eine Benachrichtigung des VO-Providers nicht erforderlich, stattdessen wird eine Fehlermeldung gesendet.

Archivierung von VO-Konfigurationen. Zum Archivieren von VO-Konfigurationen wird die Funktion `requestVOConfiguration()` aufgerufen, wobei der `requestType` den Wert `VOArchiveRequest` besitzt. Das Archiv wird über das Attribut `configurationArchive` adressiert. Der Workflow ist offensichtlich und wird hier übersprungen.

Vergleich von VO-Konfigurationen. VO-Konfigurationen werden mit der Funktion `compareVOConfigurations()` verglichen. Dazu werden der Funktion zwei Instanzen der Klasse `VOConfiguration` als Parameter übergeben sowie eine durch den Parameter `compareMethod` repräsentierte Vergleichsmethode. `compareMethod` kann die folgenden Werte annehmen: `attribute` für den Vergleich von individuellen Attributen (die dann separat angegeben werden), `full` für den Vergleich der kompletten Instanz und `instance` für die Überprüfung, ob beide Instanzen existieren. Der Workflow der Funktion `compareVOConfigurations()` ist wieder offensichtlich, auf eine detaillierte Darstellung wird deshalb hier verzichtet.

Zusammen mit dem vorgestellen Objektmodell (Konfigurationen und Konfigurierungen) definieren diese Funktionen mit ihren Schnittstellen den Funktionsbereich *Configuration Management* des VOMA-Funktionsmodells.

5.5.4 Funktionsbereich *VO-Provisioning*

Der Funktionsbereich *VO-Provisioning* ist in der `CommunityDomain` angesiedelt und fokussiert primär auf VO-Lebenszyklen.

Allgemeine Einordnung

Die Aufgabe des VO-Provisionings liegt im Management von VO-Lebenszyklen. Basierend auf einer vom VO-Configuration Management bereitgestellten Soll-Konfiguration, schaut das VO-Provisioning auf den Lebenszyklus von VOs aus der Perspektive der Community (im Gegensatz zum VO-Manager der VO-Ebene, der sich auf die Betriebsphase im Lebenszyklus konzentriert). Der VO-Provider erwartet daher Funktionen zur Abfrage des Lebenszyklusstatus, zum Anstoß von Zyklustransitionen und zur Initialisierung bzw. Terminierung von VOs. Für das Lebenszyklus-Management im VO-Provisioning können dementsprechend die folgenden Interaktionen identifiziert werden:

Abfrage des Lebenszyklusstatus. Das VO-Provisioning muss eine Schnittstelle zur Verfügung stellen, an der der aktuelle Zustand eines VO-Lebenszyklus abgefragt werden kann. Diese Funktionalität ist nicht zu eng zu sehen, sondern umfasst neben der typischen diskreten Statusabfrage auch die Bereitstellung von Lebenszyklus-Traces über einen definierten Zeitraum und regelmäßige statistische Reports den Lebenszyklus einer oder mehrerer VOs betreffend (z.B. Soll/Ist-Vergleiche). Zusätzlich informiert das VO-Provisioning über aktuelle Abweichungen von Lebenszyklus-Planungen. Diese werden in der Regel an das Configuration Management weitergeleitet, das dem VO-Initiator eine entsprechende User-Schnittstelle bietet.

Transitionen im VO-Lebenszyklus. VOs, repräsentiert durch eine aktive VO-Konfiguration (Attribut `configurationState` der Klasse `VOConfiguration`), befinden sich zu jedem Zeitpunkt ihrer Existenz in einem eindeutigen Zustand. Ein Zustandswechsel (Transition) wird in VOMA konzeptionell ereignisgesteuert ausgelöst, unabhängig davon, ob es sich um ein asynchrones Ereignis handelt oder um eine geplante Transition. Das VO-Provisioning stellt neben Notifikationsmechanismen an seiner Schnittstelle Funktionen zur Durchführung regulärer Transitionen vom Zustand X in den Zustand Y (der Normalfall) und zur Durchführung forcierter Transitionen bereit. Forcierte Transitionen sind notwendig, wenn Prioritätsregelungen und Unterbrechungsmechanismen, wie sie beispielsweise im EmerGrid-Szenario (Abschnitt 3.1.1) auftreten, realisiert werden sollen.

Aussetzen und Wiederaufnahme von Lebenszyklen. Das VO-Provisioning stellt für den Fall aktiver Eingriffe in VO-Lebenszyklen (z.B. zur Durchführung von VO-Rekonfigurationen) Methoden bereit, um diese kurzzeitig über *Suspend*- und *Resume*-Funktionalitäten anhalten und wieder starten zu können.

Konfigurierung von Lebenszyklen. Das VO-Provisioning bietet an seiner Schnittstelle die Möglichkeit, VO-Lebenszyklen zu konfigurieren (*build lifecycle*), umzu-

konfigurieren (*change lifecycle*) und zu verwerfen (*destroy lifecycle*). Zusätzlich besteht die Möglichkeit, Lebenszyklen gegen Unterbrechungen zu schützen, um „Ausführungsgarantien“ realisieren zu können (*guard lifecycle*).

Notifikationen. Vom VO-Provisioning wird erwartet, dass es über eine entsprechende Schnittstelle Notifikationen zu Zustandswechseln einer VO und Fehlersituationen bereitstellt.

Objektmodell zum VO-Provisioning

Der Gegenstand der Betrachtung des VO-Provisionings sind VO-Lebenszyklen und deren Phasen. Diese wurden im Abschnitt 5.2.3 im Kontext des VOMA-Informationsmodells ausführlich behandelt, auf eine erneute Diskussion kann deshalb hier verzichtet werden. Abbildung 5.8 auf Seite 187 zeigt das dazu gehörende Objektmodell, das in Abbildung 5.49 noch einmal aufgeführt wird, jetzt allerdings erweitert um die Assoziation mit einer VO-Konfiguration (Klasse `VOConfiguration`).

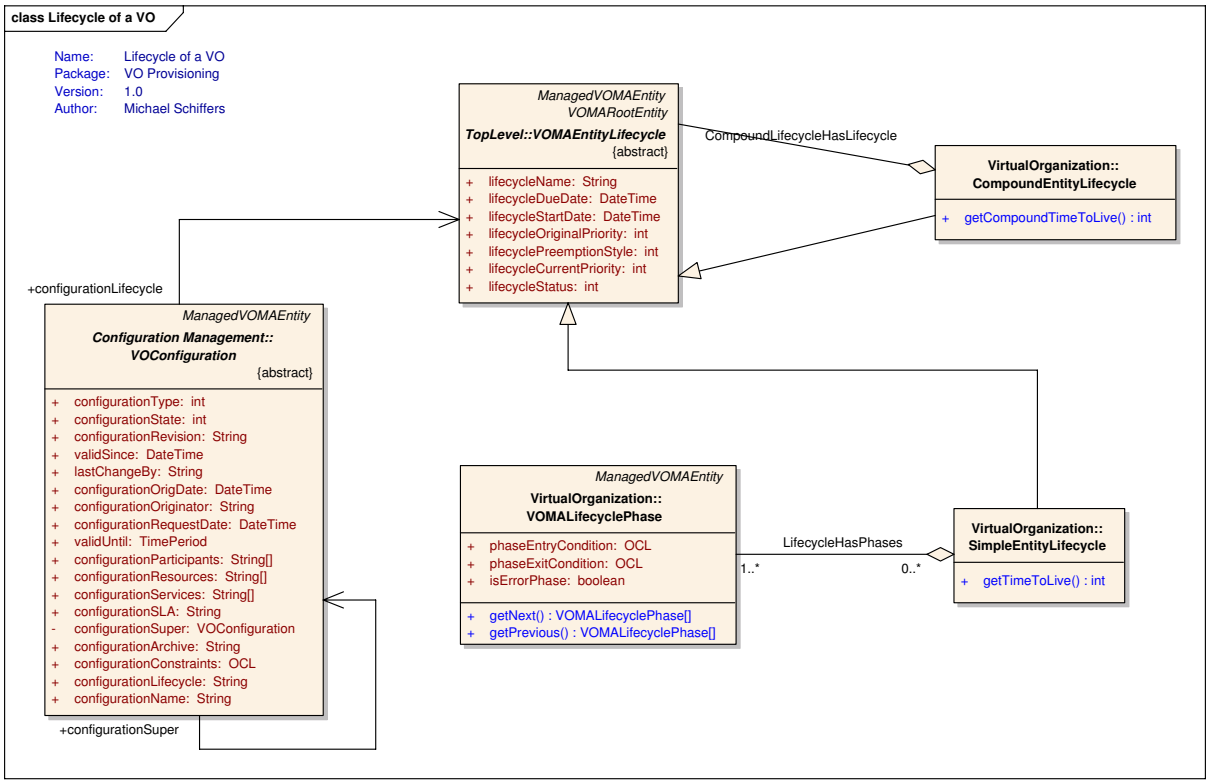


Abbildung 5.49: Konfiguration und Lebenszyklen von VOs

Der Lebenszyklus einer VO aus der Perspektive des VO-Provisioning ist in Abbildung 5.50 dargestellt, wobei die Konfigurierungsphase typischerweise vom Configuration Mana-

gement durchgeführt wird und auch dort besprochen wurde (Abschnitt 5.5.3). Vom VO-Provisioning werden alle weiteren Transitionen durch die entsprechenden Signale `StartVO`, `StopVO`, `ResumeVO`, `CloseVO` und `DestroyVO` ausgelöst. Die jeweils aktuelle Lebenszyklusphase wird im Attribut `lifecycleStatus` der Klasse `VOMAEntityLifecycle` hinterlegt. Die angezeigten Notifikationen benachrichtigen jeweils über Zustandsänderungen bzw. Fehlerkonditionen. Eine VO ist betriebsbereit, sobald sie sich in der Phase `VOStarted` befindet. Mögliche Fehlersituationen, die eine Transition in den Fehlerzustand `VOFailure` anstoßen, können a priori hier nicht festgelegt werden. Sie sind fallspezifisch und müssen im Rahmen des Deployments durch entsprechende Spezialisierungen festgelegt werden. Diese können beispielsweise über die Klasse `AtomicPolicyEventSpec` (siehe Abbildung 5.11 auf Seite 192) definiert werden.

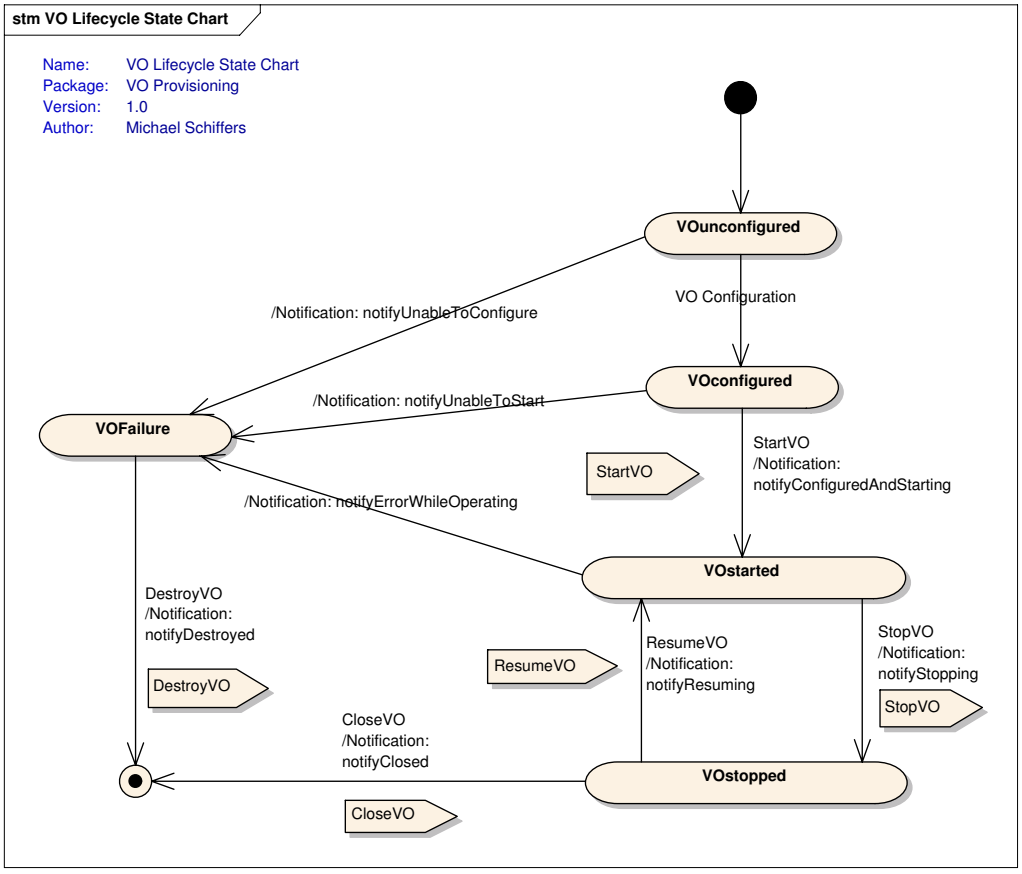


Abbildung 5.50: Zustandsübergänge im VO-Lebenszyklus

Funktionen des VO-Provisioning

Vor diesem Hintergrund bietet der Funktionsbereich des VO-Provisionings an seinen Schnittstellen die folgenden Funktionen an:

Abfrage des Lebenszyklusstatus einer VO. Zur Abfrage des Lebenszyklusstatus einer VO wird vom VO-Provisioning die Methode `getVOLifecycle()` mit dem Identifier einer VO-Konfiguration als Parameter angeboten. Alternativ kann auch der Name einer VO als Parameter mitgegeben werden, dann liefert die Methode den Lebenszyklusstatus der dieser VO zugeordneten Konfiguration mit dem `configurationState == active` und dem höchsten Revisionsstatus (Attribut `configurationRevision`). Mit der Methode `traceLifecycle()` bietet das VO-Provisioning die Möglichkeit, ausgewählte Attribute der Klasse `VOMAEntityLifecycle` zwischen zwei als Parameter angegeben Zeitpunkten zu verfolgen. Die Traces können in Spezialisierungen der Klasse `VOMACollection` gesammelt werden. Mit der Methode `getVOReport()` kann schließlich für eine ausgewählte VO-Konfiguration und einen gewünschten Zeitraum eine komplette Statistik der VO-Aktivitäten über diesen Zeiraum erzeugt werden. Die Spezifikation des Reports ist fallspezifisch. Die Sequenzen der Funktionen `getVOLifecycle()`, `traceLifecycle()` und `getVOReport()` sind jeweils offensichtlich und werden daher hier nicht weiter detailliert.

Transitionen im VO-Lebenszyklus. Transitionen im VO-Lebenszyklus werden konzeptionell durch Ereignisse (Signale) ausgelöst. Ereignisse im Kontext des VO-Provisionings können dabei Fehlersituationen sein oder gezielte Transitionskommandos (siehe auch Abbildung 5.50). Zur Durchführung von Transitionen stehen die Methoden `lifecycleTransition()` für eine reguläre Transition und `lifecycleForcedTransition()` für eine forcierte Transition zur Verfügung. Beide Methoden verwenden als Parameter eine VO-Konfiguration und eine Status-Kodierung. `lifecycleTransition()` setzt den Lebenszyklus-Status (Klasse `VOMAEntityLifecycle`, Attribut `lifecycleStatus`) auf den angegebenen Wert. Die möglichen Werte sind aus Abbildung 5.50 ersichtlich. Abbildung 5.51 zeigt exemplarisch einen Ausschnitt aus dem Workflow der regulären Transition.

Die dort angeführten Entscheidungen basieren auf den aktuellen Belegungen der Attribute `lifecycleCurrentPriority`, `lifecycleCurrentPriority` und `lifecycleCurrentPriority` der Klasse `VOMAEntityLifecycle` des Informationsmodells.

Starten, Aussetzen und Wiederaufnahme von Lebenszyklen. Zum Starten, Stoppen und Wiederaufnehmen von VO-Lebenszyklen nach einem Stopp werden vom VO-Provisioning spezialisierte Transitionsfunktionen angeboten:

- Ein VO-Lebenszyklus wird durch die Funktion `startVO()` gestartet, indem unter Verwendung der Funktion `lifecycleTransition()` eine Transition in den Zustand `VOstarted` durchgeführt wird, nachdem vorher geprüft wurde, ob sich der Lebenszyklus im Zustand `VOconfigured` befand.
- Ein VO-Lebenszyklus wird durch die Funktion `resumeVO()` wiederaufgenommen, indem unter Verwendung der Funktion `lifecycleTransition()` eine Transition in den Zustand `VOstarted` durchgeführt wird, nachdem vorher geprüft wurde, ob

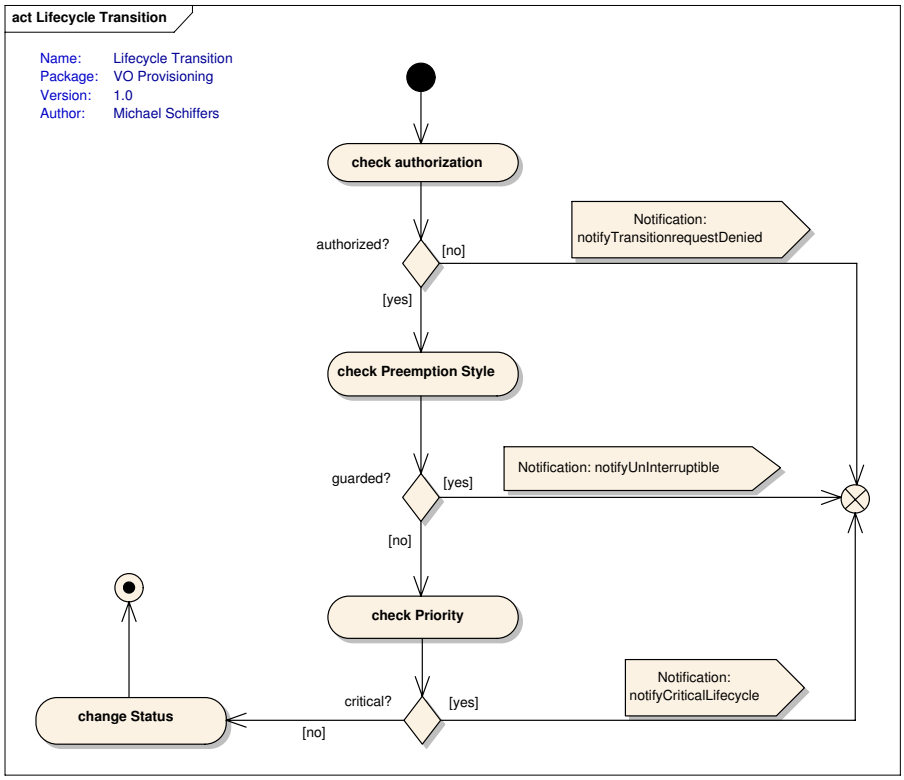


Abbildung 5.51: Reguläre Transition im VO-Lebenszyklus

sich der Lebenszyklus im Zustand `VOstopped` befand. Der letzte Zustand der VO, auf dem wieder neu aufgesetzt wird, ist aus den Attributen und Methoden der Klasse `VOMALifecyclePhase` ableitbar.

- Ein VO-Lebenszyklus wird durch die Funktion `stopVO()` gestoppt, indem unter Verwendung der Funktion `lifecycleTransition()` eine Transition in den Zustand `VOstopped` durchgeführt wird, nachdem vorher geprüft wurde, ob sich der Lebenszyklus im Zustand `VOstarted` befand.

Soll eine forcierte Transition erfolgen, wird bei `startVO()`, `stoppVO()` und `resumeVO()` statt der Funktion `lifecycleTransition()` die Funktion `lifecycleForcedTransition()` verwendet. Die Entscheidung, welche der beiden Methoden jeweils angewandt werden soll, erfolgt durch die Angabe eines weiteren Parameters `transitionMethod`.

Konfigurierung von Lebenszyklen. Zum Konfigurieren von VO-Lebenszyklen wird vom VO-Provisioning die Methode `configureLifecycle()` angeboten. Diese Methode setzt entsprechende Attribute in der Klasse `VOMAEntityLifecycle` unter Berücksichtigung von Zugriffsrechten und Prioritäten. Die Ausprägung der Funktion ist offensichtlich.

Zusammen mit dem vorgestellen Objektmodell definieren diese Funktionen mit ihren Schnittstellen den Funktionsbereich *VO-Provisioning* des VOMA-Funktionsmodells.

5.5.5 Funktionsbereich *Accounting Management*

Der Funktionsbereich *Accounting Management* ist ebenfalls in der `CommunityDomain` angesiedelt und fokussiert primär auf die Sammlung und Verrechnung von Abrechnungseinheiten VOs als „Verbrauchsobjekt“ betreffend. Im Rahmen einer möglichen FCAPS-Funktionsstrukturierung auf der Community-Ebene kann die Erörterung des Accounting Managements als Beispiel für ähnliche Ausprägungen anderer Funktionsbereiche dienen, diese Diskussionen werden hier allerdings nicht weiter verfolgt.

Allgemeine Einordnung

Die Aufgaben des Accounting Managements liegen in Analogie zum OSI-Management [Hegering u. a., 1999; Radisic, 2003] in der Sammlung, Kostenermittlung und Abrechnung von VO-Nutzungen. Die betrifft im Kontext dieser Arbeit nicht den im Rahmen der Nutzung virtueller Ressourcen und Dienste anfallenden Verbrauch (*Usage Records*, diese werden beispielsweise in [Göhner u. a., 2006] behandelt), sondern die auf der Community-Ebene erforderlichen Daten zur Nutzung von VOs. So werden beispielsweise im IPY-Szenario Rabattierungsmuster in Erwägung gezogen, um Anreize zu schaffen, als Resource Provider (d.h. als Anbieter wissenschaftlich relevanter, aber privater, Datenbestände) mehreren VOs beizutreten. Ebenso soll über Funktionen des Accounting Managements Community-weit ein (kommerziell orientiertes) Regulativ geschaffen werden, Mangel bzw. Überfluß an Ressourcen auszugleichen. Dabei wird stillschweigend davon ausgegangen, dass die Kommerzialisierung von Grids weiter voranschreiten wird. Obwohl viele Fragestellungen im Zusammenhang mit dieser speziellen Auffassung Grid-weiten Accountings noch offen sind (siehe [Göhner u. a., 2006]), lassen sich dennoch einige Interaktionen identifizieren und – zumindest rudimentär – behandeln:

Abfrage von VO-Abrechnungseinheiten. Das Accounting Management bietet die Möglichkeit, individualisierte oder Gruppen-aggregierte VO Usage Records (VO-UR) abzugreifen und diese gegen definierte Sollwerte oder historische Daten zu vergleichen.

Generierung von Accounting Reports. Unabhängig davon bietet das Accounting Management die Möglichkeit, die VO-Nutzung von Individuen oder ganzen Gruppen und die damit verbundenen Kosten über einen definierten Zeitraum nachzuweisen.

Budgetierung. Das Accounting Management bietet eine Budget-Schnittstelle, über die VO-URs abgerechnet werden können und gegen vorgegebene Credit Limits verglichen werden können. Das Überschreiten von Credit Limits resultiert in Notifikationen und Justierungen der Kreditwürdigkeit, die wiederum zur Evaluierung von Individuen und Gruppen im Rahmen anderer Funktionen verwertet werden können.

Objektmodell zum Accounting Management

Die Objekte des Accounting Managements sind VO-URs und die damit assoziierten Objekte und Entitäten. VO-URs werden in der Klasse `VOUsageRecord` modelliert. Abbildung 5.52 stellt die Attribute dieser Klasse dar und zeigt, wie die Klasse mit der Klasse `VOMASpecificSLA` des Informationsmodells verbunden ist und geloggt wird (Spezialisierung der Klasse `LoggingSpecificEvent`). Kostensätze werden über SLAs (Klasse `VOMASpecificSLA`) festgelegt.

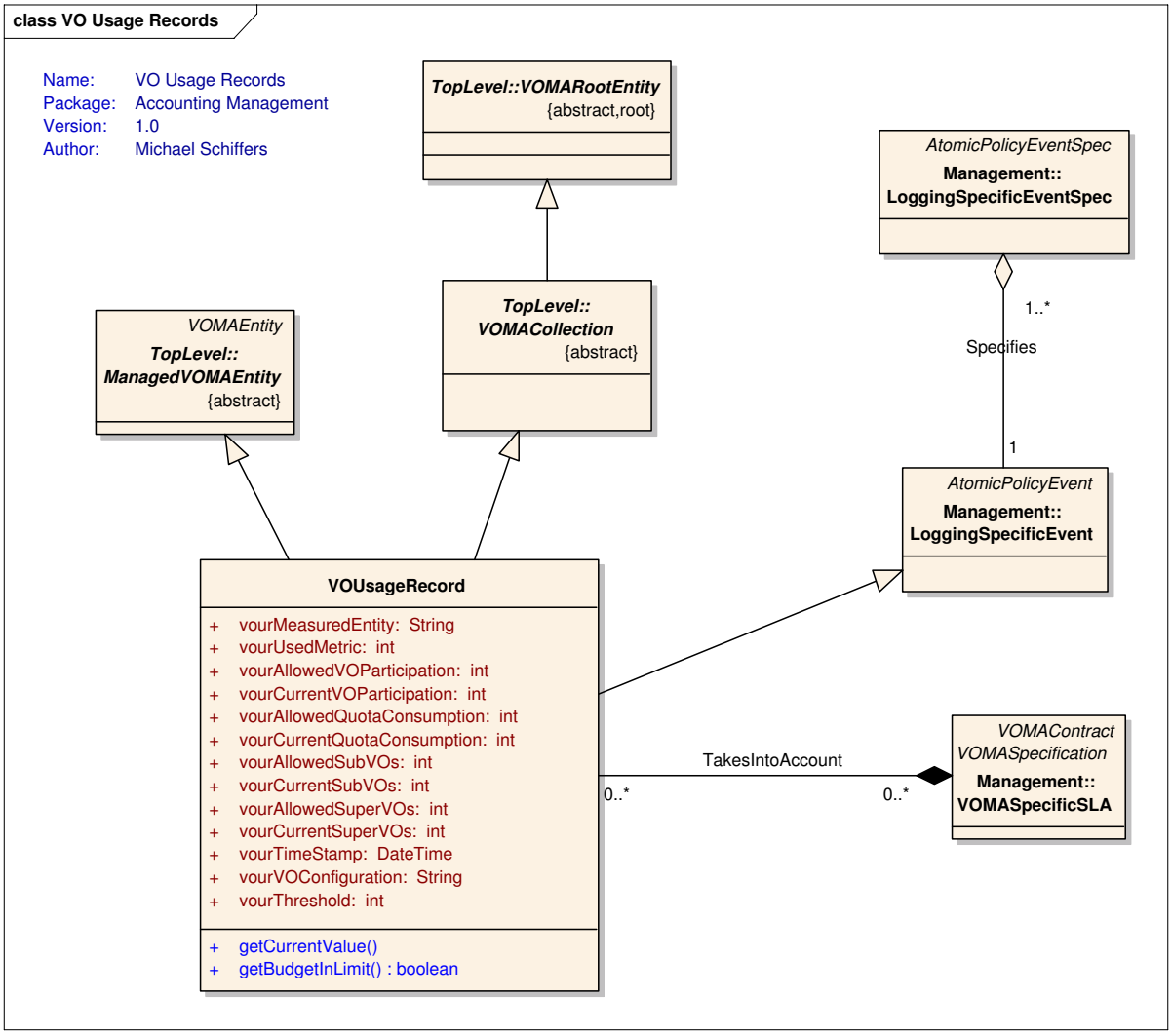


Abbildung 5.52: VO Usage Records

Im Einzelnen verwendet die Klasse `VOUsageRecord` die folgenden Attribute. Die aktuellen Werte werden über die Methode `getCurrentValue()` ermittelt.

`vourMeasuredEntity`**.** In diesem Attribut wird eine Referenz auf die beurteilte Entität hinterlegt. Dies können Individuen, Gruppen, Organisationen, Ressourcen oder Dienste sein.

`vourUsedMetric`**.** Dieses Attribut spezifiziert die verwendete Metrik dieses VO Usage Records.

`vourAllowedVOParticipation` **und** `vourCurrentVOParticipation`**.** Die erlaubte und aktuelle Anzahl möglicher VO-Teilnahmen wird hier definiert bzw protokolliert.

`vourAllowedQuotaConsumption` **und** `vourCurrentQuotaConsumption`**.** Der erlaubte und aktuelle Verbrauch von Kontigenten wird hier definiert bzw protokolliert.

`vourAllowedSubVOs` **und** `vourCurrentSubVOs`**.** Die erlaubte und aktuelle Anzahl möglicher Sub-VOs wird hier definiert bzw protokolliert.

`vourAllowedSuperVOs` **und** `vourCurrentSuperVOs`**.** Die erlaubte und aktuelle Anzahl möglicher Super-VOs wird hier definiert bzw protokolliert.

`vourTimeStamp`**.** Dies ist der Zeitstempel des aktuellen Eintrags.

`vourVOConfiguration`**.** In diesem Attribut ist der Bezug zu einer VO-Konfiguration verfügbar. Dadurch wird es möglich, auch historische Daten zu bestimmten VOs abzugreifen.

`vourThreshold`**.** In diesem Attribut wird ein allgemeiner Schwellwert für die Bestimmung von Budget-Überschreitungen festgesetzt.

Funktionen des Accounting Managements

Basierend auf diesen Objekten bietet der Funktionsbereich des Accounting Managements an seiner Schnittstelle die folgenden Funktionen an:

Abfrage von VO-Abrechnungseinheiten. Zur Abfrage von Abrechnungseinheiten (Klasse `VOUsageRecord`) stellt das Accounting Management die Methode `getVOUsageRecord(searchpattern)` bereit, über die – je nach Autorisierung – Teile eines VO-UR (z.B. die Abfrage des Attributs `vourCurrentSuperVOs`) abgefragt werden können, aber auch komplette VO-URs oder alle VO-URs, die den in `searchpattern` spezifizierten Kriterien genügen. Die Präsentation der Suchergebnisse erfolgt als Sequenz von Attribut-Wert-Paaren, die Plattform-spezifisch aufbereitet werden können. Damit besteht die Möglichkeit, individualisierte oder Gruppen-aggregierte VO-URs abzugreifen und diese mit definierten Sollwerten oder historischen Daten oder Daten anderer VO-Konfigurationen zu vergleichen. Die Sequenz der Funktion wird analog zur Abbildung 5.47 spezifiziert.

Generierung von Accounting Reports. Mit der Methode `getVOURReport()` bietet das Accounting Management für eine ausgewählte VO-Konfiguration ein entsprechendes Berichtswesen an. Die Sequenz dieser Funktion ist offensichtlich und wird hier nicht weiter detailliert.

Budgetierung. Das Accounting Management bietet mit der Methode `adjustVOURBudget()` eine einfache Budget-Schnittstelle an, über die die Sollwert-Attribute der Klasse `VOUsageRecord` und der Schwellwert `vourThreshold` gesetzt werden können. Über die Methode `getBudgetInLimit()` kann dann für ein Attribut bestimmt werden, ob sich der augenblickliche Wert im Limit befindet oder nicht. Falls nein wird eine entsprechende Notifikation gegeben (`notifyVOURBeyondThreshold`). Der Workflow der Funktion `adjustVOURBudget()` ist offensichtlich und kann hier übergangen werden.

Zusammen mit dem vorgestellen Objektmodell definieren diese Funktionen mit ihren Schnittstellen den Funktionsbereich *Accounting Management* des VOMA-Funktionsmodells.

5.5.6 Funktionsbereich *Local Management*

Der Funktionsbereich *Local Management* ist in der `RealOrganizationDomain` angesiedelt und fokussiert primär auf die Unterstützung des VO-Managements aus der Sicht der realen Organisationen.

Allgemeine Einordnung

Im Rahmen des VO-Managements besteht die Aufgabe des Local Managements in der Bereitstellung lokaler Ressourcen, Dienste und Individuen unter den Randbedingungen lokaler Policies, die in VO-spezifischen SLAs festgeschrieben werden. Das VO-Management hat keinen direkten Einfluss auf diese Entitäten, befindet sich also in einer reinen „Verbraucher“-Rolle. Auf der anderen Seite ist das Local Management an der Kenntnis der Performanz der eigenen Beiträge zur VO interessiert. Damit lassen sich für das Local Management die folgenden Interaktionen identifizieren:

Abfrage von lokal eingeschränkten VO-Konfigurationen. Dem Local Management muss die Möglichkeit gegeben werden, eine RO-spezifische Sicht auf die aktuelle VO-Situation innerhalb der Community zu erhalten. Dies beinhaltet die Darstellung einer Gesamtsicht und einer Individualsicht bzgl. Personen, Ressourcen und Dienste. Die eigentliche Verwendung der für eine VO bereitgestellten Ressourcen und Dienste wird an dieser Schnittstelle nicht angeboten, diese Daten stehen lokal sowieso zur Verfügung.

Notifikationen. Dem Local Management selbst muss die Möglichkeit geboten werden, das VO-Management über veränderte Randbedingungen zu informieren. Dies betrifft

Benachrichtigungen über missbräuchliche Ressourcennutzung ebenso wie die Modifizierung von Policies und SLAs im weitesten Sinn (also z.B. auch deren Kündigung und der Rückzug aus einer VO).

Objektmodell zum Local Management

Der Fokus des Local Managements liegt auf lokalen Sichten auf VO-Konfigurationen und deren assoziierte Klassen, auf Benachrichtigungen des VO-Managements und auf Änderungen lokal-geprägter Attribute des Informationsmodells. Letztere werden im Abschnitt 5.2.4 und Abbildung 5.17 ausführlich dargestellt, VO-Konfigurationen werden im Abschnitt 5.5.3 eingehend diskutiert. Eine weitergehende Betrachtung des Objektmodells ist daher nicht notwendig.

Funktionen des Local Managements

Basierend auf diesen Objekten bietet der Funktionsbereich des Local Managements an seiner Schnittstelle die folgenden Funktionen an:

Abfrage von lokal eingeschränkten VO-Konfigurationen. Zur Abfrage lokaler Sichten auf VO-Konfigurationen wird die Funktion `getLocalView()` angeboten, die im ersten Parameter die aufrufende Entität repräsentiert und im zweiten Parameter die zu prüfende Entität. Über einen Parameter `queryRange` kann zudem spezifiziert werden, ob sich die Abfrage auf eine spezifische VO-Konfiguration bezieht (`queryRange == singleVO`) oder auf alle VOs (`queryRange == multiVO`). Die Suche kann weiter eingeschränkt werden durch den Parameter `queryLifecycle`, der spezifiziert, welche VO-Konfigurationen betrachtet werden sollen. Die möglichen Werte sind die des Attributs `configurationState` der Klasse `VOConfiguration`. Der Workflow der Funktion `getLocalView()` ist offensichtlich und wird daher nicht weiter detailliert.

Notifikationen. Das Setzen lokaler Attribute im Informationsmodell erfolgt über die Standard-`Set`-Methoden auf den entsprechenden Klassen des Informationsmodells. Das Local Management übermittelt dabei die folgenden Notifikationen:

- `notifyPolicyUpdate`, falls eine lokale Policy sich geändert hat
- `notifySLAUpdate`, falls sich ein lokales SLA geändert hat
- `notifySLATermination`, falls ein SLA vom Local Management einseitig gekündigt wurde
- `notifyMisuse`, falls eine lokale Ressource oder ein Dienst missbräuchlich verwendet wurde oder versucht zu verwenden. In diesem Fall wird zusätzlich auch der Verursacher mitgeteilt

- `notifyConfigurationUpdate`, falls ein anderes Attribut einer VO-Konfiguration verändert wurde

Als Zusatzinformation wird die entsprechende VO-Konfiguration mitgeliefert. Notifikationen werden analog zum Abschnitt 5.5.3 behandelt.

Zusammen mit dem vorgestellen Objektmodell definieren diese Funktionen mit ihren Schnittstellen den Funktionsbereich *Local Management* des VOMA-Funktionsmodells.

5.5.7 Funktionsbereich *VO Management*

Der Funktionsbereich *VO Management* ist in der `VirtualOrganizationDomain` angesiedelt und fokussiert primär auf die Betriebsphase einer Virtuellen Organisation. Das Attribut `configurationState` der die VO repräsentierenden Klasse `VOConfiguration` ist in diesem Fall mit dem Wert `active` belegt.

Allgemeine Einordnung

Im Rahmen des VO-Managements besteht die Aufgabe des auf der Ebene der Virtuellen Organisationen angesiedelten VO Managers in der Administration der VO. Damit lassen sich für das Local Management die folgenden Interaktionen identifizieren:

Abfrage von VO-spezifischen Belegungen des Informationsmodells. Dem VO Management muss die Möglichkeit gegeben werden, eine VO-spezifische Sicht auf das Informationsmodell zu erhalten. Dies beinhaltet die Darstellung einer Gesamtsicht und einer attributierten Sicht.

Abfrage von lokal verfügbaren Informationen. Dem VO Management muss zudem die Möglichkeit gegeben werden, auf die VO-spezifische Informationsbasis des Local Managements zuzugreifen.

Setzen von VO-spezifischen Attributen des Informationsmodells. Dem VO Management muss schließlich die Möglichkeit eingeräumt werden, Attribute des Informationsmodells zu setzen.

Objektmodell zum VO Management

Der Objektmodell des VO Managements ist mit dem VOMA-Informationsmodell identisch, lediglich die Sicht auf das Informationsmodell ist VO-spezifisch. Insofern ist eine weitere Erörterung des Objektmodells an dieser Stelle nicht notwendig.

Funktionen des VO Managements

Basierend auf dem Informationsmodell (in der VO-spezifischen Sicht) bietet der Funktionsbereich des VO Managements an seiner Schnittstelle fast ausschließlich `get-` bzw `set-`Funktionen an:

Abfrage von VO-spezifischen Belegungen des Informationsmodells. Die Abfrage von VO-spezifischen Belegungen des Informationsmodells wird von Attribut-spezifischen `get`-Methoden erfüllt. Um die Privatsphäre (*privacy*) anderer VOs zu schützen, wird allerdings in einer vorgeschalteten Autorisierungsanalyse sichergestellt, dass nur auf VO-spezifische Informationen zugegriffen werden kann.

Abfrage von lokal verfügbaren Informationen. Zur Abfrage RO-spezifischer Informationen nutzt das VO Management die Funktion `getLocalView()` des Local Managements.

Setzen von VO-spezifischen Attributen des Informationsmodells. Das Setzen VO-spezifischer Attribute im Informationsmodell erfolgt über die Standard-`Set`-Methoden auf den entsprechenden Klassen des Informationsmodells, allerdings mit eingeschränkter Berechtigung.

Zusammen mit dem vorgestellen Objektmodell definieren diese Funktionen mit ihren Schnittstellen den Funktionsbereich *VO Management* des VOMA-Funktionsmodells.

5.5.8 Funktionsbereich *Member Management*

Der Funktionsbereich *Member Management* ist in der `VirtualOrganizationDomain` angesiedelt und fokussiert primär auf die Mitgliedschaften zu einer Virtuellen Organisation. Das Attribut `configurationState` der die VO repräsentierenden Klasse `VOConfiguration` ist in diesem Fall mit dem Wert `active` belegt.

Allgemeine Einordnung

Im Rahmen des VO-Managements besteht die Aufgabe des Member Managements in der Administration von VO-Mitgliedschaften, von VO-Rollen und der Relation von Mitgliedern und Rollen. Die in den Abschnitten 5.2.5 und 5.2.6 durchgeführte Detail-Darstellung des Member/Role-Managements hat gezeigt, dass in diesem Kontext zwischen Anwärtern auf eine Mitgliedschaft (`Applicants`) und autorisierten Mitgliedern (`Participants`) unterschieden werden muss. Damit lassen sich für das Member Management die folgenden Interaktionen identifizieren:

Abfrage von Mitglieds- und Rolleninformationen. Dem Member Management muss die Möglichkeit gegeben werden, eine Mitglieds- und Rollen-spezifische Sicht auf das Informationsmodell zu erhalten. Dies beinhaltet die Darstellung einer Gesamtsicht und einer attributierten Sicht.

Management von Mitgliedschaften. Dem Member Management muss die Möglichkeit gegeben werden, `Applicants` aufzunehmen, `Applicants` in `Participants` der verschiedenen Ausprägungen (`guest`, `regular`, `specific`) umzuwandeln, Attribute der `Applicants` und `Participants` zu ändern sowie `Applicants` und `Participants` zu löschen. Zusätzlich müssen `Applicants` ablehnbar sein.

Management von Rollen. Dem Member Management muss die Möglichkeit gegeben werden, `Rollen` aufzunehmen, in ihren Attributen zu ändern und Rollen zu löschen.

Assoziierung von Rollen und Participants. Dem Member Management muss die Möglichkeit gegeben werden, `Participants` Rollen zuzuordnen und umgekehrt.

Notifikationen. Dem Member Management muss schließlich die Möglichkeit eingeräumt werden, über Veränderungen im Mitglieds-/Rollen-Status zu unterrichten oder aber selbst unterrichtet zu werden, falls beispielsweise das Local Management Mitgliedschafts-relevante Änderungen mitzuteilen hat.

Objektmodell zum Member Management

Der Objektmodell des Member Managements ist mit dem VOMA-Informationsmodell identisch, lediglich die Sicht auf das Informationsmodell ist Mitglieds- bzw. Rollenspezifisch. Insofern ist eine weitere Erörterung des Objektmodells an dieser Stelle nicht notwendig. Dies ist bereits den den Abschnitten 5.2.5 und 5.2.6 geschehen.

Funktionen des Member Managements

Basierend auf dem Informationsmodell (in der Mitglieds/Rollen-spezifischen Sicht) bietet der Funktionsbereich des Member Managements an seiner Schnittstelle fast ausschließlich `get`- bzw `set`-Funktionen an:

Abfrage von Mitglieds- und Rolleninformationen. Die Abfrage von Mitglieds- und Rollen-spezifischen Belegungen des Informationsmodells wird von Attribut-spezifischen `get`-Methoden erfüllt. Um die Privatsphäre von Mitgliedern und Organisationen zu schützen, wird allerdings in einer vorgeschalteten Autorisierungsanalyse sichergestellt, dass nur auf autorisierte Informationen zugegriffen werden kann.

Management von Mitgliedschaften. Für das Management von Mitgliedschaften stellt das Member Management die Funktionen `requestParticipation ()` und `withdrawParticipation()` bereit. Beide Funktionen nutzen die im Abschnitt 5.2.5 beschriebenen Klassen und deren Attribute, um entweder eine Teilnahme an einer VO zu beantragen oder diesen Antrag zurückzuziehen. Der eigentliche Approval-Prozess ist fallspezifisch und wird deshalb hier nicht dargestellt. Im Rahmen des Deployments kann dieser in Spezialisierungen der `VOMASpecification`-Klasse und der `VOMAWorkflow`-Klasse separat modelliert werden. „Genehmigte“ Teilnehmer an

einer VO können als `guest`, `regular` oder `specific` klassifiziert werden (Attribute `TypeOfApplication` der Klasse `VOApplicants`). Attribut-Änderungen erfolgen ansonsten über die Standard-`Set`-Methoden. Erfüllt ein `Applicant` nicht die geforderten Voraussetzungen, wird er abgelehnt und eine entsprechende Notifikation gesendet (`notifyParticipationReject`), im positiven Fall wird ebenfalls benachrichtigt (Notifikation `notifyParticipationAccepted`).

Management von Rollen. Für das Management von Rollen werden im Member Management die standardmäßigen `get-` bzw. `set`-Methoden für die im Abschnitt 5.2.6 beschriebenen Klassen verwendet.

Assoziierung von Rollen und Participants. Für die Abbildung von `Participants` auf Rollen bzw. Rollen auf `Participants` nutzt das Member Management die Funktionen `assignRole()` bzw `unassignRole()`, mit denen ein angegebener `Participant` mit einer angegebenen Rolle assoziiert wird bzw. diese Assoziation aufgehoben wird. Die `unassignRole()`-Methode kann unter Umständen zu Rollen führen, die auf keine `Participants` abgebildet sind. Für diesen Fall wird die Notifikation `notifyOrphanedRole` gesendet, auf die fallspezifisch reagiert wird. Weitere Details zum Management von Rollen sind im Abschnitt 5.2.6 zu finden.

Zusammen mit dem vorgestellten Objektmodell definieren diese Funktionen mit ihren Schnittstellen den Funktionsbereich *Member Management* des VOMA-Funktionsmodells.

5.5.9 Funktionsbereich *Resource/Service Management*

Der Funktionsbereich *Resource/Service Management* ist in der `VirtualOrganizationDomain` angesiedelt und fokussiert primär auf die Bereitstellung virtueller Dienste und Ressourcen. Das Attribut `configurationState` der die VO repräsentierenden Klasse `VOConfiguration` ist in diesem Fall mit dem Wert `active` belegt.

Allgemeine Einordnung

Im Rahmen des VO-Managements besteht die Aufgabe des Resource/Service Managements in der Administration virtueller Ressourcen und Dienste. In den Abschnitten 5.2.7 und 5.2.8 wurden die entsprechenden Strukturen des Informationsmodells im Detail dargestellt. Für das Resource/Service Management auf der VO-Ebene lassen sich die folgenden Interaktionen für das VO-Management identifizieren:

Abfrage von Resource/Dienstinformationen. Dem Resource/Service Management muss die Möglichkeit gegeben werden, eine Resource/Service-spezifische Sicht auf das Informationsmodell zu erhalten. Dies beinhaltet die Darstellung einer Gesamtsicht und einer Attribut-bezogenen Sicht.

Setzen von Resource/Dienstinformationen. Dem Resource/Service Management muss die Möglichkeit gegeben werden, Resource/Service-spezifische Attribute im Informationsmodell zu setzen.

Abfrage von Nutzungsinformationen. Dem Resource/Service Management muss schließlich die Möglichkeit gegeben werden, die Aufbereitung von Nutzungsstatistiken durch das Sammeln entsprechender Usage Records zu unterstützen.

Objektmodell zum Resource/Service Management

Der Objektmodell des Resource/Service Managements ist mit dem VOMA-Informationsmodell identisch, lediglich die Sicht auf das Informationsmodell ist Resource/Service-spezifisch. Insofern ist eine weitere Erörterung des Objektmodells an dieser Stelle nicht notwendig. Dies ist bereits in den Abschnitten 5.2.7 und 5.2.8 geschehen.

Funktionen des Resource/Service Managements

Basierend auf dem Informationsmodell (in der Resource/Service-spezifischen Sicht) bietet der Funktionsbereich des Resource/Service Managements an seiner Schnittstelle ausschließlich `get-` bzw `set`-Funktionen an:

Abfrage und Setzen von Resource/Dienstinformationen. Die Abfrage und das Setzen von Resource/Service-spezifischen Belegungen des Informationsmodells wird von Attribut-spezifischen `get-` bzw. `set`-Methoden erfüllt.

Abfrage von Nutzungsinformationen. Nutzungsinformationen werden in der Klasse `VOMACollection` mit den Typen `VirtualResourceUsage` bzw. `VirtualServiceUsage` hinterlegt und stehen über `get`-Methoden zur Verfügung. Eine gesonderte Funktion ist nicht notwendig.

Zusammen mit dem vorgestellen Objektmodell definieren diese Funktionen mit ihren Schnittstellen den Funktionsbereich *Resource/Service Management* des VOMA-Funktionsmodells.

5.5.10 Zusammenfassung des Funktionsmodells

In diesem Abschnitt wurden die Funktionsbereiche der VO-Managementarchitektur (*Configuration Management*, *VO-Provisioning*, *Accounting Management*, *Local Management*, *Member Management*, *Resource/Service Management* und *VO Management*) mit ihren spezifischen Objektmodellen, ihrer Einbindung in das VOMA-Informationsmodell und den an ihren Schnittstellen zur Verfügung gestellten Funktionen dargestellt. Tabelle 5.17 führt die identifizierten Funktionen (unabhängig von `get-`/`set`-Methoden) und die wesentlichen Notifikationen noch einmal übersichtlich auf. Als „Library“ von Funktionsbausteinen betrachtet, unterstützen sie die Konstruktion von Managementanwendungen, die auf den Lebenszyklus Virtueller Organisationen fokussieren.

	Funktionsbereich					
Funktion bzw. Notifikation	Configuration	VO-Provision	Accounting	Local	VO-Management	Member	Resource
queryVOConfiguration()	✓					
requestVOConfiguration()	✓					
compareVOConfiguration()	✓					
getVOLifecycle()		✓				
traceLifecycle()		✓				
getVOReport()		✓				
lifecycleTransition()		✓				
lifecycleForcedTransition()		✓				
startVO()		✓				
resumeVO()		✓				
stopVO()		✓				
getVOUsageRecord()			✓			
getVOURReport()			✓			
adjustVOURBudget()			✓			
getLocalView()				✓		
requestParticipation()					✓	
withdrawParticipation()					✓	
assignRole()					✓	
unassignRole()					✓	
notifyReceived()	✓					
notifyRejected()	✓					
notifyAccepted()	✓					
notifyVOURBeyondThreshold()			✓			
notifyPolicyUpdate()				✓		
notifySLAUpdate()				✓		
notifySLAtermination()				✓		
notifyMisuse()				✓		
notifyConfigurationUpdate()				✓		
notifyParticipationReject()					✓	
notifyParticipationAccepted()					✓	
notifyOrphanedRole()					✓	

Tabelle 5.17: Funktionen und Notifikationen des VOMA-Funktionsmodells im Überblick

5.6 Zusammenfassung

In diesem Kapitel wurden die Teilmodelle der VO-Managementarchitektur VOMA auf der Basis der im Kapitel 3 abgeleiteten Anforderungen definiert. Im Informationsmodell wurden Virtuelle Organisationen und die mit ihnen verbundenen Entitäten als Managementobjekte beschrieben. Dabei war zu berücksichtigen, dass VO-Managementprozesse auf der Community-Ebene, der Ebene realer Organisationen und der Ebene der Virtuellen Organisationen ausgeprägt sind. Es wurde gezeigt, wie VOs zu identifizieren, zu überwachen und durch Operationen zu manipulieren sind. Im Organisationsmodell wurde anschließend dargestellt, welche Rollen auf diese Informationen zugreifen können und wie deren Interaktionsmuster gestaltet sind. Im Kommunikationsmodell wurde festgelegt, wie die Rollen zu Managementzwecken kommunizieren und welche Basisdienste erforderlich sind. Schließlich wurde im Funktionsmodell der Komplex „VO-Management“ anhand der Rollen und des Lebenszyklus einer VO strukturiert. Gezeigt wurde, dass VO-Konfigurationen die Kernstrukturen für die Funktionalbereiche *Configuration Management*, *VO Provisioning*, *Accounting Management*, *Local Management*, *Resource/Service Management*, *Member Management* und *VO Management* bilden. Im Funktionsmodell wurden die an den Schnittstellen bereitzustellenden Methoden detailliert spezifiziert. Damit steht nun insgesamt ein „Baukasten“ für das Management Virtueller Organisationen in Grids zur Verfügung.

Die Entwicklung der Architektur erfolgt insgesamt nach dem MDA-Prinzip. Bisher wurde – diesem Ansatz folgend – ein Plattform-unabhängiges Modell (VOMA-PIM) festgelegt. Dieses gilt es nun wegen des Grid-Fokus dieser Arbeit auf die spezifische Web Services-Plattform zu transformieren. Das ist der Gegenstand des nachfolgenden Kapitels 6.

Kapitel 6

WSDM-spezifische Transformation der Architektur

Inhalt des Kapitels

Nach der Spezifikation des Plattform-unabhängigen VOMA-Modells im Kapitel 5 (VOMA-PIM) schließt sich in diesem Kapitel mit dem Fokus auf das Rahmenwerk des Web Services Distributed Managements (WSDM) eine Plattform-spezifische Sicht auf die VO-Managementarchitektur an (VOMA-PSM). Rein formal werden im MDA-Ansatz Plattform-spezifische Sichten über Transformationen generiert [Kleppe u. a., 2003][1]. Da Modelle ganz allgemein Instanzen von Metamodellen sind (Abschnitt 5.1), wird zur Darstellung einer Transformation auf der Metaebene eine Beschreibung benötigt, die die Elemente des Zielmodells zu den Elementen des Quellmodells in Beziehung setzt (siehe auch [Gruhn u. a., 2006] und [Koch u. a., 2007] sowie Abbildung 6.1).

Der Schwerpunkt dieser Arbeit liegt auf dem VO-Management in Grids. Da dieser Themenkomplex bisher nicht systematisch untersucht wurde, existieren auch keine auf Organisationen fokussierenden Managementkonzepte. Eine für Grids angestrebte Umsetzung der VO-Managementarchitektur muss sich deshalb an den in diesem Umfeld relevanten Plattformen mit ihren zum Teil erheblichen konzeptionellen und sprachlichen Einschränkungen

[1] Eine Transformation ist nicht identisch mit einer *Translation*, da Transformationen die Bildung von Homomorphismen (n:m-Beziehungen) zwischen Elementen des Ausgangsmodells und Elementen des Zielmodells zulassen [Arlow u. Neustadt, 2003].

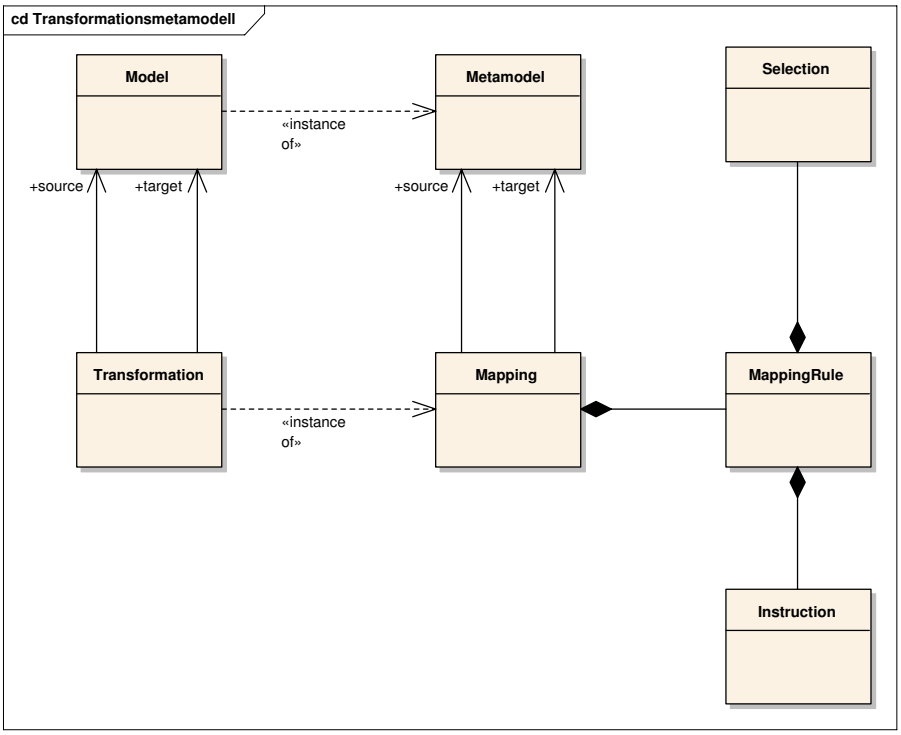


Abbildung 6.1: MDA-Transformationsmetamodell nach [Gruhn u. a., 2006]

orientieren. Diese sind durch die Web Services-Architektur im Allgemeinen und die Rahmenwerke des Web Services Distributed Managements (WSDM), des Web Services Resource Frameworks (WSRF) und der Open Grid Services Architecture (OGSA) im Besonderen gegeben. Zusammengefasst werden sie hier als Grid-PSM bezeichnet. Für den Rest dieses Kapitels spielen jedoch nur die WSDM- und WSRF-Komponenten eine Rolle.

Für die Abbildung des VOMA-PIMs auf ein Grid-PSM wird hier keine formal vollständige Modellierung der Zielumgebung und der dadurch induzierten Transformationen – wie sie methodisch erforderlich wäre – geliefert. Diese würde den Rahmen dieser Arbeit erheblich sprengen und kann hier nicht vorgenommen werden. Selbst Standardisierungsgremien [Czajkowski u. a., 2005; Foster u. a., 2006; OASIS, 2006g, h] beginnen erst jetzt, Teilaspekte der damit zusammenhängenden Fragestellungen zu adressieren. Das Ziel dieses Kapitel kann deshalb nur sein, Kapitel 5 methodisch sauber fortzusetzen, indem die erforderlichen Transformationsmechanismen *in concreto* dargestellt werden. Insbesondere sind dabei die folgenden Fragen zu beantworten:

- Wie werden die VOMA-PIM-Klassen, deren Attribute, Methoden, Assoziationen und Abhängigkeiten auf das Grid-PSM abgebildet?
- Auf welche Konstrukte des Grid-PSM werden die VOMA-PIM-Rollen abgebildet?
- Auf welche Kommunikationsmechanismen werden die VOMA-PIM-Mechanismen im

Grid-PSM abgebildet?

- Wie werden die Funktionen des Funktionsmodells abgebildet?

In der Praxis existiert zwar eine Vielzahl von Ansätzen, um MDA-Transformationsmechanismen zu implementieren [Arlow u. Neustadt, 2003; Frankel, 2003; Gruhn u. a., 2006; Kleppe u. a., 2003; Koch u. a., 2007; Petrasch u. Meimberg, 2006; Raistrick u. a., 2004]. Auf diese soll hier allerdings nicht weiter eingegangen werden und deren Anwendbarkeit soll hier auch nicht demonstriert oder gar bewertet werden. Dies wird an anderer Stelle zur Zeit unter Verwendung des Werkzeugs *AndroMDA* [Breuer, 2004] in weiterführenden eigenen Arbeiten zum VO-Management in Grids untersucht [Amikem, 2007; Cojocaru, 2007; Hellwig, 2007]. Zu erwähnen ist an dieser Stelle auch die von der Object Management Group (OMG) veröffentlichte Queries, Views and Transformations (QVT)-Spezifikation zur Abfrage und Transformation von Metamodellen ([OMG, 2005]).

Stattdessen soll im Folgenden der Fokus auf der Angabe konkreter Abbildungsvorschriften liegen. Diese werden jedoch nicht formal als Konstrukte einer Transformationssprache wie QVT [OMG, 2005] oder der im Rahmen des Generative Model Transformer (GMT)-Projektes der Eclipse Foundation entwickelten Atlas Transformation Language (ATL) [Gruhn u. a., 2006] beschrieben, sondern über einfache Zuordnungs-Templates. Dies ist ein pragmatisch orientierter Ansatz, solange die QVT-Spezifikationen noch nicht in implementierter Form vorliegen [Bohlen, 2006].

Um die Diskussion zu strukturieren, folgt dieses Kapitel der Vorgehensweise des Kapitels 5, indem zunächst Konstrukte des Informationsmodells aus Abschnitt 5.2 betrachtet werden, gefolgt von Konstrukten des Organisationsmodells aus Abschnitt 5.3, des Kommunikationsmodells aus Abschnitt 5.4 und schließlich des Funktionsmodells aus Abschnitt 5.5. Für eine kurze Einführung in die darunter liegenden Kernkonzepte (z.B. Manageability, WS-Resources) wird auf Kapitel 2 und die dort angegebene einschlägige Literatur verwiesen.

Die hier diskutierte WSDM-spezifische Transformation bildet die Elemente des VOMA-PIMs auf Konstrukte der WSDM- und WSRF-Rahmenwerke ab. Dabei orientiert sich die Diskussion an den folgenden Spezifikationen:

- Web Services Distributed Management: Management of Web Services (WSDM-MOWS), Version 1.1 [OASIS, 2006d]
- Web Services Distributed Management: Management Using Web Services (MUWS 1.1) Part 1 [OASIS, 2006e]
- Web Services Distributed Management: Management Using Web Services (MUWS 1.1) Part 2 [OASIS, 2006f]
- WS-CIM Mapping Specification [DMTF, 2006b]
- Web Services Resource 1.2 (WS-Resource) [OASIS, 2006i]
- Web Services Resource Lifetime 1.2 (WS-ResourceLifetime) [OASIS, 2006k]

- Web Services Topics 1.3 (WS-Topics) [OASIS, 2006n]
- Web Services Base Notification 1.3 (WS-BaseNotification) [OASIS, 2006b]

Als Ausgangsbasis der folgenden Überlegungen dienen zusätzlich die Abbildungen 2.27 (Seite 67) und 6.2 sowie die Beschreibungen im Abschnitt 2.3.4.

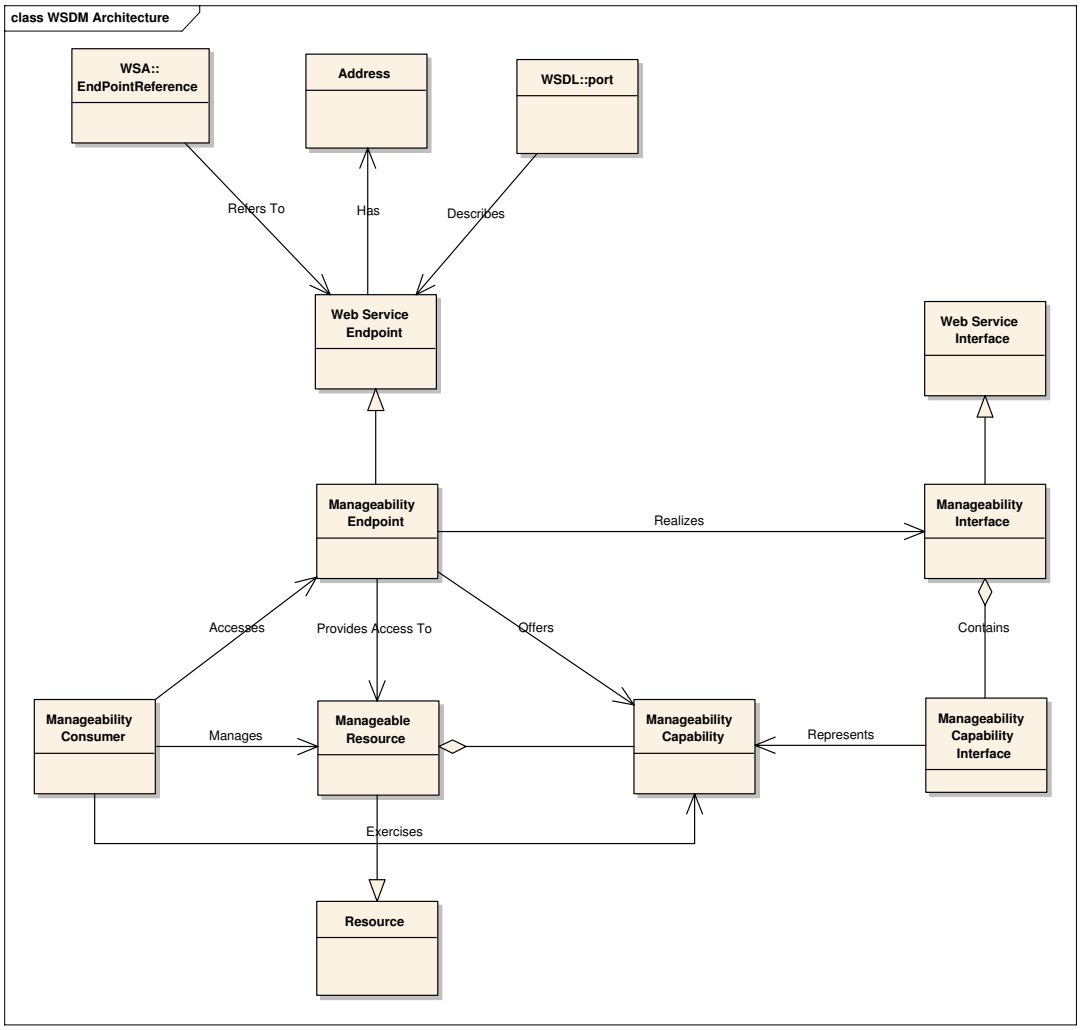


Abbildung 6.2: WSDM-Architektur nach [OASIS, 2006e]

Die WSDM-MUWS-Spezifikation [OASIS, 2006e, f] definiert, wie Manageability-Schnittstellen repräsentiert werden und wie auf sie zugegriffen werden kann. Die WSDM-MOWS-Spezifikation [OASIS, 2006d] definiert dagegen, wie Web Services als *Manageable Resource* gemanagt werden können und wie auf deren *Manageability*-Schnittstelle durch MUWS zugegriffen werden kann. WSDM ist grundsätzlich Ressourcen-orientiert und wegen der lose gekoppelten Web Services-Umgebung nicht auf das klassische Manager/Agenten-Paradigma festgelegt, da Ressourcen ihre *manageable resources* direkt unterstützen können.

WSDM definiert zudem nicht, welche Informationen die Ressourcen an ihren Managementschnittstellen zur Verfügung stellen müssen. Dies wird stattdessen vom Ressourcenmodell (z.B. CIM) geleistet. WSDM basiert konzeptionell auf *Manageablitiy Capabilities*. Eine Manageability Capability besteht aus einer Menge von Eigenschaften, Operationen, Ereignissen und Metadaten, die eine bestimmte Managementaufgabe unterstützen. Als Standard-Capabilities sind von WSDM Identität, Beschreibung, Manageability-Charakteristik, korrelierbare Eigenschaften, Metrik, Konfiguration, Zustand, operationaler Status und *Advertisement* vorgesehen.

Diese Randbedingungen sind nicht ohne Folgen für die Transformation von VOMA in Grids: VOMA-Klassen, deren Assoziationen und die Vererbungshierarchien können syntaktisch zwar direkt auf das Grid-Umfeld übertragen werden, indem sie wieder auf Klassen, Assoziationen und Vererbungsmuster einer UML-Repräsentation des Grid-Umfeldes abgebildet werden. Die allerdings durch WSDM schon vorformulierten Abhängigkeiten der Klassen (siehe Abbildung 6.2) induzieren jedoch semantische Seiteneffekte. Diese haben zur Folge, dass Virtuelle Organisationen in WSDM als `Manageable Resource` erzeugt werden müssen mit den entsprechenden `Manageability Capabilities`, die über einen `Manageability Endpoint` angeboten werden (siehe [Cojocaru, 2007]). Damit wird für einige VOMA-Klassen der Bezug zur TopLevel-Klasse `VOMARootEntity` aufgebrochen. Dies ist aber nur ein konzeptioneller Nachteil, der durch andere Konstruktions- bzw. Transformationsmechanismen wieder aufgefangen werden kann (z.B. durch zusätzliche Assoziationen).

6.1 Transformation des VOMA-Informationsmodells

WSDM ist Modell-agnostisch konzipiert und definiert daher auch kein eigenes Ressourcen-Modell, sondern zielt auf die Integration bestehender Standardmodelle, insbesondere CIM [DMTF, 2006b]. Mit einer Abbildung des VOMA-Informationsmodells auf CIM (siehe Abschnitt 5.2.10) wäre damit konzeptionell die Transformation des VOMA-Informationsmodells durchgeführt. Da allerdings die Standardisierung der nachgelagerten Abbildung „WSDM auf CIM“ noch nicht abgeschlossen ist, müssen in der Zwischenzeit möglicherweise einige Zuordnungmechanismen fallspezifisch betrachtet und implementiert werden (z.B. die VOMA BaseTypes und die in VOMA verwendeten Stereotypen ≪`VOMARole`≫ und ≪`InteractionChannel`≫). Die Abbildung „VOMA auf CIM“ ist jedoch wegen unterschiedlicher syntaktischer und semantischer Modellauffassungen nicht trivial (siehe Abschnitt 5.2.10) und kann nur in Teilen automatisiert werden, eine vollständige Diskussion dieser Problematik würde über den Rahmen dieser Arbeit hinausgehen. Eine Alternative besteht in der direkten Verwendung des VOMA-Informationsmodells in WSDM, allerdings um den Preis einer nur rudimentären Interoperabilität zwischen dem VOMA-Modell und anderen Ansätzen sowie der Notwendigkeit, einen entsprechenden VO-MIB-Browser, entweder neu zu erstellen oder bestehende (z.B. CIMOM [Hegering u. a.,

1999]) zu adaptieren.

Wird CIM [DMTF, 2007] trotz der Herausforderungen verwendet, sind in der folgenden Tabelle einige Beispiel von Klassenzuordnungen zusammengefasst. Die Zuordnungen der damit zusammenhängenden Attribute, Methoden, Generalisierungen und Assoziationen ergeben sich aus dem jeweiligen Kontext in Abschnitt 5.2. Sie müssen im Einzelfall gesondert behandelt werden und evtl. anderen CIM-Klassen zugeordnet werden, um der Organisationssemantik dieser Arbeit gerecht zu werden. Dies führt in der Regel zu einem nicht zu unterschätzenden manuellen Aufwand.

Transformation *VOMA-Klassen auf CIM-Klassen (Beispiele)*

VOMA-Klasse	CIM-Klasse
VOMAPolicy	Policy
Group	Group
TrustedAuthorities	CredentialManagementService
VirtualOrganization	Organization
virtualStaff	virtualStaff
ComputingResource	LogicalElement
LogicalRealResource	LogicalElement
PhysicalRealResource	PhysicalElement
VirtualResource	LogicalElement
VirtualService	Service
Individual	Person
ManagedVOMAEntity	ManagedElement oder LogicalElement falls es sich um eine virtuelle Resource oder einen virtuellen Dienst handelt
RealOrganization	Organization
VOMACollection	Collection
VOMARootEntity	ManagedElement

6.2 Transformation des Organisationsmodells

Das VOMA-Organisationsmodell wird im konkreten Fall durch Instanziierungen der mit dem Organisationsmodell assoziierten Klassen des Informationsmodells umgesetzt. Insofern kann die Abbildung des VOMA-Organisationsmodells auf die Transformation des

Informationsmodells zurückgeführt werden. Dabei ist allerdings zu berücksichtigen, dass der Stereotyp «`VOMARole`» an adäquater Stelle im CIM berücksichtigt wird (beispielsweise in der Klasse `Role` des CIM User-Modells). Eine Unterscheidung der VOMA-Ebenen (Community, RealOrganization, VirtuelOrganization) ist im WSDM-Kontext wegen seiner „flachen“ Sichtweise nicht möglich, stellt aber kein Hindernis für die Transformation dar, da diese Unterscheidung primär der Strukturierung der Untersuchung diente. Auch die im Organisationsmodell verwendeten Interaktionskanäle benötigen keine Sonderbehandlung im Rahmen der Transformation, da sie lediglich einen funktional orientierten Strukturierungsmechanismus darstellten. Interaktionskanäle werden im WSDM-Kontext selbst als *WS-Resource* konzipiert mit dem Workflow des Interaktionskanals als eigentlicher Ressource und einer Web Services-Schnittstelle. Damit können Interaktionskanäle über eine `EndpointReference` referenziert werden. Folgerichtig wird der Stereotyp «`InteractionChannel`» auf die Klasse `Manageable Resource` in Abbildung 6.2 abgebildet.

6.3 Transformation des Kommunikationsmodells

Wie VOMA geht auch WSDM prinzipiell von einer Manager/Agenten-Beziehung aus. Insofern steht einer Transformation des Kommunikationsmodells nichts im Wege. Konkret werden die Kern-Operationen des Kommunikationsmodells wie folgt abgebildet:

Transformation *VOMA-Kommunikationsmodell*	
Kern-Operationen in VOMA	Operation des Zielsystems
discover	GetManageabilityReference
get	GetResourceProperty
query	QueryResourceProperty
put	SetResourceProperty
create	diverse Funktionen in [OASIS, 2006d, e, f]
delete	diverse Funktionen in [OASIS, 2006k]
subscribe	diverse Funktionen in [OASIS, 2006b, n]
notify	diverse Funktionen in [OASIS, 2006b, n]

Listing 6.1 zeigt ein Beispiel einer `GetManageabilityReference`-Nachricht nach [OASIS, 2006d].

Listing 6.1: Beispiel einer `GetManageabilityReference`-Nachricht nach [OASIS, 2006d]

```
<?xml version="1.0" encoding="utf-8"?>
<w:definitions xmlns:w="http://schemas.xmlsoap.org/wsdl/"
        xmlns:xs="http://www.w3.org/2001/XMLSchema"
        xmlns:wsrf-rp="http://docs.oasis-open.org/wsrf/rp-2"
        xmlns:mows="http://docs.oasis-open.org/wsdm/mows-2.xsd"
        xmlns:mowsw="http://docs.oasis-open.org/wsdm/mows-2.wsdl"
        targetNamespace="http://docs.oasis-open.org/wsdm/mows-2.wsdl">

                <w:types>
                        <xs:import namespace="http://docs.oasis-open.org/wsdm/mows-2.xsd"
                         schemaLocation="http://docs.oasis-open.org/wsdm/mows-2.xsd"/>
                </w:types>

                <w:message name="GetManageabilityReferencesRequest">
                        <w:part name="body" element="mows:GetManageabilityReferences"/>
                </w:message>
                <w:message name="GetManageabilityReferencesResponse">
                        <w:part name="body"
                                element="mows:GetManageabilityReferencesResponse"/>
                </w:message>

</w:definitions>
```

Die im Kommunikationsmodell aufgeführten Basisdienste RDS, SRS und NOS können über diese Kern-Operationen unter Verwendung von WSRF- bzw. WSDM-Metadata-Deskriptoren realisiert werden [OASIS, 2006d, e, f].

6.4 Transformation des Funktionsmodells

Das Problem der Abbildung des VOMA-Funktionsmodells auf die WSDM-Rahmenwerk kann im wesentlichen auf das Problem der Abbildbarkeit des Informations- und des Kommunikationsmodells reduziert werden (siehe auch [Keller, 1998]), die im Abschnitt 5.5 aufgeführten Funktionen wurden nämlich weitgehend mit Hilfe des Informationsmodells unter Verwendung der Dienste des Kommunikationsmodells definiert. Diese Sichtweise wird zusätzlich dadurch unterstützt, dass beide Modellansätze auf lose gekoppelten Infrastrukturmodellen basieren, so dass auch die grundlegenden Kommunikationsmechanismen abbildbar sind. Eine Implementierung der in Tabelle 5.17 auf Seite 268 aufgeführten Funktionen wird zur Zeit für einen Einsatz im D-Grid in [Amikem, 2007; Cojocaru, 2007; Hellwig, 2007] durchgeführt.

6.5 Zusammenfassung

In diesem Kapitel wurden die im Kapitel 5 spezifizierten Plattform-unabhängigen Informations-, Organisations-, Kommunikations- und Funktionsmodelle der VO-Managementarchitektur VOMA exemplarisch auf die WSDM-Plattform transformiert, die

in dieser Arbeit als Basis für das VO-Management in Grids betrachtet wird. Es wurde gezeigt, wie die für WSDM erforderlichen PIM/PSM-Transformationen im Kontext der diversen Standardisierungsbestrebungen prinzipiell durchgeführt werden können. Insbesondere ist für das Informationsmodell jedoch festzuhalten, dass eine automatische Abbildung zwar wünschenswert ist, die semantischen Probleme diese für die Praxis allerdings erschweren.

Angemerkt sei weiterhin, dass die WSDM-Plattform nicht die einzige Basis einer VO-Managementarchitektur darstellt. Alternativ kann das VOMA-PIM auch auf das Web Services Management Rahmenwerk [Arora u. a., 2005] abgebildet werden. In dieser Arbeit wird WSDM deswegen bevorzugt, weil es am weitesten fortgeschritten und verbreitet ist und von den Grid-Gremien *OGF* und *Globus Alliance* unterstützt wird. Beide Ansätze konvergieren im übrigen zur Zeit zum *WS Unified Management* [Cohen u. a., 2007]. Auch einer Verwendung von CORBA als VOMA-PSM steht grundsätzlich nichts im Wege, die Bedeutung im Grid-Umfeld ist allerdings eher gering.

Nach dieser kurzen Darstellung der Transformation wird nun im nächsten Kapitel 7 eine Methodik zur Anwendung der Architektur eingeführt.

Kapitel 7

Deployment und operativer Einsatz

Inhalt des Kapitels

Im Kapitel 5 wurde aus den im Kapitel 3 spezifizierten Anforderungen systematisch eine Plattform-unabhängige VO-Managementarchitektur (VOMA) gewonnen. Im Informationsmodell dieser Architektur wurde dargestellt, wie die im Rahmen des VO-Managements zu betrachtenden *managed objects* (MO) durch die diversen Modellelemente in der VO-MIB repräsentiert werden. Als Kernelemente haben sich dabei die Klassen `VOConfiguration`, `VirtualOrganization`, `VOParticipant`, `RoleInVO`, `VirtualResource` und `VirtualService` gezeigt (siehe auch Abbildung 7.1).

Im Organisationsmodell wurde anschließend spezifiziert, welche Rollen für das Management Virtueller Organisationen relevant sind und welche Sichten auf das VO-Management sie bedienen. Dabei konnten mit der `CommunityDomain`, der `RealOrganizationDomain`

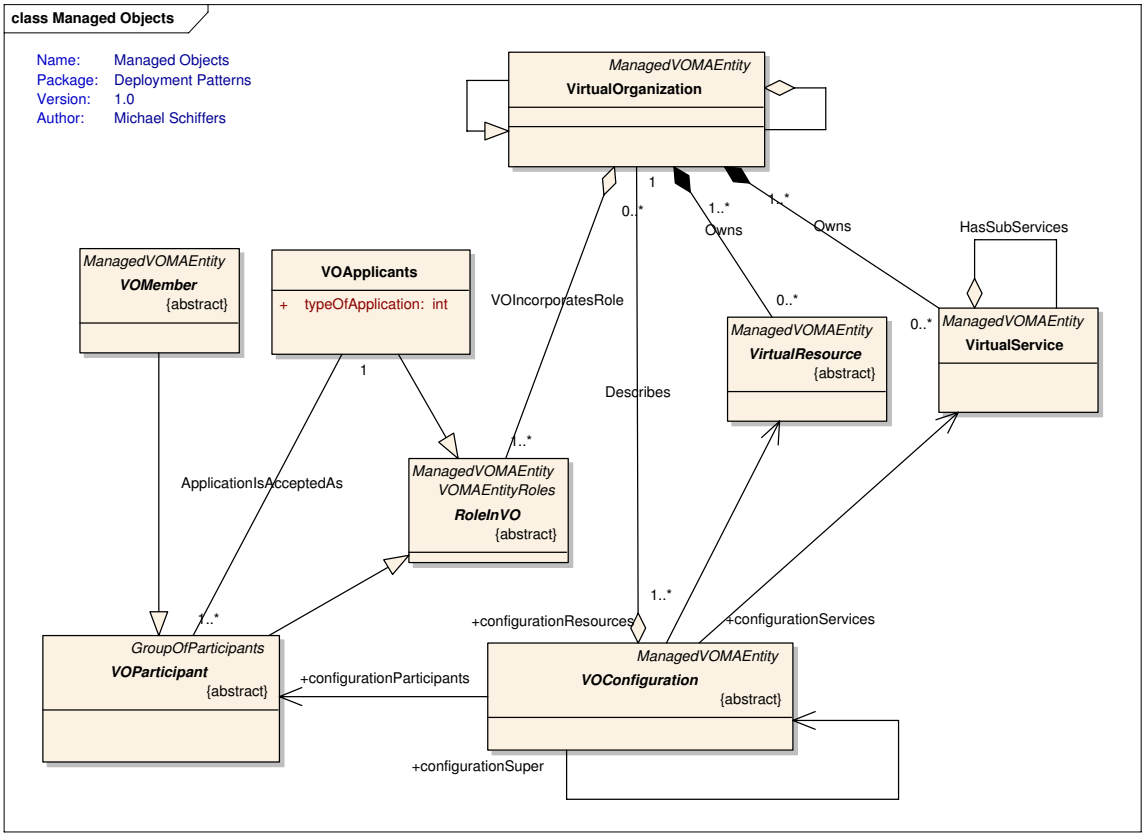


Abbildung 7.1: Managed Objects im Informationsmodell

und der `VirtualOrganizationDomain` die am VO-Management beteiligte Ebenen und deren Schlüsselrollen identifiziert werden. Im Kommunikationsmodell wurde dann gezeigt, welche Kommunikationsmechanismen diese Rollen nutzen, um Managementinformationen auszutauschen. In welchem Funktionskontext dies geschieht, wurde schließlich im Funktionsmodell dargestellt.

Mit diesen Modellen liefert VOMA ein (Plattform-unabhängiges) Rahmenwerk zum Management Virtueller Organisationen, das im Rahmen eines MDA-basierten Entwicklungsansatzes in Plattform-spezifische Modelle unter Verwendung entsprechender Vorschriften transformiert werden kann. Kapitel 6 hat dies am Beispiel einer WSDM-basierten Architektur prinzipiell demonstriert.

Nicht alle Transformationen müssen jedoch auf der Web Services-Plattform aufsetzen, sondern basieren möglicherweise auf proprietären Konzepten (z.B. DEISA). Der wesentliche Vorteil des MDA-Ansatzes – nämlich die weitgehende Wiederverwendbarkeit der verwendeten Modelle – kann jetzt für VO-Managementlösungen in Grids ausgenutzt werden, zumal aus dem MDA-Ansatz die notwendigen Terminologiefestlegungen sowie die generischen Architekturkonzepte folgen. Der Nachweis der Tragfähigkeit des Architekturansatzes wurde im Rahmen der Diskussion der Teilmodelle erbracht und wird in diesem Kapitel auf

anderer Ebene fortgesetzt.

7.1 Vorgehensweise

Die bisher erarbeitete Architektur stellt mit ihren Teilmodellen lediglich einen „Baukasten“ mit generischen Bausteinen, d.h. Konzepte und Regeln, bereit. In diesem Kapitel wird nun die notwendige Nutzungsanleitung für diesen Baukasten angegeben, die es einer am VO-Management direkt oder indirekt beteiligten Rolle ermöglicht, VO-bezogene Managementanwendungen zu planen, aufzubauen und zu betreiben. Die dazu notwendige Angabe einer entsprechenden Methodik erfordert ein allgemeines Konzept, das Plattform-agnostisch ist und flexibel an die individuellen Anforderungen einzelner „VO-Betreiber“ angepasst werden kann, nicht nur in der Entwurfsphase, sondern auch zur Laufzeit.

Um dieses zu erreichen, wird in diesem Kapitel in weitgehender Anlehnung an [Vatle, 2005] eine generische VO-Management-Infrastruktur (VOMA-I) mit den funktionalen Komponenten spezifiziert, die für ein effizientes und effektives Management Virtueller Organisationen erforderlich sind. Als VO-Management-Infrastruktur soll dabei die Gesamtheit der Entitäten verstanden werden, die an der Bereitstellung der Schnittstellen zum VO-Management im Rahmen eines Manager/Agenten-Paradigmas [Hegering u. a., 1999] beteiligt sind. Die Komponenten, die VOMA-I im eigentlichen Sinn implementieren, agieren dabei in einer aus dem OSI-Management bekannten Agentenrolle, die hier als VO-Management-Agent, kurz VO-Agent, bezeichnet wird. Die Verwendung des Manager/Agenten-Konzeptes ist in diesem Kontext insofern zulässig, als dieses auch das typische Kooperationsmuster in Grids darstellt [Foster u. a., 2006, 2003, 2001; Maciel, 2005].

Um die Komplexität des Infrastrukturmodells niedrig zu halten, werden bei der Spezifikation der VO-Management-Agenten zudem Konfigurationsmuster-Techniken [Ambler u. Hanscome, 1998; Arlow u. Neustadt, 2003; Bruegge u. Dutoit, 2003; Gamma u. a., 1995; Microsoft Corporation, 2004] eingesetzt, um dadurch eine Flexibilisierung des Deployments und der damit induzierten Nutzungsmethodik zu erreichen. Die Muster selbst sind wieder aus den Anforderungen im Kapitel 3 ableitbar.

Vor diesem Hintergrund gilt es also, die VO-Agenten so zu entwerfen, dass VOMA einer einfachen, standardisierten Nutzung in flexiblen Grid-Szenarios zugeführt werden kann. Dies ist die Fragestellung dieses Kapitels. Zur Beantwortung werden die folgenden Aspekte adressiert:

- Es muss sichergestellt werden, dass die hier definierte VO-Management-Infrastruktur auf bestehende Grid-Infrastrukturen abgebildet werden kann. Dieser Aspekt wird im Abschnitt 7.2 adressiert, indem die generische Infrastruktur VOMA-I (als PIM) auf Grid-Plattformen (als PSM) transformiert wird.

- Es müssen unterschiedliche Grid-Szenarios unterstützt werden. Dieser Frage widmet sich ebenfalls Abschnitt 7.2 durch die Angabe der funktionalen Komponenten eines VO-Agenten und deren fallspezifische Komposition, die die geforderte Flexibilität liefert.
- Wie kann ein MO-spezifisches Deployment unterstützt werden, denn nicht alle VO-Infrastruktur-Merkmale sind in allen Grid-Szenarios gleich ausgeprägt? Dieser Aspekt wird im Abschnitt 7.3 mit der Verwendung spezieller MO-abhängiger Konfigurationsmuster für VO-Agenten behandelt.

Anschließend wird im Abschnitt 7.4 eine Deploymentmethodik angegeben. Dabei wird unterschieden zwischen dem Einsatz der Architektur (Abschnitt 7.4.2) und dem Einsatz der Infrastruktur (Abschnitt 7.4.3). Im Abschnitt 7.4.4 wird dann an einigen Beispielen der Einsatz kurz demonstriert. Abschnitt 7.5 fasst die Ergebnisse dieses Kapitels zusammen.

7.2 VO-Management-Infrastrukturen

Die Anforderungsanalyse im Kapitel 3 und der Entwurf der VO-Managementarchitektur im Kapitel 5 (plus die spezifischen Betrachtungen im Kapitel 6) haben die Vielfalt der MOs und der damit induzierten möglichen Ausprägungen von VO-Agenten gezeigt. VO-Agenten bestehen aus einem generischen Teil, der die Schnittstelle zur Managementanwendung (hier generell als *Manager* bezeichnet) bildet und aus einem oder mehreren MO-spezifischen Teilen (hier gemäß [Vatle, 2005] als MO-spezifisches Agentenmodul (MOAM) bezeichnet). Während der generische Teil nur einmal zu Beginn eines Grid-Projektes entwickelt wird (bzw. von bestehenden Lösungen adaptiert wird), ist jedes MOAM MO-spezifisch. Es wird also beispielsweise ein MOAM zur Behandlung von VO-Konfigurationen geben oder ein MOAM zur Behandlung von VO-Teilnehmern. Die folgenden Ausführungen konzentrieren sich auf Bereitstellungsmuster der MOAMs und deren Sicht auf MOs.

So wie klassische Agenten über dedizierte Instrumentierungen mit den durch MOs repräsentierten Ressourcen interagieren, kapseln VO-spezifische „virtuelle“ Instrumentierungen die für VO-Agenten zugänglichen „Ressourcen“. Abbildung 7.2 zeigt VOMA-I im Überblick.

Beispiel: Setzt ein Manager ein Kommando ab, um ein Mitglied aus einer VO auszuschließen, wird dieses Kommando fallspezifisch möglicherweise in einen entsprechenden Workflow umgesetzt, der unter anderem eine schriftliche Benachrichtigung des Mitgliedes und seines Vorgesetzten beinhalten kann. In der anderen Richtung werden über die Instrumentierung Notifikationen über den VO-Agenten an den Manager gesendet, falls etwa eine Rolle plötzlich verwaist ist (`notifyOrphanedRole()` im Abschnitt 5.5.8).

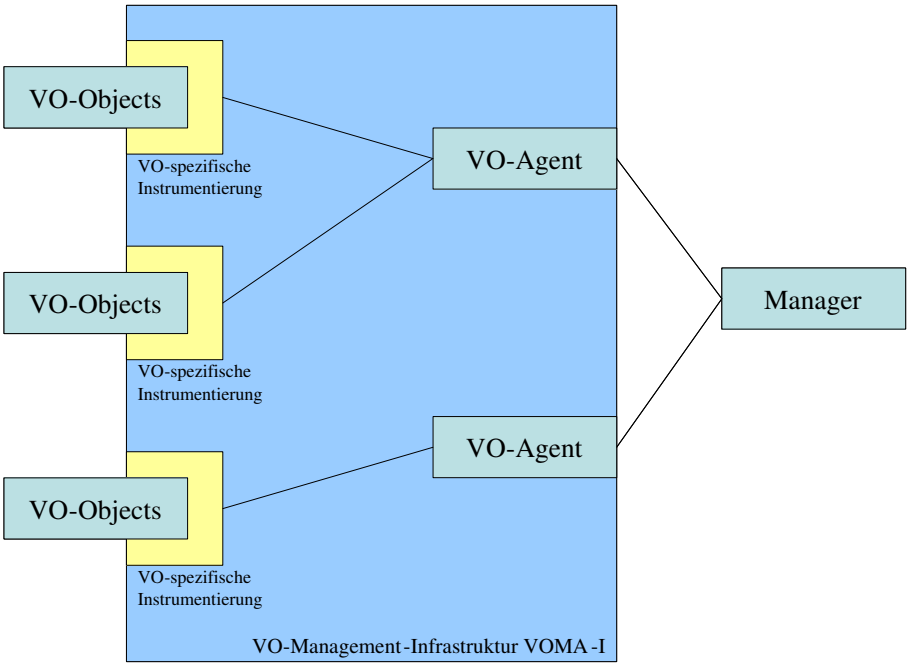


Abbildung 7.2: Generische VO-Management-Infrastruktur im Überblick nach [Vatle, 2005]

Im Kontext dieser Arbeit sind die Schnittstellen zwischen Managern und VO-Agenten bzw. VO-Agenten und Instrumentierungen zwar standardisiert (auf der Basis von XML), Protokollumsetzungen können aber dennoch in dem einen oder anderen Fall nötig sein. Entsprechende Adapter sind aber Stand der Technik.

Der generische Teil eines VO-Agenten besteht aus einem oder mehreren solcher Protokolladapter und einem VO-Agenten-Server, der die vom Manager eingehenden Kommandos an die zur Bedienung zuständigen MOAMs delegiert. Dazu verwendet der Server eine entsprechende „Routing Table“, über die m:n-Beziehungen zwischen Managern und VO-Agenten abgebildet werden [Hegering u. a., 1999]. Rückmeldungen, Antworten und Notifikationen werden von den MOAMs über den VO-Agenten-Server an den Manager weitergeleitet.

Der MO-spezifische Teil des VO-Agenten (die MOAMs) dient den folgenden Zwecken:

Transformation von Schnittstellen. MOAMs kennen die Instrumentierungsschnittstellen. Insofern ist es ihre Aufgabe, die Schnittstellen-Transformation durchzuführen. MOAMs empfangen Kommandos und senden Antworten. Welche konkreten Protokolle und Umsetzungsverfahren ein MOAM nutzt, ist für den Manager transparent. Insofern kann ein VO-Agent über mehrere MOAMs auch mehrere Instrumentierungen bedienen.

Beispiel: Ein Manager will eine neue Rolle in eine VO einbringen und diese besetzen. Dazu setzt er ein entsprechendes Kommando an den die VO betreffenden VO-Agenten ab. Dieser verwendet – transparent für den Manager –

die Schnittstelle zum Rollenmanagement, um die Rolle zu etablieren und die Schnittstelle zum Membermanagement, um die Rolle zu besetzen. Zusätzlich können Backup-Prozeduren angestoßen werden oder Notifikationen an das Accounting Management gesendet werden.

Datenmanagement. MOAMs können mit Intelligenz versehen werden, große Mengen von Daten aus der instrumentierten VO zu verwalten, zu aggregieren oder zur Fehlerbehebung zu analysieren. Ebenso können die MOAMs mit Statistikfunktionalitäten ausgestattet werden.

Notifikationen. MOAMs können Manager asynchron über Ereignisse benachrichtigen, entweder, indem sie Notifikationen weiterleiten, oder indem sie im Rahmen einer Überwachung von Schwellwerten entsprechende Alarmmeldungen proaktiv absetzen.

Abbildung 7.3 zeigt den Aufbau eines VO-Agenten mit den entsprechenden MOAMs, Abbildung 7.3(a) Plattform-unabhängig, Abbildung 7.3(b) Globus-spezifisch nach einer PIM-PSM-Transformation im MDA-Sinn.

Die hier angesprochenen VO-Instrumentierungen werden jeweils fallspezifisch unter Verwendung der im Abschnitt 5.5 definierten Funktionen entwickelt und im Rahmen des VOMA-Deployments zur Verfügung gestellt. Abbildung 7.4 zeigt informell die prinzipielle interne MOAM-Logik. Der Zweck der einzelnen Komponenten sei hier kurz dargestellt. Wir orientieren uns dabei weitgehend an [Vatle, 2005].

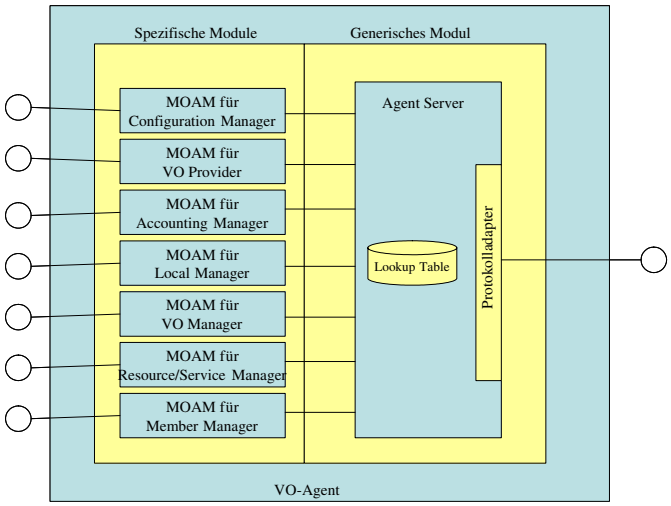
VO-Schnittstelle. Diese Komponente ermöglicht die Kommunikation zwischen dem MOAM und der instrumentierten Entität. Über diese Schnittstelle werden vom MOAM Kommandos abgesetzt und Antworten bzw. Notifikationen empfangen.

MOAM Data Manager. Diese Komponente akquiriert Daten von der instrumentierten Entität, entweder auf Grund einer Anfrage des Managers, durch eigenes „Polling“ oder indem die Entität Daten selbständig übermittelt. Dazu werden die Operationen `getElementData()` für die ersten beiden Fälle und `subscribeElementData()` für den letzten Fall verwendet.

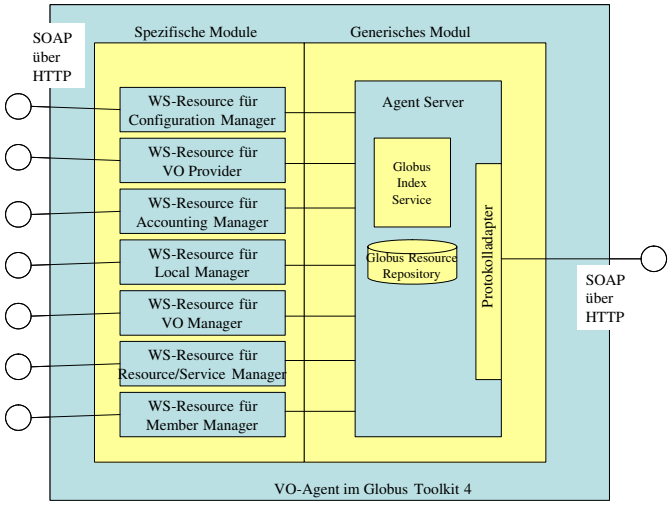
Command Processor. Diese Komponente führt über die Instrumentierungsschnittstelle die eigentlichen Management-Kommandos aus. Dazu wird vom Command Processor das Kommando `executeCommand()` verwendet.

VO-MIB Engine. Diese Komponente verwaltet die MOs des MOAM in der VO-MIB und nutzt dafür die Kommandos `getMO()`, um MOs aus der VO-MIB auszulesen, `getMOAttribute()`, um Attribute von MOs zu lesen, `setMO()`, um MOs in die VO-MIB einzutragen, `deleteMO()`, um MOs in der VO-MIB zu löschen und `setInvalid()`, um MOs nach Managementaktionen als ungültig zu markieren.

MOAM Instance Manager. Der MOAM Instance Manager generiert (*create*) MOs in der VO-MIB, basierend auf dem Klassenmodell des VOMA-Informationsmodells. Er



(a) Plattform-unabhängig



(b) Globus-spezifisch

Abbildung 7.3: Aufbau eines VO-Agenten mit MOAMs (adaptiert von [Vatle, 2005])

verwendet dafür die Operationen `getMO()`, um MOs über die MIB-Engine aus der VO-MIB auszulesen, `getMOAttribute()`, um Attribute über die MIB-Engine von MOs zu lesen, `deleteMO()`, um MOs in der VO-MIB über die MIB-Engine zu löschen und `createMO()`, um einen neuen VO-MIB-Eintrag zu generieren.

MOAM Audit Manager. Diese Komponente sichert die für Audits erforderlichen MO-spezifischen Daten in einem entsprechenden Repository. Er verwendet dafür die Operationen `saveAuditData()` und `retrieveAuditData()`.

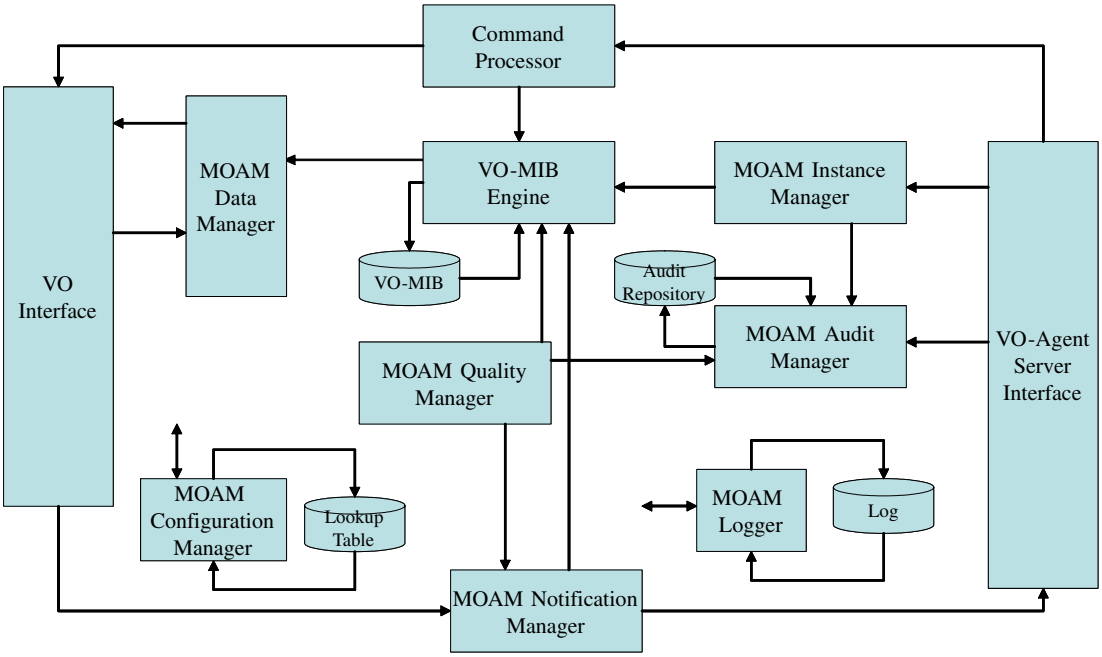


Abbildung 7.4: Bausteine eines MOAM in Anlehnung an [Vatle, 2005]

MOAM Quality Manager. Der Zweck dieser Komponente liegt in der Qualitätsüberwachung der MO im Rahmen einer proaktiven Schwellwert-Kontrolle. Der MOAM Quality Manager bietet keine expliziten Managementoperationen an, stattdessen sendet er Notifikationen.

MOAM Notification Manager. Diese Komponente generiert Notifikationen an die Managementschnittstelle. Sie verwendet dafür die Operation `notify()`.

Schnittstelle zum VO-Agent-Server. Diese Komponente bedient die Schnittstelle zum VO-Agent-Server. Sie verwendet dafür die Operationen `notify()`, `getMOAMConfiguration()`, `setMOAMConfiguration()`, `queryVOMIB()`, `executeCommand()` und `retrieveLog()`.

MOAM Configuration Manager. Diese Komponente enthält die MOAM-spezifische Konfiguration (nicht die VO-Konfiguration!). Die Konfigurationseinträge werden über `getMOAMConfiguration()` und `setMOAMConfiguration()` verwaltet.

MOAM Logger. Diese Komponente speichert die MOAM-spezifischen Log-Einträge und verwendet dafür die Operationen `retrieveLog()` und `addLog()`.

7.3 Konfigurationsmuster

VOMA ist konzipiert, um in unterschiedlichen Grid-Umgebungen eingesetzt zu werden, unter a priori nicht vorhersagbaren Randbedingungen. Einige Beispiele sollen dies verdeutlichen:

- Werden im HEP Community Grid [Gentzsch, 2005] im Rahmen des ATLAS-Projektes weitere Teilprojekte initiiert, bedeutet dies aus der Grid-Sicht unter Umständen die Gründung von Sub-VOs innerhalb der ATLAS-VO. Aus der Perspektive des VO-Managements ist es dann nicht unbedingt erforderlich, für jede Sub-VO einen VO-Agenten zu konfigurieren, wenn der VO-Agent für die ATLAS-VO die Anforderungen erfüllen kann.
- Im IPY-Szenario (siehe Abschnitt 3.1.1) tritt der Fall auf, dass das VO-Management reine Monitoring-Funktionen übernimmt (Wer nutzt welche Ressourcen (= Datensätze) wann?) und nicht aktiv interveniert.
- Im EmerGrid-Szenario (Abschnitt 3.1.1) wiederum tritt der Fall auf, Prioritätenregelungen durchsetzen zu müssen. Der entsprechende VO-Agent des Interventions-Teams (jetzt aufgefasst als VO) wird deshalb primär mit dem Command Processor konfiguriert werden, eine Ausprägung reiner Query-Funktionalitäten ist eher optional.
- Im DEISA-Szenario (Abschnitt 3.1.1) sind umgekehrt vornehmlich Query- und Monitoring-Komponenten von Bedeutung, Interventionsmöglichkeiten des VO-Managements sind eher gering, da DEISA-Ressourcen vom lokalen Management administriert werden.

Es ist daher wünschenswert, im Rahmen des *VOMA Customizings* das Deployment so zu gestalten, das das Informationsmodell und die VO-Agenten nur mit den Funktionalitäten (in Form von MOAMs) ausgestattet werden, die den Anforderungen entsprechenden. Diese werden über Konfigurationsmuster der VO-Agenten beschrieben, die die Auswahl der MOAM-Komponenten fallspezifisch steuern. Die Konfigurationsmuster orientieren sich an den folgenden aus den Anforderungen abgeleiteten Dimensionen (siehe auch Abbildung 7.5 und [Vatle, 2005]):

Deployment-Dimension. In der Deployment-Dimension wird die „logische Distanz“ des VO-Agenten zur VO spezifiziert. Dabei gilt es zwei Fälle zu unterscheiden: Der VO-Agent ist selbst Bestandteil der VO (der Standardfall) oder der VO-Agent ist von der VO entkoppelt und beispielsweise in einer anderen VO beheimatet (wie im vorher beschriebenen Fall der ATLAS-VO mit VO-Hierarchien durch Super- und Sub-VOs, wo der VO-Agent auch die Sub-VOs bedienen kann). Der erste Fall wird hier als endogenes Muster bezeichnet, der zweite als exogenes Muster. Abbildung 7.6 zeigt diese Konstellationen schematisch.

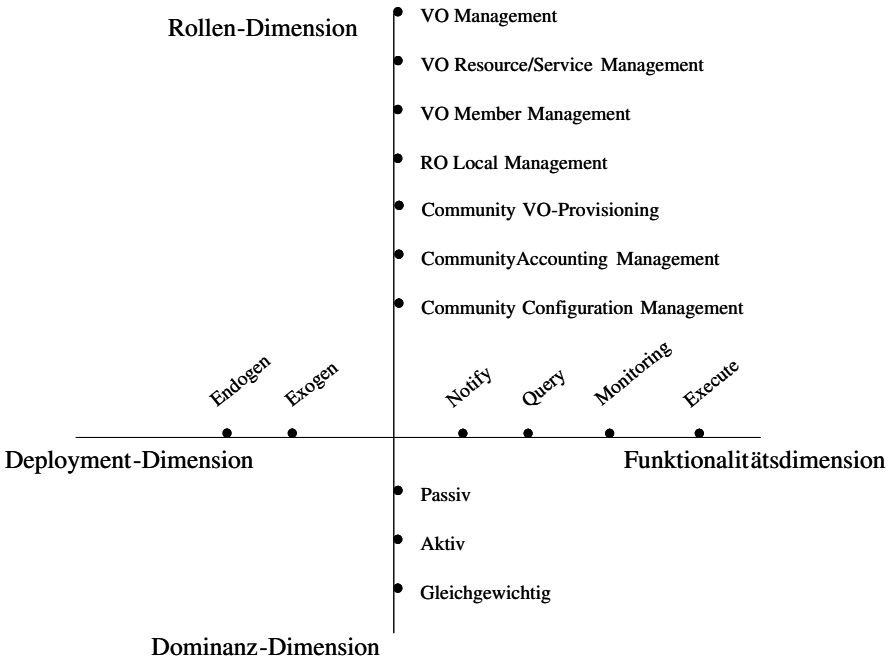


Abbildung 7.5: Dimensionen der Konfigurationsmuster

Funktionalitätsdimension. In der Funktionalitätsdimension wird der Funktionsumfang des VO-Agenten festgelegt, der sich in der Komposition der MOAM-Bausteine niederschlägt. Aus der Analyse der Anforderungen sind die folgenden Muster für VO-Agenten ableitbar:

- Der VO-Agent ist ein reiner Notifikator, der VO-Daten übernimmt und diese an den Manager weiterleitet (Notify-Muster).
- Der VO-Agent ruft VO-Daten auf Anforderung des Managers ab (Query-Muster).
- Der VO-Agent ruft proaktiv VO-Daten ab, führt sie einer Analyse zu und informiert den Manager (Monitoring-Muster).
- Der VO-Agent setzt Kommandos an die VO ab (Execute-Muster).

Dominanz-Dimension. Über die Dominanz-Dimension wird das Verhältnis des VO-Agenten zur „instrumentierten“ Entität dargestellt. In diesem Verhältnis kann der VO-Agent eine *passive* Rolle spielen, indem er nur VO-Daten empfängt, eine *aktive* Rolle, indem er nur Kommandos absetzt, oder eine *gleichgewichtige* Rolle, die beide Aspekte vereinigt.

Rollen-Dimension. Über die Rollen-Dimension wird die Existenz der MOAMs gesteuert. Die Rollen orientieren sich an den im Organisationsmodell (Abschnitt 5.3) identifizierten Rollen.

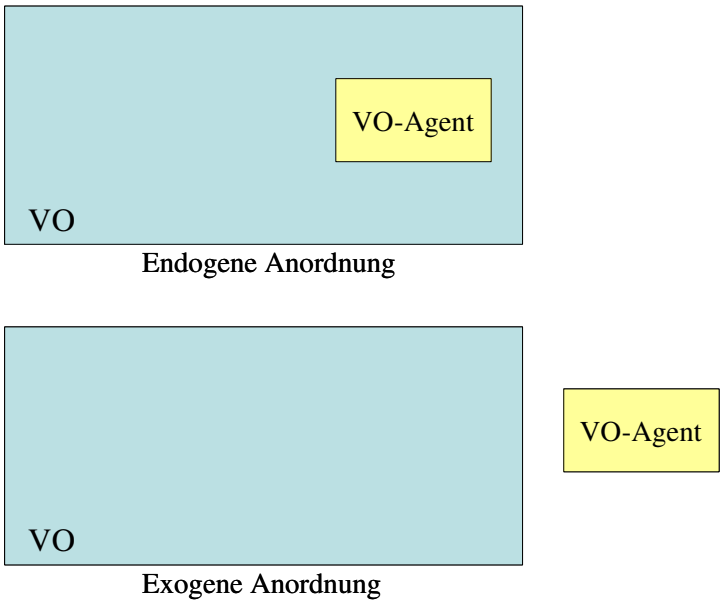


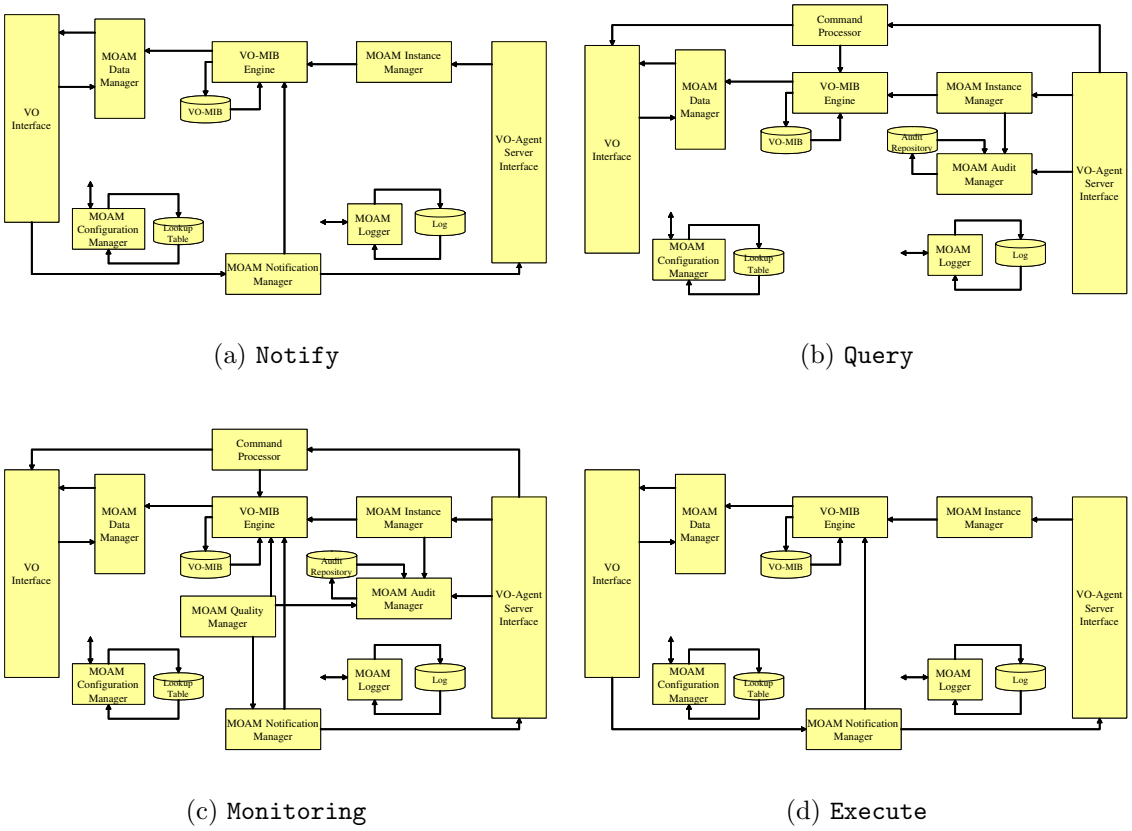
Abbildung 7.6: Konfigurationsmuster der Deployment-Dimension

Obwohl alle angegebenen Konfigurationsmuster isoliert anwendbar sind, liegt der Wert in deren Kombinierbarkeit und der fallspezifischen Komposition im Rahmen des Deployments. Die Menge der möglichen Kombinationen (120) wird erheblich eingeschränkt durch folgenden Überlegungen:

- Die Rollen-Dimension steuert nicht die interne Struktur eines MOAM, sondern die des VO-Agenten, indem MOAMs hinzugefügt oder entfernt werden.
- Die Deployment-Dimension besitzt keinen Einfluss auf die eigentliche Konfiguration des VO-Agenten, vielmehr steuert sie indirekt die physische Platzierung des VO-Agenten.
- Das Notify-Muster tritt nur in Kombination mit dem passiven Muster der Dominanz-Dimension auf, da der VO-Agent keine intervenierenden Kommandos absetzt..
- Das Query-Muster tritt nur in Kombination mit dem aktiven Muster der Dominanz-Dimension auf, da der VO-Agent proaktiv Kommandos an die Instrumentierung absetzt.
- Das Monitoring-Muster tritt aus den gleichen Gründen ebenfalls nur in Kombination mit dem aktiven Muster der Dominanz-Dimension auf.
- Auch das Execute-Muster tritt offensichtlich nur in Kombination mit dem aktiven Muster der Dominanz-Dimension auf.

Die damit verbleibenden vier MOAM-Konfigurationsmöglichkeiten können wie in den Abbildungen 7.7(a) bis 7.7(d) als Spezialisierungen der vollständigen Konfiguration aus

Abbildung 7.4 umgesetzt werden. Eine detailliertere Darstellung – allerdings nicht Grid-bezogen – ist in [Vatle, 2005] zu finden.



(a) `Notify` (b) `Query`

(c) `Monitoring` (d) `Execute`

Abbildung 7.7: MOAM Konfigurationmuster

7.4 Deployment

Mit diesem Hintergrund kann nun eine generelle Nutzungs- und Deploymentmethodik angegeben werden. In ihr wird aufgezeigt, welche Schritte notwendig sind, um VOMA in bestehende (und zukünftige) Grid-Umgebungen zu integrieren. Das Ziel der Methodik ist es, einen Leitfaden für den Aufbau, den Betrieb und die Beendigung Virtueller Organisationen unter Verwendung der in dieser Arbeit entwickelten Konzepte bereitzustellen.

In Kapitel 5 wurde VOMA mit seinen Teilmodellen als Basis für einen „VO-Management-Baukasten“ vorgestellt, der zur Entwicklung von VO-Managementanwendungen in Grids dient. Die Komponenten dieses Baukastens wurden dabei Plattform-unabhängig konzipiert. Im Kapitel 6 wurde am Beispiel des Web Services Distributed Management (WSDM) deshalb kurz gezeigt, wie die VOMA-Komponenten Plattform-spezifisch transformiert werden

können. In diesem Kapitel wurde dann diskutiert, wie das komplexe Gebilde VOMA durch vordefinierte Konfigurationsmuster fallspezifisch angepasst und damit in bestehende Umgebungen integriert werden kann. Der Rest dieses Kapitels widmet sich nun noch der Frage, wie diese Konzepte in die Praxis umgesetzt werden können. Dazu wird gemäß Abbildung 7.8 vorgegangen.

7.4.1 Überblick

Für die folgenden Betrachtungen wird die VOMA-Einführung als Projekt auf der Community-Ebene betrachtet, von der auch die Initiative zur Gründung von VOs primär ausgeht (also etwa die MediGrid-Community des D-Grid). Dies ist insofern keine Beschränkung, als andere Konstellationen sich immer darauf zurückführen lassen. Abbildung 7.8 zeigt diese Konstellation im Überblick.

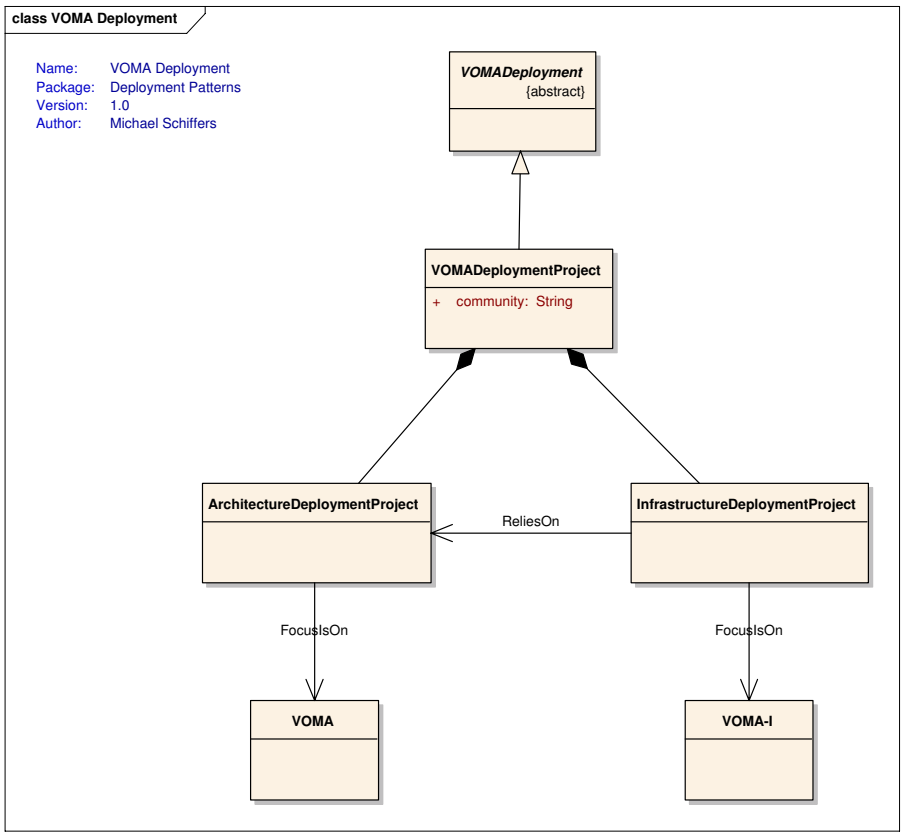


Abbildung 7.8: Deployment-Projekt im Überblick

Das Deployment-Projekt `VOMADeploymentProject` erfolgt in den Teilprojekten `ArchitectureDeploymentProject` und `InfrastructureDeploymentProject`: Im ersten Teil (`ArchitectureDeploymentProject`) werden im Rahmen des Gesamtprojektes

`VOMADeploymentProject` VOMA und seine Teilmodelle definiert. Das Ziel dieses Teilprojektes ist die fallspezifische Bereitstellung der Architektur als Rahmenwerk. Im zweiten Teilprojekt (`InfrastructureDeploymentProject`) wird die VO-Managementinfrastruktur VOMA-I vorbereitet. Das Ziel dieses Teilprojektes ist die Bereitstellung einer VO-Management-Infrastruktur, über die die Ergebnisse des ersten Teilprojektes eingesetzt werden können. Beide Teilprojekte können prinzipiell parallel durchgeführt werden.

7.4.2 Deployment der VO-Managementarchitektur

Das Teilprojekt zur Bereitstellung der VO-Managementarchitektur (`ArchitectureDeploymentProject`) besteht aus mehreren Phasen, die nun den Nutzen des verwendeten MDA-Ansatzes erkennen lassen:

Phase 1: Analyse. In der Analysephase spezifiziert die Community die Anforderungen an das VO- und VO-Management-Konzept, indem Community-spezifische Eckdaten festgelegt werden. Diese Phase mündet in einem Anforderungskatalog für die VO-Managementarchitektur. Der Katalog enthält die funktionalen und nicht-funktionalen Anforderungen, die vertraglichen Randbedingungen (z.B. Policies), die einzurichtenden Rollen und sonstige im VOMA-Informationsmodell vorgesehene Parameter. Um diesen Schritt durchführen zu können, müssen sowohl die charakteristischen Merkmale von VOs und von VO-Managementprozessen in der Community analysiert werden, als auch die technologischen Grundlagen adressiert werden (z.B. Welche Middleware-Systeme müssen berücksichtigt werden, welche Kommunikationsinfrastukturen sind erforderlich?). Damit werden die VOMA-Teilmodelle instanziiert (siehe Abbildung 7.9).

Als Kriterien für diese Analyse dienen die möglichen Zwecke von VOs, die vorhandenen Partner-Organisationen, die einzuladenden Teilnehmer, die erforderlichen Ressourcen und Dienste und generell Parameter, wie sie in Abbildung 1.2 (Seite 7) aufgeführt sind. Zusätzlich können auch historische Daten über frühere VO-Lebenszyklen genutzt werden. Die Funktionsbereiche des VOMA-Funktionsmodells werden schließlich als Kriterium verwendet, um einerseits die Funktionalität des VO-Managements zu gruppieren und um andererseits die im parallel verlaufenden zweiten Teilprojekt (`InfrastructureDeploymentProject`) zu spezifizierende VO-Management-Infrastruktur VOMA-I zu konfigurieren.

Auf dieser Basis kann in der Community entschieden werden, welche Modell-Entitäten zu berücksichtigen sind, welche Ebenen für das VO-Management wie relevant werden, welche Rollen zu definieren sind und wie die Interaktionskanäle zwischen den Rollen ausgeprägt sein müssen. Die Anforderungen werden in der Form eines entsprechenden Templates zur Verfügung gestellt, das in Teilen als Default-VO im Informationsmodell durch die Klasse `VOConfiguration` repräsentiert wird. Die Instanziierung der Modelle kann ein relativ einfacher Prozess sein (zum Beispiel dann, wenn VOMA schon in

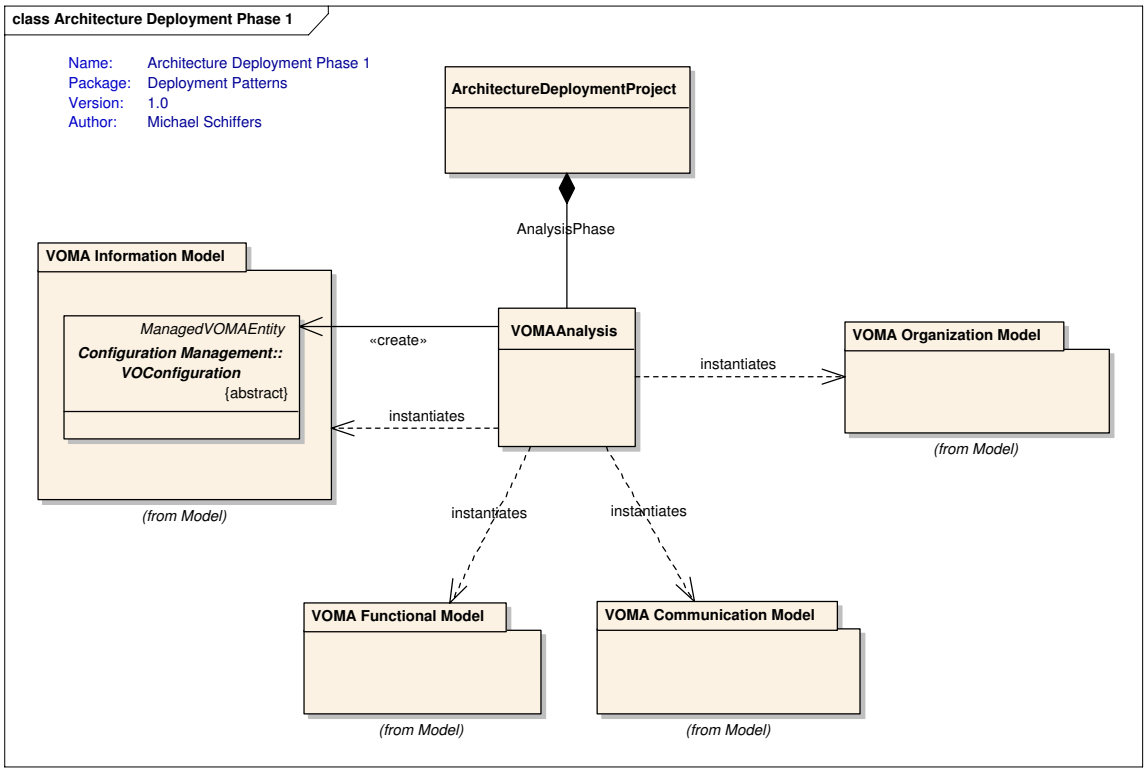


Abbildung 7.9: Phase 1 des Architektur-Deployments

großen Teilen vorliegt), kann aber auch sehr komplex werden. Eine generelle Empfehlung kann deshalb hier nicht gegeben werden.

Das Ergebnis dieses Schrittes ist ein instanziiertes Plattform-unabhängiges VOMA-Modell (instanziiertes VOMA-PIM) mit allen Teilmodellen.

Phase 2: Transformation. In der Transformationsphase wird das zuvor instanziierte VOMA-PIM in ein instanziiertes VOMA-PSM transformiert. Dazu müssen – ein Vorteil des MDA-Ansatzes – nur noch die Plattform-spezifischen Transformationsregeln definiert und angewendet werden. Eine komplette Modellerstellung ist nicht mehr nötig. Dennoch ist diese Phase alles andere als trivial, verlangt sie doch intime Kenntnisse des Zielsystems. Wiederverwendbare Transformations-Skripte können hier allerdings wertvolle Dienste leisten. Abbildung 7.10 zeigt die Transformationsphase im Überblick.

Das Ergebnis ist ein Plattform-spezifisches instanziiertes Modell, das möglicherweise schon direkt in Code umgesetzt werden kann. Falls nicht, wird die Phase 2 wiederholt angewendet.

Phase 3: Customizing. In dieser Phase werden die beiden Teilprojekte `VOMADeploymentProject` und `InfrastructureDeploymentProject` wieder synchro-

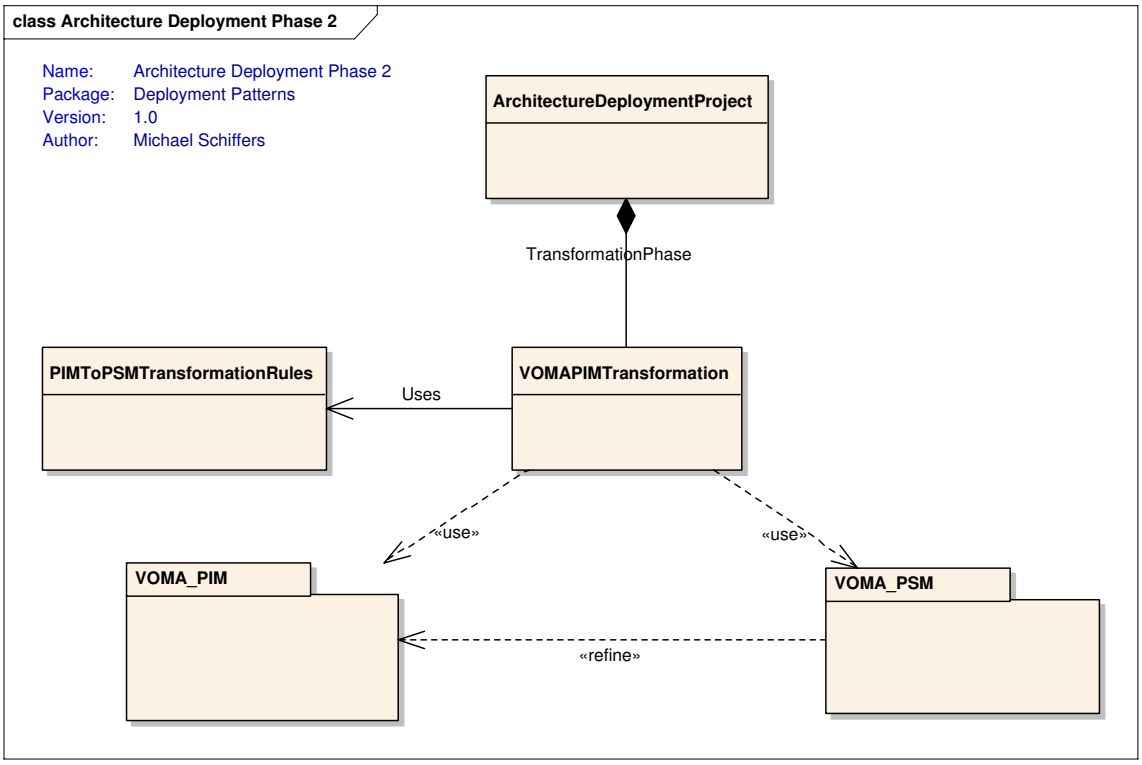


Abbildung 7.10: Phase 2 des Architektur-Deployments

nisiert, indem nun die Instrumentierung der zu managenden Entitäten durchgeführt wird und die beiden Modelle verbunden werden. Damit wird sichergestellt, dass VO-Agenten auf durch *managed objects* repräsentierte Entitäten zugreifen können. Das Ergebnis ist eine auf die speziellen Bedürfnisse der Community zugeschnittene VO-Managementarchitektur, die einsatzbereit ist, nachdem sie auf die verschiedenen Ebenen (Community, RO, VO) „installiert“ wurde. Diese Phase beinhaltet auch die Bereitstellung der erforderlichen Werkzeuge und Technologien (z.B. den VO Membership Service VOMS [Alfieri u. a., 2003] oder Groupware-Lösungen [Borghoff u. Schlichter, 2000]). Abbildung 7.11 zeigt diese Phase im Überblick.

Das Deployment kann im Detail allerdings aufwändig sein, muss VOMA doch in bestehende Managementarchitekturen im Sinne [Hegering u. a., 1999] integriert werden. Aus der Deployment-Sicht sind nämlich reale Organisationen nicht nur auf der RO-Ebene zu betrachten, sondern auch auf der Community-Ebene (die RO ist – zumindest in Teilen – Mitglied der Community) und der VO-Ebene (eine VO „besitzt“ keine Ressourcen, diese „gehören“ der RO). Eine RO „hosted“ damit alle Ebenen. Damit ergeben sich aus der Sicht des Deployments für eine Integration von VOMA mit (in der RO) bereits bestehenden oder geplanten Managementarchitekturen die folgenden Fälle:

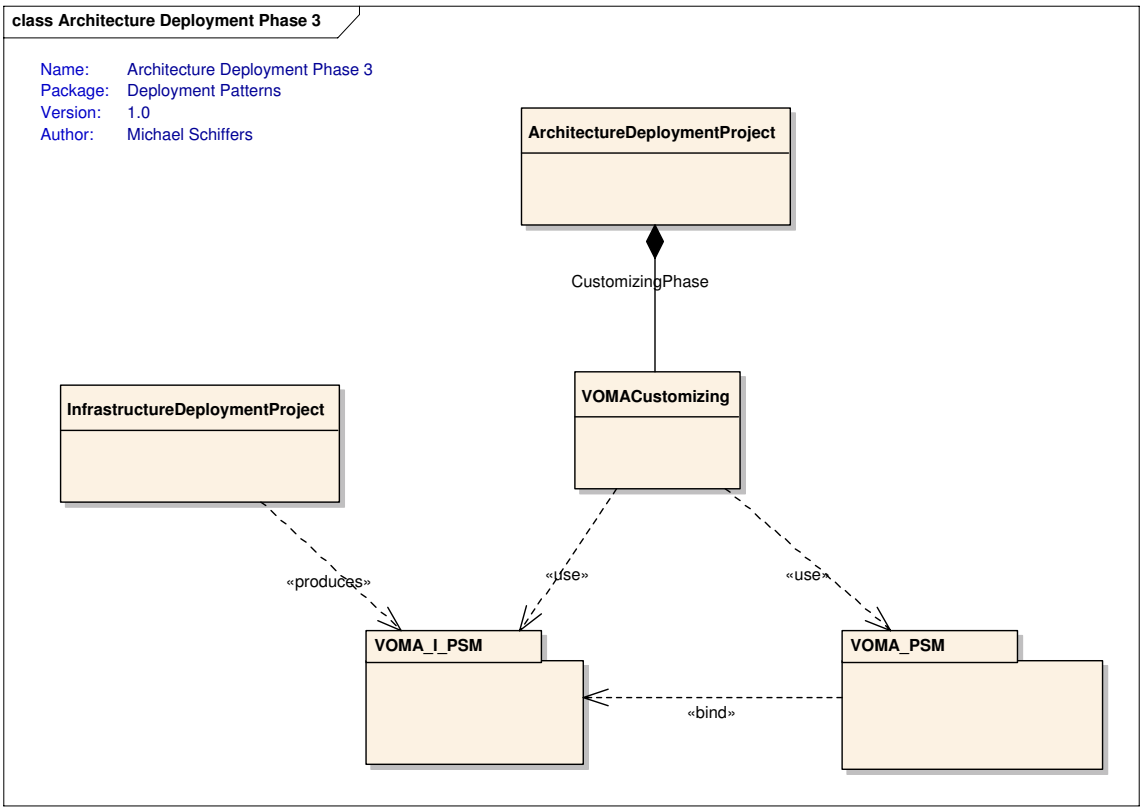


Abbildung 7.11: Phase 3 des Architektur-Deployments

Fall 1: Eine Integration ist nicht erforderlich. Dies ist der einfachste Fall, VOMA wird sozusagen „stand-alone“ betrieben. Ein direkter Übergang in die Betriebsphase ist damit möglich.

Fall 2: Eine lose gekoppelte Integration wird angestrebt. In diesem Fall werden die Rollen unterschiedlicher Managementarchitekturen lose gekoppelt in dem Sinn, dass VO-Agenten und VO-Manager mit den jeweils korrespondierenden Rollen der RO-Managementsysteme kommunizieren können. Dazu müssen allerdings die entsprechenden Rollen durch *Wrapper*-Komponenten erweitert werden, die eine Konvertierung zwischen den beiden Managementarchitekturen ermöglicht.

Fall 3: Die Integration erfolgt über Managementgateways. Im diesem Fall kommunizieren zwei Rollen unterschiedlicher Managementarchitekturen mittels eines Managementgateways, der einen transparenten Zugriff auf die jeweils andere Architektur bereitstellt (d.h. eine Rolle kann nicht erkennen, dass die andere Rolle auf einer unterschiedlichen Architektur basiert). Gateways haben den Vorteil, dass die kommunizierenden Partner nicht modifiziert werden müssen [Hegering u. a., 1999; Keller, 1998].

Ähnliche Überlegungen führen im Rahmen des Deployments auch zur Definition von

Übergängen zwischen VOMA und beliebigen Architekturen, Protokollen oder Technologien. Für VOMA sind dabei insbesondere Übergänge zu Kommunikationsprotokollen relevant (obwohl diese Übergänge in Grids momentan eine eher untergeordnete Rolle spielen), Anbindungen an Datenbank-Managementsysteme und generell Anbindungen an proprietäre Systeme (z.B. Trouble Ticket Systeme).

Die Customizing-Phase enthält damit alle Aktivitäten, die der Vorbereitung der Betriebsphase der Architektur dienen. Neben der Realisierung der schon angesprochenen spezifischen Modelle und der Integrationskonzepte erfordert diese Phase zusätzlich noch die Implementierung einer auf das Management Virtueller Organisationen zugeschnittenen (Nutzer-) Schnittstelle. Wie diese ausgeprägt wird und welche Funktionalitäten diese umfassen soll, ist nicht Gegenstand dieser Arbeit. Ein Beispiel einer Schnittstelle zur dynamischen Anzeige der D-Grid-Ressourcenanbieter ist unter `http://webmds.lrz-muenchen.de:8080/webmds/xslfiles/csm/` zu finden.

Phase 4: Betrieb. In dieser Phase kann VOMA von den in der Customizing-Phase entwickelten VO-Managementanwendungen genutzt werden.

7.4.3 Deployment der VO-Management-Infrastruktur

Das Ziel des Teilprojektes `InfrastructureDeploymentProject` (siehe Abbildung 7.8) besteht in der Bereitstellung einer an die speziellen Verhältnisse des Einsatzes angepassten VO-Management-Infrastruktur (VOMA-I). Auch dieses Teilprojekt wird in Phasen durchgeführt, die wieder den MDA-Ansatz ausnutzen. `InfrastructureDeploymentProject` nutzt die in diesem Kapitel identifizierten Konfigurationsmuster, um die VO-Agenten und deren MO-spezifischen Agentenmodule (MOAM) fallspezifisch zu konfigurieren.

Phase 1: Analyse. Basierend auf der Analysephase des ersten Teilprojektes werden in dieser Phase die Anforderungen an die VO-Management-Infrastruktur VOMA-I festgelegt. Dieser Schritt mündet in einem Anforderungskatalog für die Infrastruktur, der die funktionalen und nicht-funktionalen Ausprägungen der VO-Managementanwendungen, der VO-Agenten und der MOAMs festlegt.

Um diesen Schritt durchführen zu können, müssen die charakteristischen Merkmale der VO-Managementprozesse in der Community festgelegt und auf die „Geschäftsprozesse“ abgebildet werden.

Als Kriterien für die Analyse dienen die erforderlichen VO-Managementprozesse, die Einbindung vorhandener Partner-Organisationen und die Integration deren Managementsysteme. Auf dieser Basis kann in der Community entschieden werden, welche Modell-Entitäten zu berücksichtigen sind, welche Rollen für das VO-Management zu definieren sind und welche Instrumentierungen für die einzelnen Entitäten zu berücksichtigen sind. Dies ist die Schnittstelle zum Teilprojekt `ArchitectureDeploymentProject`. Abhängig von den Anforderungen erfolgt in dieser Phase auch die Zusammenstellung der Konfigurationsmuster.

Das Ergebnis dieser Phase ist ein instanziiertes, an die Bedürfnisse der Community angepasstes Plattform-unabhängiges VOMA-I-Modell (instanziiertes VOMA-I-PIM) in einer den erforderlichen Konfigurationsmustern entsprechenden Ausprägung der Abbildung 7.7.

Phase 2: Transformation. In der Transformationsphase wird das zuvor instanziierte VOMA-I-PIM auf Basis der durchgeführten Analysen in ein instanziiertes VOMA-I-PSM transformiert. Dazu müssen auch hier nur die Plattform-spezifischen Transformationsregeln definiert werden. Eine komplette Modellerstellung ist nicht mehr erforderlich. Das Ergebnis dieser Phase ist ein Plattform-spezifisches instanziiertes VOMA-I-Modell, das möglicherweise schon direkt in Code umgesetzt werden kann. Falls nicht, wird die Phase 2 wiederholt angewendet.

Phase 3: Customizing. In dieser Phase werden die beiden Teilprojekte wieder synchronisiert, der weitere Fortschritt wird in der Customizing-Phase des Abschnittes 7.4.2 dargestellt.

7.4.4 Anwendungsbeispiele

Nachdem durch das Deployment der VO-Managementarchitektur und der VO-Management-Infrastruktur alle Voraussetzungen für eine Nutzung gegeben sind, besteht die nächste Aufgabe darin, die erforderlichen Managementanwendungen zu entwickeln. Nach welchem Ansatz dies geschieht und welchen Projektkonstellationen dafür vorzusehen sind, kann allerdings nicht Gegenstand dieser Arbeit sein, stattdessen wird auf den Stand der Technik im Software Engineering verwiesen. Im Rahmen dieser Arbeit soll lediglich in den folgenden Abschnitten an Beispielen dargestellt werden, wie der durch VOMA und VOMA-I zur Verfügung stehende „Baukasten“ verwendet werden kann.

Deployment im D-Grid

Die vorher vorgestellte Deployment-Methodik wird im D-Grid über das Rahmenkonzept zum Management Virtueller Organisationen umgesetzt [Milke u. a., 2006b]. Wegen der Heterogenität und Historie der einzelnen Communities gestaltet sich das Deployment als außerordentlich komplex. Ein vollständiger Überblick würde den Rahmen der Arbeit erheblich sprengen. An zwei einfachen Beispielen sollen aber dennoch Spezialisierung und Instanziierung des Informationsmodells erläutert werden.

Die Ziele des VO-Rahmenkonzeptes im D-Grid sind erstens die Definition typischer Community-spezifischer Policies zum VO-Management und zweitens die Spezifikation generischer VO-Management-Konzepte. Damit soll den Gründern einer neuen VO eine Hilfestellung bei der Auswahl von Systemen, Werkzeugen, Schemata und Prozessen für das Management Virtueller Organisationen in bestehenden D-Grid-Communities geliefert werden und neuen Communities ein „How-To“ zum VO-Management in die Hand gegeben

werden. Der Fokus des Konzeptes liegt damit sowohl auf der Bereitstellung der Architektur (VOMA) als auch auf der Bereitstellung der Infrastruktur (VOMA-I) im Kontext bestehender Grid- und Nicht-Grid-Umgebungen.

Im D-Grid sind VOs immer konkret einer Community oder dem Kern-D-Grid zugeordnet. Der Zweck einer D-Grid-VO ist die Steuerung des Zugriffs auf und die Nutzung von Ressourcen einer Community oder des Kern-D-Grids durch die D-Grid-Benutzer. Die Ressourcen können von einem Community-fremden Resource Provider angeboten werden, oder es sind Ressourcen, die von der jeweiligen Community selbst verwaltet werden, oder es sind Ressourcen des Kern-D-Grids. Der *Community-Sprecher* ist für die Verhandlungen mit den Einrichtungen, die Ressourcen anbieten und zur Verfügung stellen, verantwortlich. Er entscheidet über die Gründung einer VO und die Zuordnung von Ressourcen zu einer VO. Für die organisatorische Betreuung der Lebenszyklen von VOs bestimmen die Communities und das Kern-D-Grid eine verantwortliche Person in der Rolle des *VO-Administrators*. Im Rahmen der Analyse-Phase des Teilprojektes `ArchitectureDeploymentProject` werden diese Rollen über Vererbungsmechanismen etabliert. Abbildung 7.12 zeigt dies beispielhaft.

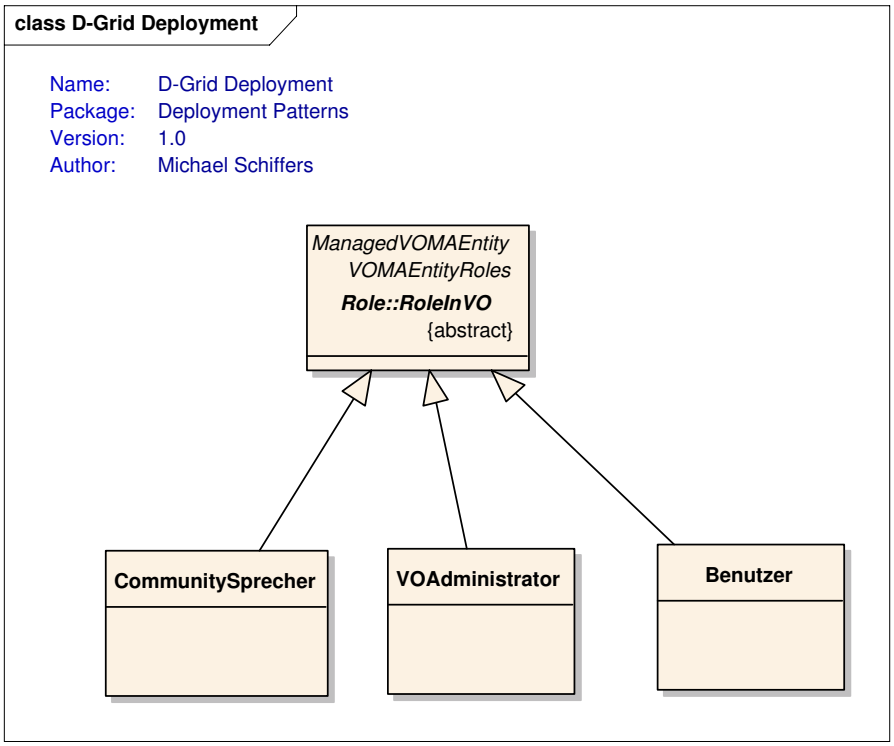


Abbildung 7.12: Beispiel zur Spezialisierung im Rahmen der Analyse-Phase

Im D-Grid sind die Sprecher der Communities in ihren Rollen als VO-Manager und/oder VO-Administrator nicht nur für die SLAs mit den Resource Providern verantwortlich, sie entscheiden auch über die Gründung einer neuen VO, über Änderungen bezüglich der Ressourcen-Zuordnung zu einer VO, über die Beendigung einer VO und über die Richt-

linien, unter denen eine VO betrieben wird. Der Benutzer beantragt Mitgliedschaft in einer bestehenden VO oder die Gründung einer neuen VO über von den Communities und dem Kern-D-Grid zur Verfügung gestellte Schnittstellen. Die Ausbildung dieser Schnittstellen ist Community-spezifisch, aber Web-basiert. Abbildung 7.13 zeigt dies am Beispiel der AstroGrid-Comunity im D-Grid, aus der sowohl die im Rahmen der Customizing-Phase notwendige Spezialisierung der Klasse `VOParticipant` ersichtlich wird (`visitor`, `candidate`, `applicant`, `member`), als auch die Integration des VO Management Registration Service (VOMRS) [VOX Project Team, 2004].

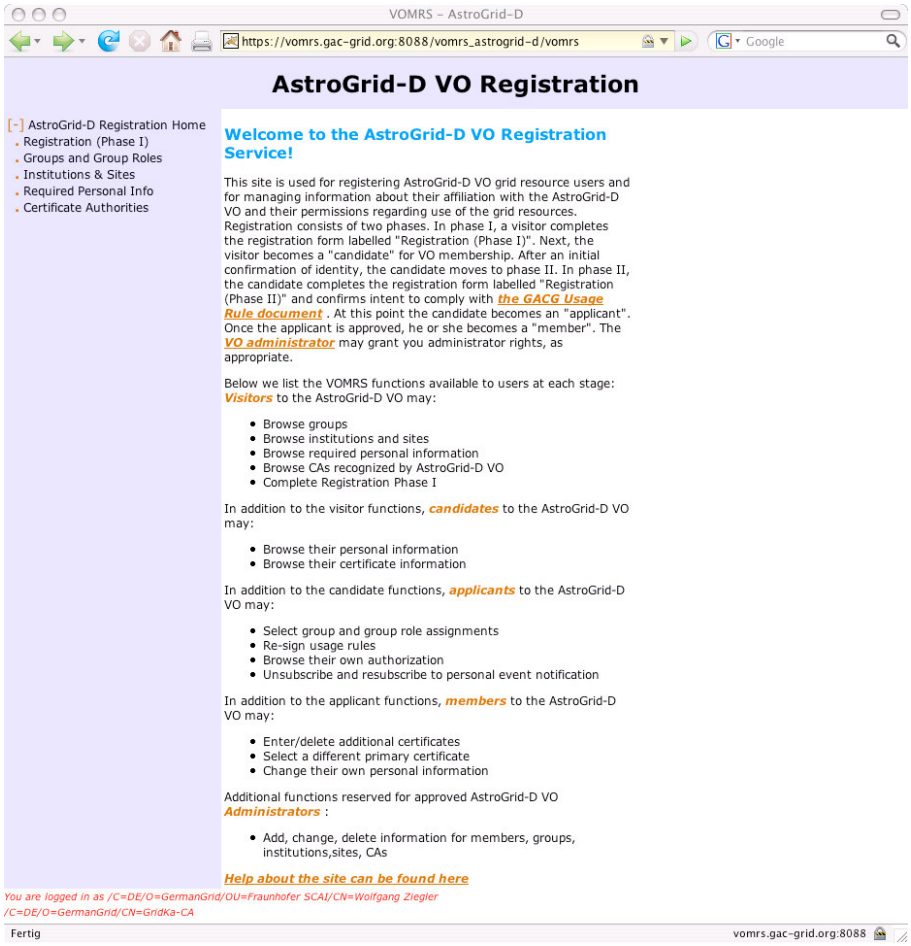


Abbildung 7.13: Beantragen von Mitgliedschaft im AstroGrid-D

Einfaches Hinzufügen von Mitgliedern

`applicants` bewerben sich um die Aufnahme als VO-Mitglieder (`participants`). Im folgenden Beispiel wird die Sequenz vom Manager, der den Aufnahmeantrag beispielsweise über ein Portal erhält, über den VO-Agenten, zur VO-Schnittstelle und schließlich zum Member Management der `VirtualOrganizationDomain` gezeigt (siehe Abbildung 7.14).

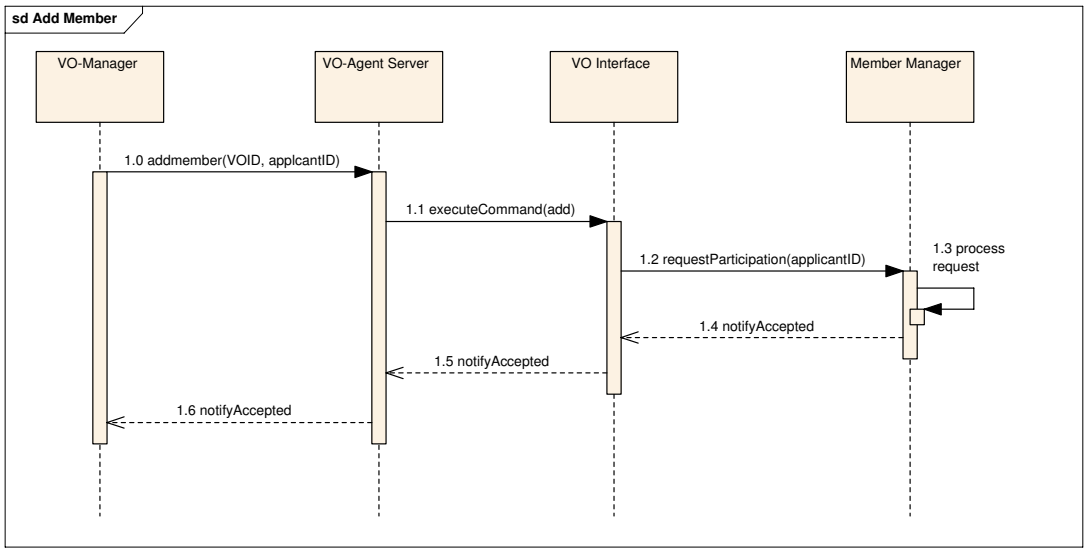


Abbildung 7.14: Einfaches Hinzufügen von Mitgliedern

Der VO-Manager sendet an den VO-Agenten einen entsprechenden Request (hier `addmember()`), mit dem dem Member Management mitgeteilt wird, den über den Parameter `applicantID` identifizierten `applicant` in die über den Parameter `VOID` identifizierte VO aufzunehmen. Der VO-Agent adressiert den für Member-MOs zuständigen MOAM und sendet das Kommando `executeCommand()` mit den entsprechenden Parametern an das VO-Interface, der eigentlichen Schnittstelle zum instrumentierten Member Management. Das Member Management der `VirtualOrganizationDomain` übersetzt die Anfrage in das Kommando `requestParticipation()`. Das Member Management führt nun alle erforderlichen Maßnahmen durch, um – in diesem Fall – die Mitgliedschaft zuzulassen. Entsprechende Notifikationen fließen zum Manager zurück. Wie das Member Management den Mitgliedschafts-Request dann behandelt, bleibt transparent, insbesondere, welche Werkzeuge dazu verwendet werden.

Monitoring von Rollenbelegungen

In Abbildung 7.15 ist dargestellt, wie mit den definierten Funktionen ein Rollen-Monitoring als Subskription realisiert werden kann. Der VO-Manager sendet an den VO-Agenten den Request `getRoles()`, der vom VO-Agent-Server in das Kommando `subscribeElementData` umgesetzt wird.

Das VO-Interface veranlasst dann das Member Management, die angeforderte Funktion auszuführen. Der Managementprozess wird über die erfolgreiche Subskription informiert (`notifySubscriptionAccepted`) und die angeforderten Daten werden asynchron bereitgestellt (`notifyList`).

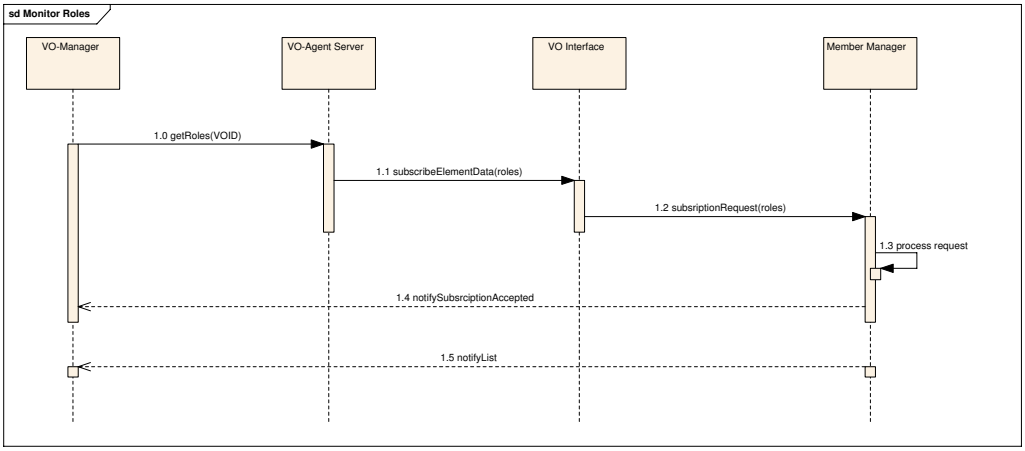


Abbildung 7.15: Monitoring von Rollenbelegungen über Subskriptionen

Beenden einer VO

Der Workflow zum Beenden einer VO ist etwasaufwändiger und in Abbildung 7.16 dargestellt. Unter der Annahme, dass der Workflow primär vom VO-Management realisiert wird, sendet dieser zunächst einen Request an den VO-Agenten, Audit-relevante Daten zur VO zu sichern (`saveAuditData`), um anschließend die betroffenen MOs zu „deaktivieren“ (`setInvalid`) und die VO zu stoppen. Dazu werden die `Participants` von ihren Rollen getrennt (`unAssignRole`), die Participants aus der VO „entlassem“ (`withdrawParticipation`) und schließlich die VO gestoppt (`stopVO`).

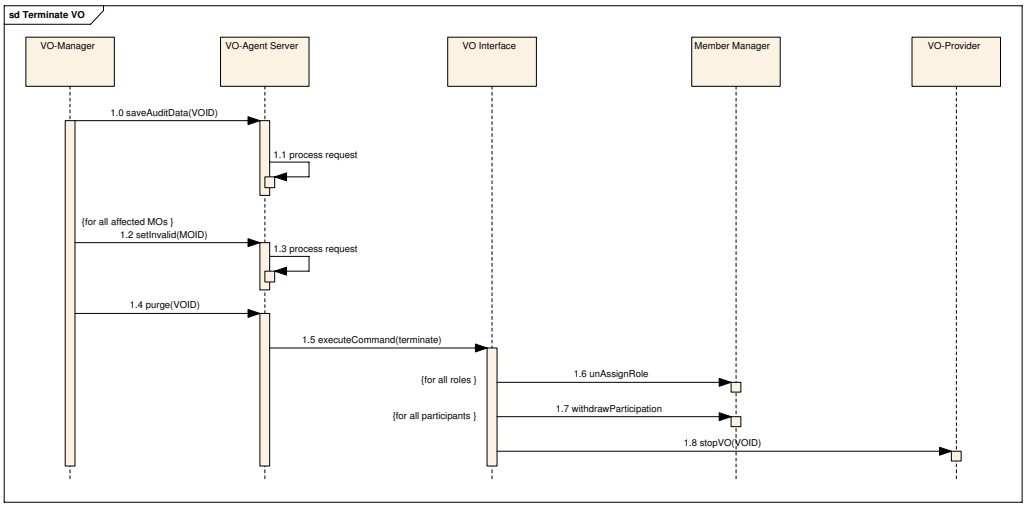


Abbildung 7.16: Beenden einer VO

7.5 Zusammenfassung

In diesem Kapitel wurde die vorher spezifizierte VO-Managementarchitektur VOMA in den Kontext eines Manager/Agenten-Paradigmas gestellt. Damit wird es möglich, einfache Management-Kommandos zum Überwachen und Kontrollieren von VOs und den damit zusammenhängenden *managed objects* (Rollen, Mitglieder, u.ä.) abzusetzen, ohne auf die Interna dieser MOs zu zielen. Das dafür eingeführte Konzept einer VO-Management-Infrastruktur (VOMA-I) erlaubt die Definition von VO-Agenten und MO-spezifischen Agentenmodulen (MOAM), die die Schnittstelle zu den MOs bedienen. Um die Komplexität der Agenten zu reduzieren, wurden aus den Anforderungen Konfigurationsmuster der VO-Agenten abgeleitet, die fallspezifisch eingesetzt werden können. Vor diesem Hintergrund wurde schließlich gezeigt, wie VOMA und VOMA-I in realen Umgebungen eingesetzt werden können. An einigen Beispielen wurde dies demonstriert.

Nach der Plattform-unabhängigen Spezifikation der Architektur im Kapitel 5 und deren Transformierbarkeit in Plattform-spezifische Umgebungen sowie der Darstellung von Deployment-Mechanismen kann zudem festgestellt werden, dass die im Kapitel 3 identifizierten Anforderungen an eine VO-Managementarchitektur für Grids mit den hier vorgestellten Konzepten komplett erfüllt werden. Sämtliche der dort aufgeführten Aktoren, Anwendungsfälle und nicht-funktionalen Anforderungen sind durch die hier auf der Basis eines MDA-Ansatzes entwickelten flexiblen Architektur und der damit verbundenen Infrastruktur umsetzbar. Zusätzlich wurden Wege aufgezeigt, wie beide Konzepte (VOMA und VOMA-I) erweitert werden können, um neue Anforderungen zu berücksichtigen bzw. neue Managementanwendungen zu unterstützen.

Kapitel 8

Zusammenfassung und Ausblick

Inhalt des Kapitels

Seitdem ab der Mitte der 1990er Jahre Organisationen mit dem Ziel kooperieren, Ressourcen einer gemeinschaftlichen Nutzung zur koordinierten Problemlösung in dynamischen multi-institutionellen Virtuellen Organisationen zuzuführen, hat das dadurch induzierte Grid-Paradigma erheblich an Bedeutung gewonnen. Nicht nur in der Wissenschaft, sondern inzwischen auch in der Industrie. Das Konzept der Virtuellen Organisation (VO) und deren Lebenszyklus sind dabei von zentraler Bedeutung. Sie implizieren aber auch neue Fragestellungen, insbesondere im Management. Fragen nach einer IT-gestützten Formation, nach einfachen Betriebsmechanismen und der gezielten Auflösbarkeit Virtueller Organisationen in hochdynamischen Umgebungen können nicht mehr „auf dem kleinen Dienstweg“ geregelt werden. Vielmehr ist hier eine adäquate Managementarchitektur gefordert und in vielen Anwendungsbereichen inzwischen sogar kritisch.

Trotz der drängenden Notwendigkeit eines auch gerade Virtuelle Organisationen als *managed objects* umfassenden, integrierten Grid-Management-Ansatzes im Sinne [Hegering u. a., 1999], sind die dazu erforderlichen Architekturen, Plattformen, Betriebskonzepte und Maßnahmen noch weitgehend ungeklärt oder liegen bestenfalls konzeptionell in Teilbereichen vor. Welche konkreten Mechanismen jedoch für das Management dynamischer Virtueller Organisationen notwendig sind und welche Routineaufgaben wie und unter welchen Randbedingungen automatisierbar sind, stellen nach wie vor offene Fragen dar. Dieser Themenkomplex wurde in der vorliegenden Arbeit erstmals systematisch untersucht.

8.1 Zusammenfassung der erzielten Ergebnisse

Das gegenwärtige frühe Stadium der Grid-Diskussionen führt dazu, dass Begriffe wie „Virtuelle Organisation“ und „VO-Management“ häufig inkonsistent, mehrdeutig und oft sogar widersprüchlich verwendet werden. Ein Anliegen der Arbeit war es daher, die Nomenklatur so zu konsolidieren, dass sie als begrifflicher Kontext für die Diskussionen dieser und später darauf aufbauender Arbeiten genutzt werden kann. Insbesondere war es erforderlich, die bereits bestehenden Konzepte im Grid-Umfeld begrifflich einzuordnen.

Um die Tragweite des Problembereichs dieser Arbeit einschätzen zu können, wurden einige Szenarios im Umfeld bestehender und geplanter Grid-Projekte systematisch untersucht. Dazu wurden sie zunächst an Hand eines umfangreichen Kriterienkataloges positioniert und anschließend hinsichtlich ihrer VO-Managementkonzepte analysiert. Die Diskussion bestätigte die erwartete Heterogenität nicht nur bzgl. der technologischen Basis, sondern insbesondere auch bzgl. der Managementprozesse und der Auffassungen, was denn VO-Management ausmache. Dennoch ließen sich die Spezifika der betrachteten Real-Szenarios zu einem geschichteten VO-Modell abstrahieren, das mächtig genug war, als Basis für die weiteren Diskussionen zu dienen.

Aus diesem Modell ließen sich nun durch eine systematische Betrachtung von Schnittstellen, Verantwortungsbereichen und Abbildungen (Virtualisierungen) Anforderungen an ein VO-Management ableiten. Dieser Zwischenschritt war deshalb notwendig, damit die anschließend zu erstellende VO-Managementarchitektur in den richtigen funktionalen Kontext gestellt werden konnte. Gleichzeitig bildete das aus den Real-Szenarios abgeleitete Modell zusammen mit den Anforderungen an das VO-Management die Grundlage für eine systematische Spezifikation der Anforderungen an eine VO-Managementarchitektur, dem eigentlichen Kern dieser Arbeit, in Form von umfangreichen Anwendungsfällen – gespiegelt am Lebenszyklus Virtueller Organisationen. Im Rahmen dieser Analyse wurden nicht nur die beteiligten Aktoren identifiziert, sondern auch die Anwendungsfälle im Detail diskutiert, an denen die Aktoren beteiligt sind. Die Überlegungen zeigten insbesondere, dass das Management Virtueller Organisationen einen vielschichtigen Fragenkomplex darstellt, der neben realen Organisationen (als Provider realer Ressourcen und Dienste) auch die Interna Virtueller Organisationen und ganz besonders die Interessen von Communities als *breeding environments* zu berücksichtigen hatte.

Die Analyse des Status Quo zum VO-Management in Grids zeigte dann, dass keine Lösungen für die in dieser Arbeit adressierten Fragestellungen vorlagen, dass sich aber in einigen Teilbereichen Ansätze identifizieren ließen, die genutzt werden konnten. Diese waren hauptsächlich unter den Arbeiten in Standardisierungsgremien zu Web Services-Technologien (WSDM, WSRF) zu finden. Von ihnen wurde in der Arbeit zwar kein direkter Gebrauch gemacht, im Rahmen Plattform-spezifischer Umsetzungen der hier vorgestellten Ansätze spielten sie aber eine wichtige Rolle. Alle untersuchten Konzepte zeigten allerdings deutlich, dass bestehende Konzepte für das Management Virtueller Organisationen in Grids erhebliche Lücken aufweisen. Dies lag insbesondere daran, dass entweder die Annahmen

zu eng gesetzt wurden oder dass die Ansätze der losen Kopplung nicht Grid-spezifisch aufgebrochen wurden. Als wesentliche Schwierigkeit hat sich jedoch herausgestellt, dass *kein* Ansatz auf den kompletten Lebenszyklus einer VO fokussierte. Dies war jedoch eine der Grundvoraussetzungen.

Vor dem Hintergrund der Wiederverwendbarkeit von Modellen und der a priori unbekannten Einsatzplattformen der VO-Managementarchitektur (VOMA) wurde ein Entwicklungsansatz nach dem Model Driven Architecture (MDA)-Konzept für die Teilmodelle der Architektur gewählt. Im VOMA-Informationsmodell wurde für alle am VO-Management beteiligten Rollen ein gemeinsames Verständnis über die Managementinformationen festgelegt, die zwischen den Rollen ausgetauscht werden. Dazu wurde eine umfassende und über alle Ebenen integrierte Informationsbasis definiert. Dem MDA-Ansatz entsprechend wurde dabei als Spezifikationssprache UML verwendet und der Ansatz so gewählt, dass keine Plattform-spezifischen Annahmen getroffen wurden. Nur so konnte die angestrebte Wiederverwendbarkeit erzielt werden. Außerdem wurde bei dem Entwurf des Informationsmodells Wert darauf gelegt, ein Maximum an Managementinformationen möglichst nahe an der Wurzel der Vererbungshierachie zu platzieren. Erreicht wurde damit neben einer übersichtlichen Darstellung vor allem der Zusatznutzen, konkrete Vorgaben an den Minimalumfang geeigneter – auf das Management Virtueller Organisationen ausgerichteter – Instrumentierungen formulieren zu können.

Im Organisationsmodell wurden die am VO-Management beteiligten Rollen identifiziert und entsprechenden Handlungsdomänen zugeordnet. Dabei zeigte sich, dass die Domäne der realen Organisationen eine eher unterstützende Rolle im Lebenszyklus Virtueller Organisationen spielt, während die eigentliche Managementverantwortung auf der Ebenen der Communities zu finden war, zumindest was den Lebenszyklus von VOs angeht. Die VO selbst verantwortete ihren eigentlichen Betrieb. Es zeigte sich außerdem, dass zwar alle Rollen mit allen anderen interagieren konnten (und dies auch realiter tun), die primären Interaktionskanäle jedoch auf einige wenige reduziert werden konnten. Diese wurden genauer spezifiziert.

Das Kommunikationsmodell identifizierte die spezifischen Anforderungen an die Kommunikation der involvierten Rollen über die im Organisationsmodell spezifizierten Beziehungen. Dabei wurden wesentliche Eigenschaften der Dienste der eingesetzten Infrastruktur definiert. Da VOMA Plattform-unabhängig konzipiert wurde, waren diese Kommunikationsmechanismen ebenfalls generisch. Es wurden Kern-Operationen und darauf aufbauende Basisdienste festgelegt.

Im Funktionsmodell der VOMA-Architektur wurde der Gesamtaufgabenkomplex des VO-Managements auf der Basis des Organisationsmodells in einzelne Funktionsbereiche auf verschiedenen Ebenen gegliedert. Für jeden Funktionsbereich wurde konkret festgelegt, auf welchen Objektmodellen er sich abstützt und welche Interaktionen zwischen den jeweils involvierten Rollen stattfinden. Mit der Spezifikation der Funktionsbereiche wurde damit ein Satz von Funktionsbausteinen definiert, der für konkrete VO-Managementdienste genutzt werden kann.

Während VOMA zunächst Plattform-unabhängig spezifiziert wurde – und damit ein allgemeines Rahmenwerk lieferte, musste die Architektur für einen realen Einsatz Plattform-spezifisch formuliert werden. Die Transformation zwischen beiden Modellen wurde im MDA-Kontext auf der Basis entsprechender Abbildungsvorschriften am Beispiel einer WSDM-Abbildung demonstriert.

Für einen realen Einsatz der Architektur zum Management Virtueller Organisationen war neben der Definition der Architektur zu klären, wie diese in bestehende oder zukünftige Grid-Projekte ausgerollt bzw. integriert werden kann. Um diese Frage zu adressieren, wurde VOMA um eine Infrastrukturkomponente (VOMA-I) erweitert, über die VOMA in einem klassischen Manager/Agent-Paradigma zum Einsatz gebracht werden konnte. Dabei wurde nicht nur geklärt, wie die Komplexität möglicher Konfigurationen reduziert werden kann, sondern auch, wie VOMA für Managementanwendungen transparent genutzt werden kann. Die erste Fragestellung wurde über VO-Agenten und Konfigurationsmuster gelöst, die zweite über die konsequente Betrachtung Virtueller Organisationen und der damit eng zusammenhängenden Entitäten als instrumentierte *managed objects*. Der Vorteil dieses Ansatzes lag darin, nur einen kleinen Satz dedizierter Managementdienste zu benötigen. Die Tragfähigkeit des Konzeptes ist an Beispielen demonstriert worden.

8.2 Ausblick

Die in dieser Arbeit vorgelegte Definition einer VO-Managementarchitektur für Grids eignet sich hervorragend als Basis für eine systematische Entwicklung von VO-Managementanwendungen und zur Konstruktion neuer VO-Managementkonzepte, die insbesondere mit dynamischeren VOs erforderlich werden. Die hier vorgestellte *Library* von Managementdiensten muss dann einem zuverlässigen und schnellen Kompositionsmechanismus zur Laufzeit zugeführt werden. Inwieweit der hier genutzte MDA-Ansatz ein solches Vorgehen unterstützt, ist im Einzelfall zu klären.

Aus dem Kontext des Modellierungsansatzes der hier vorgestellten Lösung ergeben sich einige Fragen, die im Rahmen dieser Arbeit nicht behandelt werden konnten und wesentlich auf Plattform-Spezifika beruhen. Für die Transformation des Plattform-unabhängigen Modells in Plattform-spezifische Umgebungen müssen sowohl die Modelle als auch die Abbildungsvorschriften zur Verfügung stehen. Benötigt werden insbesondere Modelle für WSDM-Umgebungen, für das Globus Toolkit, für UNICORE und für gLite-Umgebungen, da diese die Basis fast aller Grid-Projekte bilden [Milke u. a., 2006b]. Transformationsmechanismen im Sinne vorgefertigter *Cartridges* [Breuer, 2004] sind hier wünschenswert. In der Arbeit konnte diese Transformation nur rudimentär am Beispiel WSDM demonstriert werden, da die WSDM-Konzepte, das darunter liegende WSRF-Rahmenwerk und dessen Grid-spezifische Abstraktion (OGSA) in den aktuell vorliegenden Versionen noch

unvollständig sind. Inwieweit die Apache Muse-Implementierung der WSDM-Spezifikation[1] hierbei hilfreich sein kann, wird zur Zeit ebenso untersucht wie die Integrierbarkeit vorhandener Werkzeuge wie VOMS, VOMRS und Shibboleth [Amikem, 2007; Cojocaru, 2007; Hellwig, 2007; Ziegler u. Grimm, 2006].

Im Rahmen dieser Arbeit konnten keine aussagekräftigen Komplexitäts- und Skalierbarkeitsuntersuchungen durchgeführt werden. So ist zur Zeit noch unklar, welche Parameter ab welcher Größenordnung einen deutlichen Effekt auf die Anwendbarkeit des Ansatzes besitzen und welche Parameter für ein gezieltes Leistungsmanagement kritisch sind. Eine interessante Fragestellung in diesem Zusammenhang ist die optimale Konfigurationsstrategie für VOMA-I: Unter welchen Umständen ist es besser, einen VO-Agenten mit vielen MOAMs zu konfigurieren, statt viele Agenten mit einer MOAM? Gibt es bestimmte Muster für bestimmte Situationen?

In dieser Arbeit wurden zahlreiche Fragen ausgeklammert, die jedoch an anderer Stelle intensiv behandelt werden. Eine offene Frage betrifft insbesondere die Ableitung des `TrustLevels` in den Klassen `TrustedAuthorities` und `VOMAEntityIdentification`. Dies führt unmittelbar zur Einführung eines Qualitätsbegriffs für Organisationen, dem nicht nur Vertrauensattribute zu Grunde liegen müssen, sondern auch – gerade bei Virtuellen Organisationen – die Historie der Organisation und möglicherweise deren verbleibende Lebensdauer. Der Einfluss eines solchen Qualitätsbegriffs ist in dieser Arbeit unberücksichtigt geblieben. Ebenso ist unberücksichtigt geblieben, wie die hier vorgestellte Lösung in ein Grid-weites Service- und Supportkonzept integriert werden kann, insbesondere unter der Prämisse eines global operierenden *Grid Operations Centers* [Avery u. Foster, 2003].

Diese Überlegungen sind in zukünftigen Arbeiten von Bedeutung, wenn darauf abgezielt wird, die hier entwickelten Konzepte fortzuführen und in die Grid Community über Standardisierungs-Plattformen der Globus Alliance und des Open Grid Forums (OGF) einzubringen.

[1] `http://ws.apache.org/muse/`

Anhang A

Vollständiges Verzeichnis der verwendeten UML-Packages

public package *Base Types*	
Beschreibung	Abschnitt 5.2.2
Abhängigkeit	keine
Klassen	OCL
	Path
	Template
	TimePeriod
	URI
	URL

public package *Management*	
Beschreibung	Abschnitt 5.2.4
Siehe auch	VirtualOrganization, TopLevel
Abhängigkeit	[VOMA Information Model].VirtualOrganization
	[VOMA Information Model].TopLevel
Klassen	ActivityNode
	AtomicPolicyEvent
	AtomicPolicyEventSpec
	CommonMemberScope
	CommonResourceScope
	CommonRoleScope
	CommonServiceScope

public package *Management* (Fortsetzung)
CommonVOMAPolicyScope
CommonVOScope
CompositePolicyEvent
CompositePolicyEventSpec
ControlNode
LocalManagementSystem
LoggingSpecificEvent
LoggingSpecificEventSpec
MemberAsTarget
MemberSpecificEvent
MemberSpecificEventSpec
RoleAsTarget
RoleSpecificEvent
RoleSpecificEventSpec
SiteInvolvementDetails
VOAsTarget
VOMAAgent
VOMAArchive
VOMAContract
VOMAContractItem
VOMALoggingRecord
VOMAManagingRole
VOManagementInteraction
VOManagementInteractionItem
VOMANotification
VOMAOperationRequest
VOMAPolicy
VOMAPolicyApplication
VOMAPolicyEvent
VOMAPolicyEventSpecification
VOMAPolicyRepository
VOMAPolicyRuleSpecification
VOMAPolicySpecification
VOMAPolicyTarget
VOMAResponse
VOMASpecificSLA
VOMAWorkflow
VOMAWorkflowNode
VOMAWorkflowTransition

public package *Management* (Fortsetzung)	
	VOSpecificEvent
	VOSpecificEventSpec

public package *Member*	
Beschreibung	Abschnitt 5.2.5
Siehe auch	VirtualOrganization, TopLevel
Abhängigkeit	[VOMA Information Model].VirtualOrganization
	[VOMA Information Model].TopLevel
Klassen	AccountPassword
	ApplicationIsAcceptedDetails
	AuthorizationDetails
	EmployeeIdentification
	Group
	GroupName
	GroupOfParticipants
	IdentityCard
	IndividualAppliesforMembershipDetails
	IndividualIdentification
	IndividualName
	OrganizationAppliesForMembershipDetails
	PublicKeyCertificate
	RealOrganizationIdentification
	RealOrganizationName
	TradeRegisterCertificate
	VOMember

public package *Role*	
Beschreibung	Abschnitt 5.2.6
Siehe auch	TopLevel, VirtualOrganization
Abhängigkeit	[VOMA Information Model].TopLevel

public package *Role* (Fortsetzung)	
Klassen	[VOMA Information Model].VirtualOrganization
	RoleIdentification
	RoleInVO
	RoleName
	VOMACapabilitiesSpecification

public package *Trusted Entities*	
Beschreibung	Abschnitt 5.2.9
Siehe auch	VirtualOrganization
Abhängigkeit	[VOMA Information Model].VirtualOrganization
Klassen	Escrow
	Notary
	TrustedAuthorities

public package *VirtualOrganization*	
Beschreibung	Abschnitt 5.2.3
Siehe auch	Management, Member, Role, Trusted Entities, TopLevel
Abhängigkeit	[VOMA Information Model].Management
	[VOMA Information Model].Member
	[VOMA Information Model].Trusted Entities
	[VOMA Information Model].TopLevel
	[VOMA Information Model].Role
Klassen	CollaborationMethod
	CompoundEntityLifecycle
	ContactMethod
	emailContact
	External
	FaxContact
	GroupWare
	OrdinaryMember

public package *VirtualOrganization* (Fortsetzung)	
	PhoneContact
	PostalContact
	RoleOfVO
	SimpleEntityLifecycle
	VirtualOrganization
	VirtualResourceConsumer
	VirtualResourceProvider
	VirtualServiceConsumer
	VirtualServiceProvider
	virtualStaff
	VOAdministrator
	VOApplicants
	VOArchiveManager
	VOAsConsumer
	VOAsManager
	VOAsMember
	VOAsProvider
	VOIdentification
	VOInitiator
	VOMAEntityIdentification
	VOMAEntityName
	VOMALifecyclePhase
	VOManager
	VOMARoleSpecification
	VOName
	VOParticipant
	VOProvider
	VOSLAManager
	WebPageContact
	WebPortal
	WikiPlatform

public package *VirtualResource*	
Beschreibung	Abschnitt 5.2.7
Siehe auch	TopLevel

public package *VirtualResource* (Fortsetzung)	
Abhängigkeit	[VOMA Information Model].TopLevel
Klassen	Binding
	ComputingResource
	LogicalRealResource
	PhysicalRealResource
	StorageResource
	VirtualResource
	VRSpecificEvent
	VRSpecificEventSpec

public package *VirtualService*	
Beschreibung	Abschnitt 5.2.8
Siehe auch	TopLevel
Abhängigkeit	[VOMA Information Model].TopLevel
Klassen	KeyValuePairs
	VirtualService
	VSSpecificEvent
	VSSpecificEventSpec

public package *TopLevel*	
Beschreibung	Abschnitt 5.2.2
Siehe auch	VirtualOrganization, Management, Member
Abhängigkeit	[Domain Model].VirtualOrganization
	[Domain Model].Management
	[Domain Model].Member
Klassen	Community
	GovernmentAgency
	Individual
	LegalCompany
	ManagedVOMAEntity

public package *TopLevel* (Fortsetzung)
NonProfitOrganization
RealOrganization
RealOrgResource
RealOrgService
UnManagedVOMAEntity
VirtualOrganization
VirtualResource
VirtualService
VOMAEntity
VOMAEntityIdentification
VOMAEntityLifecycle
VOMAEntityName
VOMAEntityRoles
VOMAManagementArea
VOMARootEntity
VOMember
VORole

Anhang B

Vollständiges Verzeichnis der verwendeten UML-Klassen

B.1 Klassen des Package `BaseType`

public class *OCL*	
Beschreibung	Abschnitt 5.2.2
Stereotyp	≪`VOMABaseType`≫
Siehe auch	URI
Attribute	private URI `linkToOCLConstruct`

public class *Path*	
Beschreibung	Abschnitt 5.2.2
Stereotyp	≪`VOMABaseType`≫
Attribute	private String `path`

public class *Template*	
Beschreibung	Abschnitt 5.2.2
Stereotyp	≪`VOMABaseType`≫
Siehe auch	URI
Attribute	private URI `linkToTemplateSpecification`

public class *TimePeriod*	
Beschreibung	Abschnitt 5.2.2
Stereotyp	≪VOMABaseType≫
Attribute	private DateTime `startTime`
	private DateTime `endTime`

public class *URI*	
Beschreibung	Abschnitt 5.2.2
Stereotyp	≪VOMABaseType≫
Siehe auch	Template, URL, OCL
Abhängigkeit	Generalisierung: [Base Types].URL
Attribute	private String `scheme`
	private String `schemeSpecificPart`
	private String `authority`
	private String `userInfo`
	private String `host`
	private String `port`
	private String `path`
	private String `query`
	public String[] `fragment`

public class *URL*	
Beschreibung	Abschnitt 5.2.2
Extends	URI
Stereotyp	≪VOMABaseType≫
Siehe auch	URI
Abhängigkeit	Generalisierung: [Base Types].URI
Attribute	private String `protocol`

B.2 Klassen des Package `TopLevel`

public class *CommonVOMAManagementScope*	
Beschreibung	Abschnitt 5.2.2
Extends	VOMARootEntity
Siehe auch	VOMARootEntity, ManagedVOMAEntity, CommonMemberScope, CommonRoleScope, CommonVOMAPolicyScope, CommonResourceScope, CommonServiceScope, CommonVOScope
Abhängigkeit	Generalisierung: [TopLevel].VOMARootEntity Aggregation: [TopLevel].CommonVOMAManagementScope Aggregation: [TopLevel].ManagedVOMAEntity Generalisierung: [Management].CommonMemberScope Generalisierung: [Management].CommonRoleScope Generalisierung: [Management].CommonVOMAPolicyScope Generalisierung: [Management].CommonResourceScope Generalisierung: [Management].CommonServiceScope Generalisierung: [Management].CommonVOScope
Attribute	public String `scopeName` public String `scopeCharacteristic`

public class *Community*	
Beschreibung	Abschnitt 5.2.2
Extends	UnManagedVOMAEntity
Siehe auch	UnManagedVOMAEntity, RealOrganization, Individual, VirtualOrganization, TrustedAuthorities
Abhängigkeit	Generalisierung: [TopLevel].UnManagedVOMAEntity Aggregation: [TopLevel].RealOrganization Aggregation: [TopLevel].Individual Aggregation: [VirtualOrganization].VirtualOrganization Assoziation: [Trusted Entities].TrustedAuthorities
Attribute	public String `communitySpokesman`

public class *GovernmentAgency*	
Beschreibung	Abschnitt 5.2.2
Extends	RealOrganization
Siehe auch	RealOrganization
Abhängigkeit	Generalisierung: [TopLevel].RealOrganization

public class *Individual*	
Beschreibung	Abschnitt 5.2.2
Constraints	country codes MUST comply to ISO 3166, academicTitle MUST comply to RFC 1274, gender MUST comply to ISO 5218
Extends	UnManagedVOMAEntity
Siehe auch	IndividualName, UnManagedVOMAEntity, Community, IndividualIdentification, RealOrganization, VOApplicants
Abhängigkeit	Aggregation: [Member].IndividualName Generalisierung: [TopLevel].UnManagedVOMAEntity Aggregation: [TopLevel].Community Aggregation: [Member].IndividualIdentification Aggregation: [TopLevel].RealOrganization Assoziation: [VirtualOrganization].VOApplicants
Attribute	public DateTime `dateOfBirth` public String `currentNationality` public String `homeAddress` public String `placeOfBirth` public String `originalNationality` public String `gender` public String `academicTitle`

public class *LegalCompany*	
Beschreibung	Abschnitt 5.2.2

public class *LegalCompany* (Fortsetzung)	
Extends	RealOrganization
Siehe auch	RealOrganization
Abhängigkeit	Generalisierung: [TopLevel].RealOrganization

public class *ManagedVOMAEntity*	
Beschreibung	Abschnitt 5.2.2
Extends	VOMAEntity
Siehe auch	VOMAEntity, VirtualOrganization, RoleInVO, VOMember, VOMAEntityLifecycle, VOMAEntityLifecycle, CommonVOMAManagementScope, VOMAPolicy, VOMAPolicyApplication, GroupOfParticipants, VirtualResource, VirtualService, VOMALifecyclePhase
Abhängigkeit	Generalisierung: [TopLevel].VOMAEntity Generalisierung: [VirtualOrganization].VirtualOrganization Generalisierung: [Role].RoleInVO Generalisierung: [Member].VOMember Aggregation: [TopLevel].VOMAEntityLifecycle Generalisierung: [TopLevel].VOMAEntityLifecycle Aggregation: [TopLevel].CommonVOMAManagementScope Generalisierung: [Management].VOMAPolicy Generalisierung: [Management].VOMAPolicyApplication Generalisierung: [Member].GroupOfParticipants Generalisierung: [VirtualResource].VirtualResource Generalisierung: [VirtualService].VirtualService Generalisierung: [Configuration Management].VOConfiguration Generalisierung: [VOMA Communication Model].VOMASession Generalisierung: [VOMA Communication Model].VOMANotificationService Generalisierung: [VOMA Communication Model].VOMALoggingService Generalisierung: [Configuration Management].VOConfigRequest Generalisierung: [Accounting Management].VOUsageRecord Generalisierung: [VirtualOrganization].VOMALifecyclePhase

public class *NonProfitOrganization*	
Beschreibung	Abschnitt 5.2.2
Extends	RealOrganization
Siehe auch	RealOrganization
Abhängigkeit	Generalisierung: [TopLevel].RealOrganization

public class *RealOrganization*	
Beschreibung	Abschnitt 5.2.2
Constraint	country codes MUST comply to ISO 3166
Extends	UnManagedVOMAEntity
Siehe auch	UnManagedVOMAEntity, LegalCompany, NonProfitOrganization, GovernmentAgency, Community, Individual, RealOrganizationName, RealOrganizationIdentification, VOApplicants, RealOrgResource, LocalManagementSystem, RealOrgService, VOMASite
Abhängigkeit	Generalisierung: [TopLevel].RealOrganization Generalisierung: [TopLevel].UnManagedVOMAEntity Generalisierung: [TopLevel].LegalCompany Generalisierung: [TopLevel].NonProfitOrganization Generalisierung: [TopLevel].GovernmentAgency Aggregation: [TopLevel].RealOrganization Aggregation: [TopLevel].Community Aggregation: [TopLevel].Individual Aggregation: [Member].RealOrganizationName Aggregation: [Member].RealOrganizationIdentification Assoziation: [VirtualOrganization].VOApplicants Aggregation: [TopLevel].RealOrgResource Aggregation: [Management].LocalManagementSystem Aggregation: [TopLevel].RealOrgService Aggregation: [TopLevel].VOMASite
Attribute	public DateTime `foundationDate` public String `legalAddress` public String `originalNationality` public String `currentNationality` public String `motherOrganization`

public class *RealOrgResource*	
Beschreibung	Abschnitt 5.2.2
Extends	UnManagedVOMAEntity
Siehe auch	UnManagedVOMAEntity, RealOrganization, LocalManagementSystem, PhysicalRealResource, LogicalRealResource, VirtualResource
Abhängigkeit	Generalisierung: [TopLevel].UnManagedVOMAEntity Aggregation: [TopLevel].RealOrganization Aggregation: [Management].LocalManagementSystem Generalisierung: [VirtualResource].PhysicalRealResource Generalisierung: [VirtualResource].LogicalRealResource Aggregation: [VirtualResource].VirtualResource

public class *RealOrgService*	
Beschreibung	Abschnitt 5.2.2
Extends	UnManagedVOMAEntity
Siehe auch	UnManagedVOMAEntity, LocalManagementSystem, RealOrganization, VirtualService
Abhängigkeit	Generalisierung: [TopLevel].UnManagedVOMAEntity Aggregation: [Management].LocalManagementSystem Aggregation: [TopLevel].RealOrganization Aggregation: [VirtualService].VirtualService Aggregation: [TopLevel].RealOrgService

public class *UnManagedVOMAEntity*	
Beschreibung	Abschnitt 5.2.2
Extends	VOMAEntity
Siehe auch	VOMAEntity, RealOrgService, RealOrgResource, Individual, RealOrganization, Community, VOMAPolicyApplication, LocalManagementSystem

public class *UnManagedVOMAEntity* (Fortsetzung)	
Abhängigkeit	Generalisierung: [TopLevel].VOMAEntity Generalisierung: [TopLevel].RealOrgService Generalisierung: [TopLevel].RealOrgResource Generalisierung: [TopLevel].Individual Generalisierung: [TopLevel].RealOrganization Generalisierung: [TopLevel].Community Assoziation: [Management].VOMAPolicyApplication Generalisierung: [Management].LocalManagementSystem

public class *VOMACollection*	
Beschreibung	Abschnitt 5.2.2
Extends	VOMARootEntity
Siehe auch	VOMARootEntity, VOMAPolicyEvent, VOMAPolicyRepository, VOMAArchive, VOMALoggingRecord
Abhängigkeit	Generalisierung: [TopLevel].VOMARootEntity Generalisierung: [Management].VOMAPolicyEvent Generalisierung: [Management].VOMAPolicyRepository Generalisierung: [Management].VOMAArchive Generalisierung: [Management].VOMALoggingRecord Generalisierung: [Accounting Management].VOUsageRecord
Attribute	public int `collectionType`

public class *VOMAEntity*	
Beschreibung	Abschnitt 5.2.2
Extends	VOMARootEntity
Siehe auch	VOMARootEntity, ManagedVOMAEntity, UnManagedVOMAEntity, VOMAEntityIdentification, VOMAPolicyTarget, VOManagementInteractionItem
Abhängigkeit	Generalisierung: [TopLevel].VOMARootEntity Generalisierung: [TopLevel].ManagedVOMAEntity

public class *VOMAEntity* (Fortsetzung)	
	Generalisierung: [TopLevel].UnManagedVOMAEntity
	Aggregation: [VirtualOrganization].VOMAEntityIdentification
	Generalisierung: [Management].VOMAPolicyTarget
	Generalisierung: [Management].VOManagementInteractionItem
Attribute	public DateTime `creationDate`

public class *VOMAEntityLifecycle*	
Beschreibung	Abschnitt 5.2.2
Extends	VOMARootEntity, ManagedVOMAEntity
Siehe auch	VOMARootEntity, ManagedVOMAEntity, ManagedVOMAEntity, SimpleEntityLifecycle, CompoundEntityLifecycle, CompoundEntityLifecycle
Abhängigkeit	Generalisierung: [TopLevel].VOMARootEntity
	Aggregation: [TopLevel].ManagedVOMAEntity
	Generalisierung: [TopLevel].ManagedVOMAEntity
	Generalisierung: [VirtualOrganization].SimpleEntityLifecycle
	Generalisierung: [VirtualOrganization].CompoundEntityLifecycle
	Aggregation: [VirtualOrganization].CompoundEntityLifecycle
	Assoziation: [Configuration Management].VOConfiguration
Attribute	public String `lifecycleName`
	public DateTime `lifecycleDueDate`
	public DateTime `lifecycleStartDate`
	public int `lifecycleOriginalPriority`
	public int `lifecyclePreemptionStyle`
	public int `lifecycleCurrentPriority`
	public int `lifecycleStatus`

public class *VOMAEntityRoles*	
Beschreibung	Abschnitt 5.2.2
Extends	VOMARootEntity

public class *VOMAEntityRoles* (Fortsetzung)	
Siehe auch	CollaborationMethod, VOMARootEntity, RoleOfVO, ContactMethod, VOMARoleSpecification, RoleInVO, VOManagementInteractionItem, VOMAManagingRole, VOMAAgent
Abhängigkeit	Assoziation: [VirtualOrganization].CollaborationMethod Generalisierung: [TopLevel].VOMARootEntity Generalisierung: [VirtualOrganization].RoleOfVO Assoziation: [VirtualOrganization].ContactMethod Aggregation: [VirtualOrganization].VOMARoleSpecification Generalisierung: [Role].RoleInVO Assoziation: [Management].VOManagementInteractionItem Generalisierung: [Management].VOMAManagingRole Generalisierung: [Management].VOMAAgent
Attribute	public int `roleID` public String `roleCategory` public TimePeriod `valiedFor` public int `roleType`

public class *VOMARootEntity*	
Beschreibung	Abschnitt 5.2.2
Extends	VOMAEntity, VOMAEntityIdentification, VOMAEntityLifecycle, CommonVOMAManagementScope, VOMAEntityRoles, VOMASpecification, VOMACollection
Siehe auch	Main VOMA Top Level Entities, VO Naming and Identification, Roles of VOs, VO Contact and Collaboration, VO Management Domains, Individual VO Membership, VOMA Policiy Framework, VOMA Policy Events, VOMA Policy Repository, Roles in VOs, Organizational VO Membership, Generalization Dependencies, VOMA Archive
Abhängigkeit	Generalisierung: [TopLevel].VOMAEntityLifecycle Generalisierung: [TopLevel].CommonVOMAManagementScope Generalisierung: [TopLevel].VOMAEntityRoles Generalisierung: [TopLevel].VOMASpecification Generalisierung: [TopLevel].VOMACollection
Attribute	public String `objectID` public String `entityName` public String `description`

public class *VOMARootEntity* (Fortsetzung)	
Operationen	public String `vomaVersion` public `getEntityName():String` public `setEntityName():Integer` public `getDescription():String` public `setDescription():Integer` public `getObjectID():String` public `setObjectID():Integer` public `getVOMAVersion():String` public `setVOMAVersion():Integer`

public class *VOMASite*	
Beschreibung	Abschnitt 5.2.2
Extends	
Siehe auch	See also: VOMAPolicyRepository, LocalManagementSystem, VirtualResource, VOManagementInteraction, ContactMethod, RealOrganization, VirtualOrganization, VirtualService, VOMAArchive
Abhängigkeit	Aggregation: [Management].VOMAPolicyRepository Aggregation: [Management].LocalManagementSystem Aggregation: [VirtualResource].VirtualResource Assoziation: [Management].VOManagementInteraction Aggregation: [VirtualOrganization].ContactMethod Aggregation: [TopLevel].RealOrganization Aggregation: [VirtualOrganization].VirtualOrganization Aggregation: [VirtualService].VirtualService Aggregation: [Management].VOMAArchive
Attribute	public int `siteID` public String `siteDescription` public String `siteAddress` public String `siteName` public URI `siteWebPage`

public class *VOMASpecification*	
Beschreibung	Abschnitt 5.2.2
Extends	VOMARootEntity
Siehe auch	See also:
Abhängigkeit	Generalisierung: [VirtualOrganization].CollaborationMethod Generalisierung: [VirtualOrganization].ContactMethod Generalisierung: [Management].VOMAPolicySpecification Generalisierung: [TopLevel].VOMARootEntity Generalisierung: [Management].VOMAWorkflow Generalisierung: [VirtualOrganization].VOMARoleSpecification Generalisierung: [Role].VOMACapabilitiesSpecification Generalisierung: [Management].VOMASpecificSLA

B.3 Klassen des Package `VirtualOrganization`

public class *CollaborationMethod*	
Beschreibung	Abschnitt 5.2.3
Extends	VOMASpecification
Siehe auch	VOMASpecification, WikiPlatform, WebPortal, GroupWare, VOMAEntityRoles
Abhängigkeit	Generalisierung: [TopLevel].VOMASpecification Generalisierung: [VirtualOrganization].WikiPlatform Generalisierung: [VirtualOrganization].WebPortal Generalisierung: [VirtualOrganization].GroupWare Assoziation: [TopLevel].VOMAEntityRoles
Attribute	public int `ContactType`

public class *CompoundEntityLifecycle*	
Beschreibung	Abschnitt 5.2.3

public class *CompoundEntityLifecycle* (Fortsetzung)	
Extends	VOMAEntityLifecycle
Siehe auch	VOMAEntityLifecycle, VOMAEntityLifecycle
Abhängigkeit	Generalisierung: [TopLevel].VOMAEntityLifecycle
	Aggregation: [TopLevel].VOMAEntityLifecycle
Operationen	public `getCompoundTimeToLive():int`

public class *ContactMethod*	
Beschreibung	Abschnitt 5.2.3
Extends	VOMASpecification
Siehe auch	FaxContact, PostalContact, emailContact, VOMASpecification, VOMAEntityRoles, PhoneContact, WebPageContact, VOMASite
Abhängigkeit	Generalisierung: [VirtualOrganization].FaxContact
	Generalisierung: [VirtualOrganization].PostalContact
	Generalisierung: [VirtualOrganization].emailContact
	Generalisierung: [TopLevel].VOMASpecification
	Assoziation: [TopLevel].VOMAEntityRoles
	Generalisierung: [VirtualOrganization].PhoneContact
	Generalisierung: [VirtualOrganization].WebPageContact
	Aggregation: [TopLevel].VOMASite
Attribute	public int `ContactType`

public class *emailContact*	
Beschreibung	Abschnitt 5.2.3
Constraints	URI MUST be compliant to RFC 2396
Extends	ContactMethod
Siehe auch	ContactMethod
Abhängigkeit	Generalisierung: [VirtualOrganization].ContactMethod
Attribute	public URI `emailAddress`

public class *External*	
Beschreibung	Abschnitt 5.2.3
Extends	VOParticipant
Siehe auch	VOParticipant
Abhängigkeit	Generalisierung: [VirtualOrganization].VOParticipant

public class *FaxContact*	
Beschreibung	Abschnitt 5.2.3
Extends	ContactMethod
Siehe auch	ContactMethod
Abhängigkeit	Generalisierung: [VirtualOrganization].ContactMethod
Attribute	public String `internationalFaxNumber`
	public String `internalFaxNumber`

public class *GroupWare*	
Beschreibung	Abschnitt 5.2.3
Extends	CollaborationMethod
Siehe auch	CollaborationMethod
Abhängigkeit	Generalisierung: [VirtualOrganization].CollaborationMethod
Attribute	public String `groupwareProduct`
	public String `groupwareVersion`

public class *OrdinaryMember*	
Beschreibung	Abschnitt 5.2.3
Extends	VOMember
Siehe auch	VOMember
Abhängigkeit	Generalisierung: [Member].VOMember

public class *PhoneContact*	
Beschreibung	Abschnitt 5.2.3
Extends	ContactMethod
Siehe auch	ContactMethod
Abhängigkeit	Generalisierung: [VirtualOrganization].ContactMethod
Attribute	public String `internalPhoneNumber`
	public String `internationalPhoneNumber`
	public String `mobilePhoneNumber`

public class *PostalContact*	
Beschreibung	Abschnitt 5.2.3
Extends	ContactMethod
Siehe auch	ContactMethod
Abhängigkeit	Generalisierung: [VirtualOrganization].ContactMethod
Attribute	public String `postalAddress`

public class *RoleOfVO*	
Beschreibung	Abschnitt 5.2.3
Extends	VOMAEntityRoles
Siehe auch	VOAsProvider, VOAsConsumer, VOAsManager, VOMAEntityRoles, VirtualOrganization, VOAsMember
Abhängigkeit	Generalisierung: [VirtualOrganization].VOAsProvider
	Generalisierung: [VirtualOrganization].VOAsConsumer
	Generalisierung: [VirtualOrganization].VOAsManager
	Generalisierung: [TopLevel].VOMAEntityRoles
	Aggregation: [VirtualOrganization].VirtualOrganization
	Generalisierung: [VirtualOrganization].VOAsMember
Attribute	public DateTime `roleCreationDate`
	public int `voRoleType`

public class *SimpleEntityLifecycle*	
Beschreibung	Abschnitt 5.2.3
Extends	VOMAEntityLifecycle
Siehe auch	VOMAEntityLifecycle, VOMALifecyclePhase
Abhängigkeit	Generalisierung: [TopLevel].VOMAEntityLifecycle
	Aggregation: [VirtualOrganization].VOMALifecyclePhase
Operationen	public `getTimeToLive():int`

public class *VirtualOrganization*	
Beschreibung	Abschnitt 5.2.3
Extends	ManagedVOMAEntity
Siehe auch	ManagedVOMAEntity, VOIdentification, VOName, Community, RoleOfVO, VOAsTarget, RoleInVO, VirtualResource, VirtualService, VOMASite
Abhängigkeit	Generalisierung: [TopLevel].ManagedVOMAEntity
	Generalisierung: [VirtualOrganization].VirtualOrganization
	Aggregation: [VirtualOrganization].VirtualOrganization
	Aggregation: [VirtualOrganization].VOIdentification
	Aggregation: [VirtualOrganization].VOName
	Aggregation: [TopLevel].Community
	Aggregation: [VirtualOrganization].RoleOfVO
	Aggregation: [Management].VOAsTarget
	Aggregation: [Role].RoleInVO
	Aggregation: [VirtualResource].VirtualResource
	Aggregation: [VirtualService].VirtualService
	Aggregation: [TopLevel].VOMASite
	Aggregation: [Configuration Management].VOConfiguration
Attribute	public DateTime `foundationDate`
	public TimePeriod `validFor`
	public String `founder`
	public String `foundingCommunity`
	public String[] `listofDirectSubVOs`
	public String[] `listofDirectSuperVOs`

public class *VirtualOrganization* (Fortsetzung)	
	public String[] `listOfVirtualResourcesOffered`
	public String[] `listOfVirtualResourcesActive`
	public String[] `listOfVirtualResourcesBlocked`
	public String[] `listOfVirtualServicesOffered`
	public String[] `listOfVirtualServicesActive`
	public String[] `listOfVirtualServicesBlocked`
	public String[] `listOfRoles`
	public String[] `listOfGuestMembers`
	public String[] `listOfExternalMembers`
	public String[] `listOfOrdinaryMembers`
	public String[][] `assignedRolesToMembers`
	public String[] `listOfParticipatingROs`
	public String[] `listOfVOMASites`
	public int `currentLifecycleStatus`
Operationen	public `getTimeToLive():int`
	public `purgeLists():boolean`
	public `purgeQueues():boolean`

public class *VirtualResourceConsumer*	
Beschreibung	Abschnitt 5.2.3
Extends	VOAsConsumer
Siehe auch	VOAsConsumer
Abhängigkeit	Generalisierung: [VirtualOrganization].VOAsConsumer

public class *VirtualResourceProvider*	
Beschreibung	Abschnitt 5.2.3
Extends	VOAsProvider
Siehe auch	VOAsProvider
Abhängigkeit	Generalisierung: [VirtualOrganization].VOAsProvider

public class *VirtualServiceConsumer*	
Beschreibung	Abschnitt 5.2.3
Extends	VOAsConsumer
Siehe auch	VOAsConsumer
Abhängigkeit	Generalisierung: [VirtualOrganization].VOAsConsumer

public class *VirtualServiceProvider*	
Beschreibung	Abschnitt 5.2.3
Extends	VOAsProvider
Siehe auch	VOAsProvider
Abhängigkeit	Generalisierung: [VirtualOrganization].VOAsProvider

public class *virtualStaff*	
Beschreibung	Abschnitt 5.2.3
Constraint	memberType = 1 (regular)
Extends	VOMember, VOMAManagingRole
Siehe auch	VOMember, VOManager, VOAdministrator, VOArchiveManager, VOSLAManager, VOMAManagingRole
Abhängigkeit	Generalisierung: [Member].VOMember Generalisierung: [VirtualOrganization].VOManager Generalisierung: [VirtualOrganization].VOAdministrator Generalisierung: [VirtualOrganization].VOArchiveManager Generalisierung: [VirtualOrganization].VOSLAManager Generalisierung: [Management].VOMAManagingRole

public class *VOAdministrator*	
Beschreibung	Abschnitt 5.2.3
Extends	virtualStaff
Siehe auch	virtualStaff
Abhängigkeit	Generalisierung: [VirtualOrganization].virtualStaff

public class *VOApplicants*	
Beschreibung	Abschnitt 5.2.3
Extends	RoleInVO
Siehe auch	RoleInVO, Individual, VOParticipant, RealOrganization
Abhängigkeit	Generalisierung: [Role].RoleInVO
	Assoziation: [TopLevel].Individual
	Assoziation: [VirtualOrganization].VOParticipant
	Assoziation: [TopLevel].RealOrganization
Attribute	public int `typeOfApplication`

public class *VOArchiveManager*	
Beschreibung	Abschnitt 5.2.3
Extends	virtualStaff
Siehe auch	virtualStaff, VOMAArchive
Abhängigkeit	Generalisierung: [VirtualOrganization].virtualStaff
	Assoziation: [Management].VOMAArchive

public class *VOAsConsumer*	
Beschreibung	Abschnitt 5.2.3
Extends	RoleOfVO

public class *VOAsConsumer* (Fortsetzung)	
Siehe auch	RoleOfVO, VirtualServiceConsumer, VirtualResourceConsumer
Abhängigkeit	Generalisierung: [VirtualOrganization].RoleOfVO
	Generalisierung: [VirtualOrganization].VirtualServiceConsumer
	Generalisierung: [VirtualOrganization].VirtualResourceConsumer

public class *VOAsManager*	
Beschreibung	Abschnitt 5.2.3
Extends	RoleOfVO, VOMAManagingRole
Siehe auch	RoleOfVO, VOInitiator, VOProvider, VOMAManagingRole
Abhängigkeit	Generalisierung: [VirtualOrganization].RoleOfVO
	Generalisierung: [VirtualOrganization].VOInitiator
	Generalisierung: [VirtualOrganization].VOProvider
	Generalisierung: [Management].VOMAManagingRole

public class *VOAsMember*	
Beschreibung	Abschnitt 5.2.3
Extends	RoleOfVO, VOMember
Siehe auch	RoleOfVO, VOMember
Abhängigkeit	Generalisierung: [VirtualOrganization].RoleOfVO
	Generalisierung: [Member].VOMember

public class *VOAsProvider*	
Beschreibung	Abschnitt 5.2.3
Extends	RoleOfVO
Siehe auch	RoleOfVO, VirtualServiceProvider, VirtualResourceProvider

public class *VOAsProvider* (Fortsetzung)	
Abhängigkeit	Generalisierung: [VirtualOrganization].RoleOfVO Generalisierung: [VirtualOrganization].VirtualServiceProvider Generalisierung: [VirtualOrganization].VirtualResourceProvider

public class *VOIdentification*	
Beschreibung	Abschnitt 5.2.3
Extends	VOMAEntityIdentification
Siehe auch	VOMAEntityIdentification, VirtualOrganization, TrustedAuthorities
Abhängigkeit	Generalisierung: [VirtualOrganization].VOMAEntityIdentification Aggregation: [VirtualOrganization].VirtualOrganization Assoziation: [Trusted Entities].TrustedAuthorities
Attribute	public String `identificationDetails`

public class *VOInitiator*	
Beschreibung	Abschnitt 5.2.3
Extends	VOAsManager
Siehe auch	VOAsManager
Abhängigkeit	Generalisierung: [VirtualOrganization].VOAsManager

public class *VOMAEntityIdentification*	
Beschreibung	Abschnitt 5.2.3
Extends	VOMARootEntity
Siehe auch	VOIdentification, VOMARootEntity, VOMAEntityName, VOMAEntity, IndividualIdentification, RealOrganizationIdentification, RoleIdentification

public class *VOMAEntityIdentification* (Fortsetzung)	
Abhängigkeit	Generalisierung: [VirtualOrganization].VOIdentification
	Generalisierung: [TopLevel].VOMARootEntity
	Generalisierung: [VirtualOrganization].VOMAEntityName
	Aggregation: [TopLevel].VOMAEntity
	Generalisierung: [Member].IndividualIdentification
	Generalisierung: [Member].RealOrganizationIdentification
	Generalisierung: [Role].RoleIdentification
Attribute	public DateTime `issueDate`
	public DateTime `validUntil`
	public String `issuingAuthority`
	public int `issuingAuthorityQualifier`
	public int `trustLevel`
	public String `identitficationType`
Operationen	public `getValidTimeFrame():int`

public class *VOMAEntityName*	
Beschreibung	Abschnitt 5.2.3
Extends	VOMAEntityIdentification
Siehe auch	IndividualName, VOName, VOMAEntityIdentification, RoleName, RealOrganizationName, GroupName
Abhängigkeit	Generalisierung: [Member].IndividualName
	Generalisierung: [VirtualOrganization].VOName
	Generalisierung: [VirtualOrganization].VOMAEntityIdentification
	Generalisierung: [Role].RoleName
	Generalisierung: [Member].RealOrganizationName
	Generalisierung: [Member].GroupName
Attribute	public String `nameSuffix`
	public String `currentName`
	public String `originalName`
	public DateTime `currentNameValidSince`
	public boolean `isThisAlias`
Operationen	public `hasNameChanged():boolean`

public class *VOMALifecyclePhase*	
Beschreibung	Abschnitt 5.2.3
Extends	ManagedVOMAEntity
Siehe auch	SimpleEntityLifecycle, ManagedVOMAEntity
Abhängigkeit	Aggregation: [VirtualOrganization].SimpleEntityLifecycle
	Generalisierung: [TopLevel].ManagedVOMAEntity
Attribute	public OCL `phaseEntryCondition`
	public OCL `phaseExitCondition`
	public boolean `isErrorPhase`
Operationen	public `getNext():VOMALifecyclePhase[]`
	public `getPrevious():VOMALifecyclePhase[]`

public class *VOManager*	
Beschreibung	Abschnitt 5.2.3
Extends	virtualStaff
Siehe auch	virtualStaff
Abhängigkeit	Generalisierung: [VirtualOrganization].virtualStaff

public class *VOMARoleSpecification*	
Beschreibung	Abschnitt 5.2.3
Extends	VOMASpecification
Siehe auch	VOMASpecification, VOMAEntityRoles
Abhängigkeit	Generalisierung: [TopLevel].VOMASpecification
	Aggregation: [TopLevel].VOMAEntityRoles

public class *VOName*	
Beschreibung	Abschnitt 5.2.3

public class *VOName* (Fortsetzung)	
Extends	VOMAEntityName
Siehe auch	VOMAEntityName, VirtualOrganization
Abhängigkeit	Generalisierung: [VirtualOrganization].VOMAEntityName
	Aggregation: [VirtualOrganization].VirtualOrganization
Attribute	public String `nameChangedBy`

public class *VOParticipant*	
Beschreibung	Abschnitt 5.2.3
Constraints	homeInstitution MUST comply to RFC 1035
Extends	RoleInVO, GroupOfParticipants
Siehe auch	VOMember, RoleInVO, External, VOApplicants, GroupOfParticipants
Abhängigkeit	Generalisierung: [Member].VOMember
	Generalisierung: [Role].RoleInVO
	Generalisierung: [VirtualOrganization].External
	Assoziation: [VirtualOrganization].VOApplicants
	Generalisierung: [Member].GroupOfParticipants
	Assoziation: [Configuration Management].VOConfiguration
Attribute	public DateTime `acceptanceDate`
	public DateTime `approvalDate`
	public String `approvedBy`
	public DateTime `startDate`
	public String `homeInstitution`

public class *VOProvider*	
Beschreibung	Abschnitt 5.2.3
Extends	VOAsManager
Siehe auch	VOAsManager
Abhängigkeit	Generalisierung: [VirtualOrganization].VOAsManager

public class *VOSLAManager*	
Beschreibung	Abschnitt 5.2.3
Extends	virtualStaff
Siehe auch	virtualStaff
Abhängigkeit	Generalisierung: [VirtualOrganization].virtualStaff

public class *WebPageContact*	
Beschreibung	Abschnitt 5.2.3
Extends	ContactMethod
Siehe auch	ContactMethod
Abhängigkeit	Generalisierung: [VirtualOrganization].ContactMethod
Attribute	public URI `contactURI`

public class *WebPortal*	
Beschreibung	Abschnitt 5.2.3
Extends	CollaborationMethod
Siehe auch	CollaborationMethod
Abhängigkeit	Generalisierung: [VirtualOrganization].CollaborationMethod
Attribute	public URI `webPortalURI`
	public String `webPortalName`
	public String `webPortalWebMaster`
	public String `webPortalVersion`

public class *WikiPlatform*	
Beschreibung	Abschnitt 5.2.3
Extends	CollaborationMethod

public class *WikiPlatform* (Fortsetzung)	
Siehe auch	CollaborationMethod
Abhängigkeit	Generalisierung: [VirtualOrganization].CollaborationMethod
Attribute	public URI `wikiURI`
	public String `wikiAdministrator`
	public String `wikiName`
	public String `wikiVersion`

B.4 Klassen des Package `Management`

public class *ActivityNode*	
Beschreibung	Abschnitt 5.2.4
Extends	VOMAWorkflowNode
Siehe auch	VOMAWorkflowNode, RoleInVO
Abhängigkeit	Generalisierung: [Management].VOMAWorkflowNode
	Assoziation: [Role].RoleInVO
Attribute	public int `activityPriority`

public class *AtomicPolicyEvent*	
Beschreibung	Abschnitt 5.2.4
Extends	VOMAPolicyEvent
Siehe auch	VOMAPolicyEvent, VOSpecificEvent, RoleSpecificEvent, MemberSpecificEvent, VRSpecificEvent, VSSpecificEvent, LoggingSpecificEvent
Abhängigkeit	Generalisierung: [Management].VOMAPolicyEvent
	Generalisierung: [Management].VOSpecificEvent
	Generalisierung: [Management].RoleSpecificEvent
	Generalisierung: [Management].MemberSpecificEvent
	Generalisierung: [VirtualResource].VRSpecificEvent
	Generalisierung: [VirtualService].VSSpecificEvent

public class *AtomicPolicyEvent* (Fortsetzung)	
Generalisierung: [Management].LoggingSpecificEvent	

public class *AtomicPolicyEventSpec*	
Beschreibung	Abschnitt 5.2.4
Extends	VOMAPolicyEventSpecification
Siehe auch	VOSpecificEventSpec, RoleSpecificEventSpec, MemberSpecificEventSpec, VOMAPolicyEventSpecification, VRSpecificEventSpec, VSSpecificEventSpec, LoggingSpecificEventSpec
Abhängigkeit	Generalisierung: [Management].VOSpecificEventSpec Generalisierung: [Management].RoleSpecificEventSpec Generalisierung: [Management].MemberSpecificEventSpec Generalisierung: [Management].VOMAPolicyEventSpecification Generalisierung: [VirtualResource].VRSpecificEventSpec Generalisierung: [VirtualService].VSSpecificEventSpec Generalisierung: [Management].LoggingSpecificEventSpec

public class *CommonMemberScope*	
Beschreibung	Abschnitt 5.2.4
Extends	CommonVOMAManagementScope
Siehe auch	CommonVOMAManagementScope
Abhängigkeit	Generalisierung: [TopLevel].CommonVOMAManagementScope
Attribute	public String `specialMemberScopeCharacteristic`

public class *CommonResourceScope*	
Beschreibung	Abschnitt 5.2.4

public class *CommonResourceScope* (Fortsetzung)	
Extends	CommonVOMAManagementScope
Siehe auch	CommonVOMAManagementScope
Abhängigkeit	Generalisierung: [TopLevel].CommonVOMAManagementScope
Attribute	public String `specialResourceScopeCharacteristic`

public class *CommonRoleScope*	
Beschreibung	Abschnitt 5.2.4
Extends	CommonVOMAManagementScope
Siehe auch	CommonVOMAManagementScope
Abhängigkeit	Generalisierung: [TopLevel].CommonVOMAManagementScope
Attribute	public String `specialRoleScopeCharacteristic`

public class *CommonServiceScope*	
Beschreibung	Abschnitt 5.2.4
Extends	CommonVOMAManagementScope
Siehe auch	CommonVOMAManagementScope
Abhängigkeit	Generalisierung: [TopLevel].CommonVOMAManagementScope
Attribute	public String `specialServiceScopeCharacteristic`

public class *CommonVOMAPolicyScope*	
Beschreibung	Abschnitt 5.2.4
Extends	CommonVOMAManagementScope
Siehe auch	CommonVOMAManagementScope, VOMAPolicy, VOMAPolicyRepository
Abhängigkeit	Generalisierung: [TopLevel].CommonVOMAManagementScope

public class *CommonVOMAPolicyScope* (Fortsetzung)	
	Assoziation: [Management].VOMAPolicy Assoziation: [Management].VOMAPolicyRepository

public class *CommonVOScope*	
Beschreibung	Abschnitt 5.2.4
Extends	CommonVOMAManagementScope
Siehe auch	CommonVOMAManagementScope
Abhängigkeit	Generalisierung: [TopLevel].CommonVOMAManagementScope
Attribute	public String `specialVOScopeCharacteristic`

public class *CompositePolicyEvent*	
Beschreibung	Abschnitt 5.2.4
Extends	VOMAPolicyEvent
Siehe auch	VOMAPolicyEvent, VOMAPolicyEvent
	Aggregation: [Management].VOMAPolicyEvent

public class *CompositePolicyEventSpec*	
Beschreibung	Abschnitt 5.2.4
Extends	VOMAPolicyEventSpecification
Siehe auch	VOMAPolicyEventSpecification, VOMAPolicyEventSpecification
Abhängigkeit	Generalisierung: [Management].VOMAPolicyEventSpecification
	Aggregation: [Management].VOMAPolicyEventSpecification

public class *ControlNode*	
Beschreibung	Abschnitt 5.2.4
Extends	VOMAWorkflowNode
Siehe auch	VOMAWorkflowNode
Abhängigkeit	Generalisierung: [Management].VOMAWorkflowNode
Attribute	public int `controlType`

public class *LocalManagementSystem*	
Beschreibung	Abschnitt 5.2.4
Extends	UnManagedVOMAEntity
Siehe auch	VOMASite, VOManagementInteraction, VOMAPolicyRepository, VOMAPolicy, UnManagedVOMAEntity, RealOrganization, RealOrgResource, RealOrgService
Abhängigkeit	Aggregation: [TopLevel].VOMASite Assoziation: [Management].VOManagementInteraction Assoziation: [Management].VOMAPolicyRepository Assoziation: [Management].VOMAPolicy Generalisierung: [TopLevel].UnManagedVOMAEntity Aggregation: [TopLevel].RealOrganization Aggregation: [TopLevel].RealOrgResource Aggregation: [TopLevel].RealOrgService

public class *LoggingSpecificEvent*	
Beschreibung	Abschnitt 5.2.4
Extends	AtomicPolicyEvent
Siehe auch	AtomicPolicyEvent, LoggingSpecificEventSpec, VOMALoggingRecord
Abhängigkeit	Generalisierung: [Management].AtomicPolicyEvent Aggregation: [Management].LoggingSpecificEventSpec Aggregation: [Management].VOMALoggingRecord Assoziation: [VOMA Communication Model].VOMALoggingService Generalisierung: [Accounting Management].VOUsageRecord

public class *LoggingSpecificEventSpec*	
Beschreibung	Abschnitt 5.2.4
Extends	AtomicPolicyEventSpec
Siehe auch	AtomicPolicyEventSpec, LoggingSpecificEvent
Abhängigkeit	Generalisierung: [Management].AtomicPolicyEventSpec Aggregation: [Management].LoggingSpecificEvent

public class *MemberAsTarget*	
Beschreibung	Abschnitt 5.2.4
Extends	VOMAPolicyTarget
Siehe auch	VOMAPolicyTarget, VOMember
Abhängigkeit	Generalisierung: [Management].VOMAPolicyTarget Aggregation: [Member].VOMember

public class *MemberSpecificEvent*	
Beschreibung	Abschnitt 5.2.4
Extends	AtomicPolicyEvent
Siehe auch	AtomicPolicyEvent, IndividualAppliesforMembershipDetails, ApplicationIsAcceptedDetails, OrganizationAppliesForMembershipDetails, MemberSpecificEventSpec
Abhängigkeit	Generalisierung: [Management].AtomicPolicyEvent Generalisierung: [Member].IndividualAppliesforMembershipDetails Generalisierung: [Member].ApplicationIsAcceptedDetails Generalisierung: [Member].OrganizationAppliesForMembershipDetails Aggregation: [Management].MemberSpecificEventSpec

public class *MemberSpecificEventSpec*	
Beschreibung	Abschnitt 5.2.4
Extends	AtomicPolicyEventSpec
Siehe auch	AtomicPolicyEventSpec, MemberSpecificEvent
Abhängigkeit	Generalisierung: [Management].AtomicPolicyEventSpec
	Aggregation: [Management].MemberSpecificEvent

public class *RoleAsTarget*	
Beschreibung	Abschnitt 5.2.4
Extends	VOMAPolicyTarget
Siehe auch	VOMAPolicyTarget, RoleInVO
Abhängigkeit	Generalisierung: [Management].VOMAPolicyTarget
	Aggregation: [Role].RoleInVO
Attribute	class: RoleAsTarget

public class *RoleSpecificEvent*	
Beschreibung	Abschnitt 5.2.4
Extends	AtomicPolicyEvent
Siehe auch	AtomicPolicyEvent, RoleSpecificEventSpec
Abhängigkeit	Generalisierung: [Management].AtomicPolicyEvent
	Aggregation: [Management].RoleSpecificEventSpec
Attribute	class: RoleSpecificEvent

public class *RoleSpecificEventSpec*	
Beschreibung	Abschnitt 5.2.4
Extends	AtomicPolicyEventSpec

public class *RoleSpecificEventSpec* (Fortsetzung)	
Siehe auch	AtomicPolicyEventSpec, RoleSpecificEvent
Abhängigkeit	Generalisierung: [Management].AtomicPolicyEventSpec
	Aggregation: [Management].RoleSpecificEvent

public class *SiteInvolvementDetails*	
Beschreibung	Abschnitt 5.2.4
Siehe auch	VOManagementInteractionItem
Abhängigkeit	Assoziation: [Management].VOManagementInteractionItem

public class *VOAsTarget*	
Beschreibung	Abschnitt 5.2.4
Extends	VOMAPolicyTarget
Siehe auch	VOMAPolicyTarget, VirtualOrganization
Abhängigkeit	Generalisierung: [Management].VOMAPolicyTarget
	Aggregation: [VirtualOrganization].VirtualOrganization

public class *VOMAAgent*	
Beschreibung	Abschnitt 5.2.4
Extends	VOMAEntityRoles
Siehe auch	VOMAEntityRoles, VOMAPolicyTarget
Abhängigkeit	Generalisierung: [TopLevel].VOMAEntityRoles
	Generalisierung: [Management].VOMAPolicyTarget

public class *VOMAArchive*	
Beschreibung	Abschnitt 5.2.4
Extends	VOMACollection
Siehe auch	VOMACollection, VOMALoggingRecord, VOMASite, VOArchiveManager
Abhängigkeit	Generalisierung: [TopLevel].VOMACollection Aggregation: [Management].VOMALoggingRecord Aggregation: [TopLevel].VOMASite Assoziation: [VirtualOrganization].VOArchiveManager

public class *VOMAContract*	
Beschreibung	Abschnitt 5.2.4
Extends	VOManagementInteraction
Siehe auch	VOManagementInteraction, VOMASpecificSLA, VOMAContractItem
Abhängigkeit	Generalisierung: [Management].VOManagementInteraction Generalisierung: [Management].VOMASpecificSLA Aggregation: [Management].VOMAContractItem
Attribute	public String `contractNumber` public TimePeriod `validFor` public String `contractObjectives` public DateTime `approvalDate` public String `approvedBy` public DateTime `contractInceptionDate`
Operationen	public `getContractTimeToLive():DateTime`

public class *VOMAContractItem*	
Beschreibung	Abschnitt 5.2.4
Extends	VOManagementInteractionItem
Siehe auch	VOManagementInteractionItem, VOMAContract
Abhängigkeit	Generalisierung: [Management].VOManagementInteractionItem Aggregation: [Management].VOMAContract

public class *VOMALoggingRecord*	
Beschreibung	Abschnitt 5.2.4
Extends	VOMACollection
Siehe auch	VOMACollection, LoggingSpecificEvent, VOMAArchive
Abhängigkeit	Generalisierung: [TopLevel].VOMACollection Aggregation: [Management].LoggingSpecificEvent Aggregation: [Management].VOMAArchive

public class *VOMAManagingRole*	
Beschreibung	Abschnitt 5.2.4
Extends	VOMAEntityRoles
Siehe auch	virtualStaff, VOAsManager, VOMAEntityRoles
Abhängigkeit	Generalisierung: [VirtualOrganization].virtualStaff Generalisierung: [VirtualOrganization].VOAsManager Generalisierung: [TopLevel].VOMAEntityRoles

public class *VOManagementInteraction*	
Beschreibung	Abschnitt 5.2.4
Extends	
Siehe auch	LocalManagementSystem, VOMAPolicyRepository, VOMAPolicy, VOMAContract, VOManagementInteractionItem, VOMASite, VOMAOperationRequest, VOMAResponse, VOMANotification
Abhängigkeit	Assoziation: [Management].LocalManagementSystem Assoziation: [Management].VOMAPolicyRepository Assoziation: [Management].VOMAPolicy Generalisierung: [Management].VOMAContract Aggregation: [Management].VOManagementInteractionItem Assoziation: [TopLevel].VOMASite

public class *VOManagementInteraction* (Fortsetzung)	
	Generalisierung: [Management].VOMAOperationRequest
	Generalisierung: [Management].VOMAResponse
	Generalisierung: [Management].VOMANotification
Attribute	public int `interactionID`
	public String `interactionDescription`
	public DateTime `initiationDate`
	public DateTime `plannedCompletionDate`
	public DateTime `projectedCompletionDate`
	public int `interactionStatus`
	`public int` interactionFocusType

public class *VOManagementInteractionItem*	
Beschreibung	Abschnitt 5.2.4
Extends	VOMAEntity
Siehe auch	VOMAContractItem, VOMAEntity, VOManagementInteraction, VOMAEntityRoles, SiteInvolvementDetails
Abhängigkeit	Generalisierung: [Management].VOMAContractItem
	Generalisierung: [TopLevel].VOMAEntity
	Aggregation: [Management].VOManagementInteraction
	Assoziation: [TopLevel].VOMAEntityRoles
	Assoziation: [Management].SiteInvolvementDetails

public class *VOMANotification*	
Beschreibung	Abschnitt 5.2.4
Extends	VOManagementInteraction
Siehe auch	VOManagementInteraction
Abhängigkeit	Generalisierung: [Management].VOManagementInteraction

public class *VOMAOperationRequest*	
Beschreibung	Abschnitt 5.2.4
Extends	VOManagementInteraction
Siehe auch	VOManagementInteraction
Abhängigkeit	Generalisierung: [Management].VOManagementInteraction

public class *VOMAPolicy*	
Beschreibung	Abschnitt 5.2.4
Extends	ManagedVOMAEntity
Siehe auch	ManagedVOMAEntity, VOMAPolicyApplication, CommonVOMAPolicyScope, VOMAPolicySpecification, VOManagementInteraction, VOMAPolicyRepository, LocalManagementSystem, VOMAWorkflow
Abhängigkeit	Assoziation: [Configuration Management].VOConfiguration Generalisierung: [TopLevel].ManagedVOMAEntity Assoziation: [Management].VOMAPolicyApplication Assoziation: [Management].CommonVOMAPolicyScope Aggregation: [Management].VOMAPolicySpecification Assoziation: [Management].VOManagementInteraction Assoziation: [Management].VOMAPolicyRepository Assoziation: [Management].LocalManagementSystem Generalisierung: [Management].VOMAWorkflow
Attribute	public String `vomaPolicyName` public String[] `vomaPoliciyTypes` public TimePeriod `validFor`

public class *VOMAPolicyApplication*	
Beschreibung	Abschnitt 5.2.4
Extends	ManagedVOMAEntity
Siehe auch	VOMAPolicyTarget, VOMAPolicy, UnManagedVOMAEntity, ManagedVOMAEntity
Abhängigkeit	Assoziation: [Management].VOMAPolicyTarget

public class *VOMAPolicyApplication* (Fortsetzung)	
	Assoziation: [Management].VOMAPolicy Assoziation: [TopLevel].UnManagedVOMAEntity Generalisierung: [TopLevel].ManagedVOMAEntity

public class *VOMAPolicyEvent*	
Beschreibung	Abschnitt 5.2.4
Extends	VOMACollection
Siehe auch	VOMACollection, AtomicPolicyEvent, CompositePolicyEvent, CompositePolicyEvent
Abhängigkeit	Generalisierung: [TopLevel].VOMACollection Generalisierung: [Management].AtomicPolicyEvent Generalisierung: [Management].CompositePolicyEvent Aggregation: [Management].CompositePolicyEvent
Attribute	public int `evaluationStatus`
Operationen	public `pullEvents():void`

public class *VOMAPolicyEventSpecification*	
Beschreibung	Abschnitt 5.2.4
Extends	VOMAPolicySpecification
Siehe auch	VOMAPolicySpecification, AtomicPolicyEventSpec, CompositePolicyEventSpec, CompositePolicyEventSpec
Abhängigkeit	Generalisierung: [Management].VOMAPolicySpecification Generalisierung: [Management].AtomicPolicyEventSpec Generalisierung: [Management].CompositePolicyEventSpec Aggregation: [Management].CompositePolicyEventSpec

public class *VOMAPolicyRepository*	
Beschreibung	Abschnitt 5.2.4
Extends	VOMACollection
Siehe auch	CommonVOMAPolicyScope, VOMASite, VOMACollection, VOManagementInteraction, VOMAPolicy, LocalManagementSystem
Abhängigkeit	Assoziation: [Management].CommonVOMAPolicyScope Aggregation: [TopLevel].VOMASite Generalisierung: [TopLevel].VOMACollection Assoziation: [Management].VOManagementInteraction Assoziation: [Management].VOMAPolicy Assoziation: [Management].LocalManagementSystem
Attribute	public String `repositoryName` public URI `repositoryAddress` public DateTime `creationDate` public TimePeriod `validFor` public String `repositoryContact` public String `repositoryOwner`

public class *VOMAPolicyRuleSpecification*	
Beschreibung	Abschnitt 5.2.4
Extends	VOMAPolicySpecification
Siehe auch	VOMAPolicySpecification
Abhängigkeit	Generalisierung: [Management].VOMAPolicySpecification
Attribute	public int `ruleActionOrdering` public OCL `ruleExecutionStrategy`

public class *VOMAPolicySpecification*	
Beschreibung	Abschnitt 5.2.4
Extends	VOMASpecification
Siehe auch	VOMAPolicy, VOMASpecification, VOMAPolicyRuleSpecification, VOMAPolicyEventSpecification

public class *VOMAPolicySpecification* (Fortsetzung)	
Abhängigkeit	Aggregation: [Management].VOMAPolicy Generalisierung: [TopLevel].VOMASpecification Generalisierung: [Management].VOMAPolicyRuleSpecification Generalisierung: [Management].VOMAPolicyEventSpecification

public class *VOMAPolicyTarget*	
Beschreibung	Abschnitt 5.2.4
Extends	VOMAEntity, VOMAAgent
Siehe auch	VOMAPolicyApplication, VOAsTarget, VOMAEntity, RoleAsTarget, MemberAsTarget, VOMAAgent
Abhängigkeit	Assoziation: [Management].VOMAPolicyApplication Generalisierung: [Management].VOAsTarget Generalisierung: [TopLevel].VOMAEntity Generalisierung: [Management].RoleAsTarget Generalisierung: [Management].MemberAsTarget Generalisierung: [Management].VOMAAgent

public class *VOMAResponse*	
Beschreibung	Abschnitt 5.2.4
Extends	VOManagementInteraction
Siehe auch	VOManagementInteraction
Abhängigkeit	Generalisierung: [Management].VOManagementInteraction

public class *VOMASpecificSLA*	
Beschreibung	Abschnitt 5.2.4

public class *VOMASpecificSLA* (Fortsetzung)	
Extends	VOMAContract, VOMASpecification
Siehe auch	VOMAContract, VOMASpecification
Abhängigkeit	Generalisierung: [Management].VOMAContract
	Generalisierung: [TopLevel].VOMASpecification
	Assoziation: [Configuration Management].VOConfiguration
	Aggregation: [Accounting Management].VOUsageRecord

public class *VOMAWorkflow*	
Beschreibung	Abschnitt 5.2.4
Extends	VOMASpecification, VOMAPolicy
Siehe auch	VOMASpecification, VOMAWorkflowNode, VOMAWorkflowTransition, VOMAPolicy
Abhängigkeit	Generalisierung: [TopLevel].VOMASpecification
	Aggregation: [Management].VOMAWorkflowNode
	Aggregation: [Management].VOMAWorkflowTransition
	Generalisierung: [Management].VOMAPolicy

public class *VOMAWorkflowNode*	
Beschreibung	Abschnitt 5.2.4
Extends	VOMAWorkflow, ControlNode, ActivityNode, VOMAWorkflowTransition
Siehe auch	VOMA Workflows
Abhängigkeit	Generalisierung: [Management].ActivityNode
	Assoziation: [Management].VOMAWorkflowTransition
Attribute	public String `nodeName`

public class *VOMAWorkflowTransition*	
Beschreibung	Abschnitt 5.2.4
Extends	
Siehe auch	VOMAWorkflow, VOMAWorkflowNode
Abhängigkeit	Aggregation: [Management].VOMAWorkflow
	Assoziation: [Management].VOMAWorkflowNode

public class *VOSpecificEvent*	
Beschreibung	Abschnitt 5.2.4
Extends	AtomicPolicyEvent
Siehe auch	AtomicPolicyEvent, VOSpecificEventSpec
Abhängigkeit	Generalisierung: [Management].AtomicPolicyEvent
	Aggregation: [Management].VOSpecificEventSpec

public class *VOSpecificEventSpec*	
Beschreibung	Abschnitt 5.2.4
Extends	AtomicPolicyEventSpec
Siehe auch	AtomicPolicyEventSpec, VOSpecificEvent
Abhängigkeit	Generalisierung: [Management].AtomicPolicyEventSpec
	Aggregation: [Management].VOSpecificEvent

B.5 Klassen des Package `Member`

public class *AccountPassword*	
Beschreibung	Abschnitt 5.2.5
Extends	IndividualIdentification
Siehe auch	IndividualIdentification
Abhängigkeit	Generalisierung: [Member].IndividualIdentification

public class *AccountPassword*	
Beschreibung	Abschnitt 5.2.5
Extends	IndividualIdentification
Siehe auch	IndividualIdentification
Abhängigkeit	Generalisierung: [Member].IndividualIdentification
Attribute	public Template `approvedParticipationScope`
	public String `approvedInitialRole`

public class *AuthorizationDetails*	
Beschreibung	Abschnitt 5.2.5
Extends	
Siehe auch	
Attribute	public boolean `organizationSpecificationMatches`

public class *CertificateOfAuthority*	
Beschreibung	Abschnitt 5.2.5
Extends	RealOrganizationIdentification

public class *CertificateOfAuthority* (Fortsetzung)	
Siehe auch	RealOrganizationIdentification, TradeRegisterCertificate, TrustedAuthorities
Abhängigkeit	Generalisierung: [Member].RealOrganizationIdentification
	Assoziation: [Member].TradeRegisterCertificate
	Assoziation: [Trusted Entities].TrustedAuthorities
Attribute	public String `certificateHolder`

public class *EmployeeIdentification*	
Beschreibung	Abschnitt 5.2.5
Extends	IndividualIdentification
Siehe auch	IndividualIdentification
Abhängigkeit	Generalisierung: [Member].IndividualIdentification
Attribute	public String `employeeNumber`
	public String `employeeStatus`

public class *Group*	
Beschreibung	Abschnitt 5.2.5
Extends	GroupOfParticipants
Siehe auch	GroupOfParticipants, GroupOfParticipants, GroupName
Abhängigkeit	Generalisierung: [Member].GroupOfParticipants
	Aggregation: [Member].GroupOfParticipants
	Aggregation: [Member].GroupName
Attribute	public TimePerios `validFor`
	public DateTime `creationDate`

public class *GroupName*	
Beschreibung	Abschnitt 5.2.5
Extends	VOMAEntityName
Siehe auch	VOMAEntityName, Group
Abhängigkeit	Generalisierung: [VirtualOrganization].VOMAEntityName
	Aggregation: [Member].Group
Attribute	public String `groupNameChangedBy`

public class *GroupOfParticipants*	
Beschreibung	Abschnitt 5.2.5
Extends	ManagedVOMAEntity
Siehe auch	ManagedVOMAEntity, Group, VOParticipant, Group
Abhängigkeit	Generalisierung: [TopLevel].ManagedVOMAEntity
	Generalisierung: [Member].Group
	Generalisierung: [VirtualOrganization].VOParticipant
	Aggregation: [Member].Group
Attribute	public TimePeriod `validFor`
	public DateTime `creationDate`
	public String `groupSpokesman`
Operationen	public `calculateMaxTimeToLive():TimePeriod`
	public `calculateMinTimeToLive():TimePeriod`

public class *IdentityCard*	
Beschreibung	Abschnitt 5.2.5
Extends	IndividualIdentification
Siehe auch	IndividualIdentification
Abhängigkeit	Generalisierung: [Member].IndividualIdentification
Attribute	public String `cardID`
	private int `cardType`

public class *IndividualAppliesforMembershipDetails*	
Beschreibung	Abschnitt 5.2.5
Extends	MemberSpecificEvent
Siehe auch	MemberSpecificEvent
Abhängigkeit	Generalisierung: [Management].MemberSpecificEvent
Attribute	ublic DateTime `applicationRequestDate`
	public Template `applicationRequestScope`
	public String `applicationRequestJustification`
	public DateTime `membershipRequestDate`

public class *IndividualIdentification*	
Beschreibung	Abschnitt 5.2.5
Extends	VOMAEntityIdentification
Siehe auch	VOMAEntityIdentification, IdentityCard, EmployeeIdentification, Individual, PublicKeyCertificate, AccountPassword
Abhängigkeit	Generalisierung: [VirtualOrganization].VOMAEntityIdentification
	Generalisierung: [Member].IdentityCard
	Generalisierung: [Member].EmployeeIdentification
	Aggregation: [TopLevel].Individual
	Generalisierung: [Member].PublicKeyCertificate
	Generalisierung: [Member].AccountPassword

public class *IndividualName*	
Beschreibung	Abschnitt 5.2.5
Extends	VOMAEntityName
Siehe auch	VOMAEntityName, Individual
Abhängigkeit	Generalisierung: [VirtualOrganization].VOMAEntityName
	Aggregation: [TopLevel].Individual
Attribute	public String `nickName`
	public String `birthName`

public class *OrganizationAppliesForMembershipDetails*	
Beschreibung	Abschnitt 5.2.5
Extends	MemberSpecificEvent
Siehe auch	MemberSpecificEvent
Abhängigkeit	Generalisierung: [Management].MemberSpecificEvent
Attribute	public DateTime `applicationRequestDate`
	public Template `applicationRequestScope`
	public String `applicationRequestJustification`
	public DateTime `membershipRequestDate`
Operationen	public `createMembershipForAllOrgMembers():String`

public class *PublicKeyCertificate*	
Beschreibung	Abschnitt 5.2.5
Constraint	RFC 3281
Extends	IndividualIdentification
Siehe auch	IndividualIdentification
Abhängigkeit	Generalisierung: [Member].IndividualIdentification

public class *RealOrganizationIdentification*	
Beschreibung	Abschnitt 5.2.5
Extends	VOMAEntityIdentification
Siehe auch	RealOrganization, VOMAEntityIdentification, CertificateOfAuthority, TradeRegisterCertificate, TrustedAuthorities
Abhängigkeit	Aggregation: [TopLevel].RealOrganization
	Generalisierung: [VirtualOrganization].VOMAEntityIdentification
	Generalisierung: [Member].CertificateOfAuthority
	Generalisierung: [Member].TradeRegisterCertificate
	Assoziation: [Trusted Entities].TrustedAuthorities

public class *RealOrganizationIdentification* (Fortsetzung)		
Attribute	public String	`identificationNumber`
	public String	`registeredAt`
	public String[]	`originalRepresentatives`
	public String[]	`currentRepresentatives`

public class *RealOrganizationName*	
Beschreibung	Abschnitt 5.2.5
Extends	VOMAEntityName
Siehe auch	VOMAEntityName, RealOrganization
Abhängigkeit	Generalisierung: [VirtualOrganization].VOMAEntityName
	Aggregation: [TopLevel].RealOrganization
Attribute	public String `nameChangedBy`

public class *TradeRegisterCertificate*	
Beschreibung	Abschnitt 5.2.5
Extends	RealOrganizationIdentification
Siehe auch	RealOrganizationIdentification, CertificateOfAuthority
Abhängigkeit	Generalisierung: [Member].RealOrganizationIdentification
	Assoziation: [Member].CertificateOfAuthority

public class *VOMember*	
Beschreibung	Abschnitt 5.2.5
Extends	ManagedVOMAEntity, VOParticipant
Siehe auch	ManagedVOMAEntity, VOParticipant, OrdinaryMember, virtualStaff, MemberAsTarget, VOAsMember

public class *VOMember* (Fortsetzung)	
Abhängigkeit	Generalisierung: [TopLevel].ManagedVOMAEntity
	Generalisierung: [VirtualOrganization].VOParticipant
	Generalisierung: [VirtualOrganization].OrdinaryMember
	Generalisierung: [VirtualOrganization].virtualStaff
	Aggregation: [Management].MemberAsTarget
	Generalisierung: [VirtualOrganization].VOAsMember
Attribute	public int `memberType`
	public DateTime `memberSince`
	public DateTime `memberUntil`
	public String[] `assignedRoles`
	public DateTime `membershipApprovalDate`
	public String `membershipApprovalBy`

B.6 Klassen des Package `Role`

public class *RoleIdentification*	
Beschreibung	Abschnitt 5.2.6
Extends	VOMAEntityIdentification
Siehe auch	RoleInVO, VOMAEntityIdentification
Abhängigkeit	Aggregation: [Role].RoleInVO
	Generalisierung: [VirtualOrganization].VOMAEntityIdentification

public class *RoleInVO*	
Beschreibung	Abschnitt 5.2.6
Extends	ManagedVOMAEntity, VOMAEntityRoles
Siehe auch	ManagedVOMAEntity, RoleName, VirtualOrganization, VOMAEntity-Roles, VOParticipant, VOApplicants, RoleAsTarget, RoleIdentification, VOMACapabilitiesSpecification, ActivityNode

public class *RoleInVO* (Fortsetzung)	
Abhängigkeit	Generalisierung: [TopLevel].ManagedVOMAEntity Aggregation: [Role].RoleName Aggregation: [VirtualOrganization].VirtualOrganization Generalisierung: [TopLevel].VOMAEntityRoles Generalisierung: [VirtualOrganization].VOParticipant Generalisierung: [VirtualOrganization].VOApplicants Aggregation: [Management].RoleAsTarget Aggregation: [Role].RoleIdentification Aggregation: [Role].VOMACapabilitiesSpecification Assoziation: [Management].ActivityNode Generalisierung: [Deployment Patterns].CommunitySprecher Generalisierung: [Deployment Patterns].VOAdministrator Generalisierung: [Deployment Patterns].Benutzer
Attribute	public DateTime `roleCreationDate` public TimePeriod `validFor` public boolean `isAssigned` public String[] `assignedMembers`

public class *RoleName*	
Beschreibung	Abschnitt 5.2.6
Extends	VOMAEntityName
Siehe auch	VOMAEntityName, RoleInVO
Abhängigkeit	Generalisierung: [VirtualOrganization].VOMAEntityName Aggregation: [Role].RoleInVO
Attribute	

public class *VOMACapabilitiesSpecification*	
Beschreibung	Abschnitt 5.2.6
Extends	VOMASpecification
Siehe auch	VOMASpecification, RoleInVO

public class *VOMACapabilitiesSpecification* (Fortsetzung)	
Abhängigkeit	Generalisierung: [TopLevel].VOMASpecification Aggregation: [Role].RoleInVO

B.7 Klassen des Package `VirtualResource`

public class *Binding*	
Beschreibung	Abschnitt 5.2.7
Extends	
Siehe auch	
Attribute	public Path `mountPoint` public int `priority`

public class *ComputingResource*	
Beschreibung	Abschnitt 5.2.7
Extends	VirtualResource
Siehe auch	VirtualResource, StorageResource
Abhängigkeit	Generalisierung: [VirtualResource].VirtualResource Assoziation: [VirtualResource].StorageResource
Attribute	public String[] `resourceCapabilities` public String[] `resourceContact` public Path `applicationDirectory` public Path `dataDirectory` public StorageResource `defaultStorageResource` public int[] `jobsInQueues`

public class *LogicalRealResource*	
Beschreibung	Abschnitt 5.2.7
Extends	RealOrgResource
Siehe auch	RealOrgResource
Abhängigkeit	Generalisierung: [TopLevel].RealOrgResource

public class *PhysicalRealResource*	
Beschreibung	Abschnitt 5.2.7
Extends	RealOrgResource
Siehe auch	RealOrgResource
Abhängigkeit	Generalisierung: [TopLevel].RealOrgResource

public class *StorageResource*	
Beschreibung	Abschnitt 5.2.7
Extends	VirtualResource
Siehe auch	VirtualResource, ComputingResource
Abhängigkeit	Generalisierung: [VirtualResource].VirtualResource
	Assoziation: [VirtualResource].ComputingResource
Attribute	public int `storageMediaType`
	public long `totalCapacity`
	public long `freeCapacity`

public class *VirtualResource*	
Beschreibung	Abschnitt 5.2.7
Extends	ManagedVOMAEntity
Siehe auch	ComputingResource, StorageResource, VirtualOrganization, VRSpecificEvent, RealOrgResource, VOMASite, ManagedVOMAEntity

public class *VirtualResource* (Fortsetzung)	
Abhängigkeit	Generalisierung: [VirtualResource].ComputingResource
	Generalisierung: [VirtualResource].StorageResource
	Aggregation: [VirtualOrganization].VirtualOrganization
	Aggregation: [VirtualResource].VRSpecificEvent
	Aggregation: [TopLevel].RealOrgResource
	Aggregation: [TopLevel].VOMASite
	Generalisierung: [TopLevel].ManagedVOMAEntity
	Assoziation: [Configuration Management].VOConfiguration
Attribute	public String `virtualResourceName`
	public String `resourceDescription`
	public DateTime `creationDate`
	public TimePeriod `validFor`
	public URI `informationServiceLink`
	public int `resourceQueueStatus`

public class *VRSpecificEvent*	
Beschreibung	Abschnitt 5.2.7
Extends	AtomicPolicyEvent
Siehe auch	AtomicPolicyEvent, VirtualResource, VRSpecificEventSpec
Abhängigkeit	Generalisierung: [Management].AtomicPolicyEvent
	Aggregation: [VirtualResource].VirtualResource
	Aggregation: [VirtualResource].VRSpecificEventSpec

public class *VRSpecificEventSpec*	
Beschreibung	Abschnitt 5.2.7
Extends	AtomicPolicyEventSpec
Siehe auch	AtomicPolicyEventSpec, VRSpecificEvent
Abhängigkeit	Generalisierung: [Management].AtomicPolicyEventSpec
	Aggregation: [VirtualResource].VRSpecificEvent

B.8 Klassen des Package `VirtualService`

public class *KeyValuePairs*	
Beschreibung	Abschnitt 5.2.8
Extends	
Siehe auch	VirtualService
Abhängigkeit	Aggregation: [VirtualService].VirtualService
Attribute	public String `Key`
	public String `Value`

public class *VirtualService*	
Beschreibung	Abschnitt 5.2.8
Extends	ManagedVOMAEntity
Siehe auch	VirtualOrganization, RealOrgService, VSSpecificEvent, KeyValuePairs, VOMASite, ManagedVOMAEntity
Abhängigkeit	Aggregation: [VirtualOrganization].VirtualOrganization
	Aggregation: [TopLevel].RealOrgService
	Aggregation: [VirtualService].VirtualService
	Aggregation: [VirtualService].VSSpecificEvent
	Aggregation: [VirtualService].KeyValuePairs
	Aggregation: [TopLevel].VOMASite
	Generalisierung: [TopLevel].ManagedVOMAEntity
	Assoziation: [Configuration Management].VOConfiguration
Attribute	public String `serviceName`
	public int `serviceType`
	public DateTime `creationDate`
	public TimePeriod `validFor`
	public String `serviceVersion`
	public int `serviceStatus`
	public DateTime `lastStart`
	public String[] `serviceOwner`
	public String `serviceDescription`

public class *VSSpecificEvent*	
Beschreibung	Abschnitt 5.2.8
Extends	AtomicPolicyEvent
Siehe auch	VirtualService, AtomicPolicyEvent, VSSpecificEventSpec
Abhängigkeit	Aggregation: [VirtualService].VirtualService Generalisierung: [Management].AtomicPolicyEvent Aggregation: [VirtualService].VSSpecificEventSpec

public class *VSSpecificEventSpec*	
Beschreibung	Abschnitt 5.2.8
Extends	AtomicPolicyEventSpec
Siehe auch	AtomicPolicyEventSpec, VSSpecificEvent
Abhängigkeit	Generalisierung: [Management].AtomicPolicyEventSpec Aggregation: [VirtualService].VSSpecificEvent

B.9 Klassen des Package `TrustedEntities`

public class *Escrow*	
Beschreibung	Abschnitt 5.2.9
Extends	TrustedAuthorities
Siehe auch	TrustedAuthorities
Abhängigkeit	Generalisierung: [Trusted Entities].TrustedAuthorities

public class *Notary*	
Beschreibung	Abschnitt 5.2.9
Extends	TrustedAuthorities
Siehe auch	TrustedAuthorities
Abhängigkeit	Generalisierung: [Trusted Entities].TrustedAuthorities

public class *TrustedAuthorities*	
Beschreibung	Abschnitt 5.2.9
Extends	
Siehe auch	VOIdentification, Notary, Community, Escrow, RealOrganizationIdentification, CertificateOfAuthority
Abhängigkeit	Assoziation: [VirtualOrganization].VOIdentification Generalisierung: [Trusted Entities].Notary Assoziation: [TopLevel].Community Generalisierung: [Trusted Entities].Escrow Assoziation: [Member].RealOrganizationIdentification Assoziation: [Member].CertificateOfAuthority
Attribute	protected String `name` protected String[] `address` protected String `type` protected DateTime `creationDate` public int `trustLevel` public int `trustLevelVisibility`

Abbildungsverzeichnis

Tabellenverzeichnis

LITERATUR

Abarca u. a. 1997
Abarca, C. ; Farley, P. ; Forslöw, J. ; García, J. C. ; Hamada, T. ; Hansen, P. F. ; Hogg, S. ; Kamata, H. ; Kristiansen, L. ; Licciardi, C. A. ; Mulder, H. ; Utsunomiya, E. ; Yates, M.: Service Architecture Verison 5.0 / TINA Consortium. 1997. – Forschungsbericht

AKT e-Response Team 2007
AKT e-Response Team: The Application of Advanced Knowledge Technologies for Emergency Response. In: Walle, B. V. (Hrsg.) ; Burghardt, P. (Hrsg.) ; Nieuwenhuis, C. (Hrsg.): *Proceedings of the 4th International ISCRAM Conference*. Delft, Niederlande, Mai 2007

Alfieri u. a. 2003
Alfieri, R. ; Cecchini, R. ; Ciaschini, V. ; dell'Agnello, L. ; Gianoli, A. ; Spataro, F. ; Bonnassieux, F. ; Broadfoot, P. ; Lowe, G. ; Cornwall, L. ; Jensen, J. ; Kelsey, D. ; Frohne, Á. ; Groep, D.L. ; Cerff, W. S. ; Steenbakkers, M. ; Venekamp, G. ; Kouril, D. ; McNab, A. ; Mulmo, O. ; Silander, M. ; Hahkala, J. ; Lörentey, K.: Managing Dynamic User Communities in a Grid of Autonomous Resources. In: *Proceedings Computing in High Energy and Nuclear Physics (CHEP 2003)* Bd. C0303241. La Jolla, USA : ECONF, März 2003. – `http://www.slac.stanford.edu/econf/C0303241/proceedings.html`

Allison u. a. 2007
Allison, Ian ; Béland, Michel ; Alverson, Keith ; Bell, Robin ; Carlson, David ; Danell, Kjell ; Ellis-Evans, Cynan ; Fahrbach, Eberhard ; Fanta, Edith ; Fujii, Yoshiyuki ; Glaser, Gisbert ; Goldfarb, Leah ; Hovelsrud, Grete ; Huber, Johannes ; Kotlyakov, Vladimir ; Krupnik, Igor ; Lopez-Martinez, Jeronimo ; Mohr, Tillmann ; Qin, Dahe ; Rachold, Volker ; Rapley, Chris ; Rogne, Odd ; Sarukhanian, Eduard ; Summerhayes, Colin ; Xiao, Cunde: *The Scope of Science for the International Polar Year 2007-2008*. Version: Februar 2007. `http://216.70.123.96/images/uploads/LR*PolarBrochureScientific_IN.pdf`. Report WMO/TD-No. 1364 of the World Meteorological Organization. – Online–Ressource, Abruf: 18.02.2007

Allsopp u. a. 2002
Allsopp, David N. ; Beautement, Patrick ; Bradshaw, Jeffrey M. ; Durfee, Edmund H. ; Kirton, Michael ; Knoblock, Craig A. ; Suri, Niranjan ; Tate, Austin ; Thompson, Craig W.: Coalition Agents Experiment: Multiagent Cooperation in International Coalitions. In: *IEEE Intelligent Systems* 17 (2002), Nr. 3, 26–35. `http://i-x.info/documents/2002/2002-ksco-coax-ieee-is-version.pdf`. – ISSN 1541–1672

Ambler u. Hanscome 1998
Ambler, Scott W. ; Hanscome, Barbara: *Process Patterns: Building Large-Scale Systems Using Object Technology.* Cambridge University Press, 1998. – ISBN 0521645689

Amikem 2007
Amikem, Hervé: *Prozess-orientiertes Monitoring Virtueller Organisationen in Globus-basierten Grids*, Fakultät für Informatik der Technischen Universität München, Diplomarbeit, September 2007

Andreozzi u. a. 2007
Andreozzi, Sergio ; Burke, Stephen ; Donno, Flavia ; Field, Laurence ; Fisher, Steve ; Jensen, Jens ; Konya, Balazs ; Litmaath, Maarteen ; Mambelli, Marco ; Schopf, Jennifer M. ; Viljoen, Matt ; Wilson, Antony ; Zappi, Riccardo: *GLUE Schema Specification, Version 1.3, Draft 2.* Version: Januar 2007. `http://glueschema.forge.cnaf.infn.it/Spec/V13`. Draft der Glue Schema Working Group des OGF. – Online–Ressource, Abruf: 24. 5. 2007

Arlow u. Neustadt 2003
Arlow, Jim ; Neustadt, Ila: *Enterprise Patterns and MDA: Building Better Software with Archetype Patterns and UML.* Addison-Wesley Professional, 2003 (The Addison-Wesley Object Technology Series). – ISBN 032111230X

Arora u. a. 2005
Arora, Akhil ; Cohen, Josh ; Davis, Jim ; Dutch, Mike ; Golovinsky, Eugene ; Hagiwara, Yasuhiro ; He, Jackson ; Hines, David ; Inohara, Reiji ; Kämpfe, Christane ; McCollum, Raymond ; Milenkovic, Milan ; Montgomery, Paul ; Nosov, Alexander ; Padlia, Abhay ; Reich, Roger ; Russon, Larry ; Schlimmer, Jeffrey ; Suen, Enoch ; Tewari, Vijay ; Wilson, Kirk: *Web Services for Management.* Spezifikation WS-Management, 2005

Avery u. Foster 2003
Avery, P. ; Foster, I.: iVDGL Annual Report for 2002-2003 / The iVDGL Collaboration. 2003 (GriPhyN/iVDGL 2003-27). – Report

Avižienis u. a. 2001
Avižienis, A. ; Laprie, J. ; Randell, B.: Fundamental Concepts of Dependability / LAAS-CNRS. Version: April 2001. `http://www.cs.ncl.ac.uk/research/pubs/trs/papers/739.pdf`. Toulouse, France, April 2001 (01-145). – Research Report. – Elektronische Ressource

Barton u. a. 2006
Barton, Tom ; Basney, Jim ; Freeman, Tim ; Scavo, Tom ; Siebenlist, Frank ; Welch, Von ; Ananthakrishnan, Rachana ; Baker, Bill ; Goode, Monte ; Keahey., Kate: Identity Federation and Attribute-based Authorization through the Globus Toolkit, Shibboleth, Gridshib, and MyProxy. In: *Proceedings 5th Annual PKI R&D Workshop*, 2006

Barton u. McRae 2005
Barton, Tom ; McRae, Lynn: *Managing Authorization with Signet and Grouper*. Version: Mai 2005. `http://events.internet2.edu/2005/spring-mm/sessionDetails.cfm?session=1946&event=229`. Präsentation im Rahmen des Spring 2005 Internet2 Member Meetings. – Online–Ressource, Abruf: 16. 3. 2007

Bauernhansl 2003
Bauernhansl, Thomas: *Bewertung von Synergiepotenzialen im Maschinenbau*. Deutscher Universitäts-Verlag, Wiesbaden, 2003. – ISBN 3824406888

Beaumont u. Khan 2005
Beaumont, Nicholas ; Khan, Zaffer: A Taxonomy of Refereed Outsourcing Literature / Monash University, Department of Management. Version: Mai 2005. `http://www.buseco.monash.edu.au/depts/mgt/research/working_papers/2005/wp22-05.pdf`. Melbourne, Mai 2005 (22/05). – Working Paper. – Elektronische Ressource

Bell u. a. 2005
Bell, D. ; Kojo, T. ; Golsack, P. ; Loughran, S. ; Milojicic, D. ; Schaefer, S. ; Tatemura, J. ; Toft, P.: *Configuration Description, Deployment, and Lifecycle Management (CDDLM) Foundation Document*. Version: August 2005. `http://www.gridforum.org/documents/GFD.50.pdf`. Report GFD-I.050 der CDDLM Working Group des Global Grid Forums. – Online–Ressource, Abruf: 25. Apr. 2006

van den Berg u. a. 2000
Berg, Roel van d. ; Hannus, Matti ; Pedersen, Jens-Dahl ; Tølle, Martin ; Zwegers, Arian: *Evaluation of State of the Art Technologies, Deliverable 411 des EU-GLOBEMEN-Projektes*. Version: 2000. `http://cic.vtt.fi/projects/globemen/D411_final.zip`. – Online–Ressource, Abruf: 26. Sep. 2005

Berman u. a. 2003
Berman, Fran (Hrsg.) ; Fox, Geoffrey (Hrsg.) ; Hey, Tony (Hrsg.): *Grid Computing - Making the Global Infrastructure a Reality*. Chichester, England : J. Wiley & Sons, ISBN 0-470-85319-0, 2003 (Series in Communications Networking & Distributed Systems)

Berry u. a. 2005
Berry, D. ; Usmani, A. ; Torero, J. ; Tate, A. ; McLaughlin, S. ; Potter, S. ; Trew, A. ; Baxter, R. ; Bull, M. ; Atkinson, M.: FireGrid: Integrated Emergency Response and Fire Safety Engineering for the Future Built Environment.

In: Cox, Simon J. (Hrsg.) ; Walker, David W. (Hrsg.): *Proceedings of the UK e-Science All Hands Meeting (AHM 2005)*. Nottingham, UK, September 2005, 1034–1041

BMI 2005
BMI: *Schutz Kritischer Infrastrukturen - Basisschutzkonzept; Empfehlungen für Unternehmen*. Broschüre des Bundesministeriums des Inneren. `http://www.bbk.bund.de/cln_027/nn_398734/SharedDocs/Publikationen/Publikationen_20Kritis/Basisschutzkonzept__Kritis,templateId=raw,property=publicationFile.pdf/Basisschutzkonzept_Kritis.pdf`. Version: August 2005

Bohlen 2006
Bohlen, Matthias: QVT und Multi-Metamodell-Transformationen in MDA. In: *OBJEKTspektrum* (2006), März/April, Nr. 2, S. 51–56

Bolie u. a. 2006
Bolie, Jeremy ; Cardella, Michael ; Blanvalet, Stany ; Juric, Matjaz ; Carey, Sean ; Chandran, Praveen ; Coene, Yves ; Geminiuc, Kevin ; Zirn, Markus ; Gaur, Harish: *BPEL Cookbook: Best Practices for SOA-based integration and composite applications development*. Packt Publishing, 2006. – ISBN 1904811337

Booth u. a. 2004
Booth, David ; Haas, Hugo ; McCabe, Francis ; Newcomer, Eric ; Champion, Michael ; Ferris, Chris ; Orchard, David: *Web Services Architecture*. Version: Februar 2004. `http://www.w3.org/TR/ws-arch/wsa.pdf`. W3C Working Group Note. – Online–Ressource, Abruf: 23. Aug. 2006

Borghoff u. Schlichter 2000
Borghoff, Uwe M. ; Schlichter, Johann H.: *Computer-Supported Cooperative Work. Introduction to Distributed Applications*. Springer-Verlag, Berlin, 2000. – ISBN 3540669841

Bosworth u. a. 2004
Bosworth, Adam ; Box, Don ; Christensen, Erik ; Curbera, Francisco ; Ferguson, Donald ; Frey, Jeffrey ; Kaler, Chris ; Langworthy, David ; Leymann, Frank ; Lovering, Brad ; Lucco, Steve ; Millet, Steve ; Mukhi, Nirmal ; Nottingham, Mark ; Orchard, David ; Shewchuk, John ; Storey, Tony ; Weerawarana, Sanjiva: *Web Services Addressing (WSAddressing)*. Standard, März 2004

Bourbonnais u. a. 2004
Bourbonnais, S. ; Gogate, V. M. ; Haas, L. M. ; Horman, R. W. ; Malaika, S. ; Narang, I. ; Raman, V.: Towards an Information Infrastructure for the Grid. In: *IBM Systems Journal* 43 (2004), Nr. 4, S. 665–688

Boursas u. Hommel 2006
Boursas, L. ; Hommel, W.: Policy–gesteuerte Datenfreigaben und Trust Management im organisationsübergreifenden Identitäts–Management. In: *Proceedings der*

SICHERHEIT 2006, 3. Jahrestagung des GI–Fachbereichs Sicherheit Bd. 2006. Magdeburg, Deutschland : Springer, Februar 2006, S. 91–101

Brenner u. a. 2006
Brenner, M. ; Sailer, M. ; Schaaf, T. ; Garschhammer, M.: CMDB – Yet Another MIB? On Reusing Management Model Concepts in ITIL Configuration Management. In: *Proceedings of the 17th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM 2006)* Bd. 2006, Springer Berlin / Heidelberg, Oktober 2006, S. 269–280

Brenner 2007
Brenner, Michael: *Werkzeugunterstützung für ITIL-orientiertes Dienstmanagement – ein modellbasierter Ansatz*, Fakultät für Mathematik, Informatik und Statistik der Ludwig-Maximilians-Universität München, Dissertation, 2007

Breuer 2004
Breuer, Holger: *Model Driven Architecture: Ein Ansatz mit dem OpenSource Generator AndroMDA*. GRIN Verlag, 2004. – ISBN 3638281795

Broy u. a. 2002
Broy, M. ; Hegering, H.-G. ; Picot, A. ; Buttermann, A. ; Garschhammer, M. ; Vogel, S.: *Integrierte Gebäudesysteme - Technologien, Sicherheit und Märkte*. Ingelheim : SecuMedia Verlag, ISBN 3–922746–39–X, 2002

Bruegge u. Dutoit 2003
Bruegge, Bernd ; Dutoit, Allen H.: *Object-Oriented Software Engineering: Using UML, Patterns and Java, Second Edition*. Prentice Hall, 2003. – ISBN 0130471100

Bundesamt für Bevölkerungsschutz und Katastrophenhilfe 2007
Bundesamt für Bevölkerungsschutz und Katastrophenhilfe, BBK: *Katastrophenschutzgesetze der Bundesländer mit Änderungsgesetzen und Fundstellen*. Broschüre der Fachinformationsstelle Zivil- und Katastrophenschutz zu den Rechtsvorschriften der Bundesländer zum Katastrophenschutz. `http://www.bbk.bund.de/cln_027/nn_398542/DE/06__Fachinformationsstelle/02__Rechtsgrundlagen/04__Bundeslaender/ListeKatSG,templateId=raw,property=publicationFile.pdf/ListeKatSG.pdf`. Version: Februar 2007

Bundesverband deutscher Banken 2004
Bundesverband deutscher Banken: *Management von kritischen Infrastrukturen*. Version: Mai 2004. `http://www.bankenverband.de/pic/artikelpic/052004/br0405_rb_infrastruktur.pdf`. Bericht des Bundesverbandes deutscher Banken e.V.. – Online–Ressource, Abruf: 9. 3. 2007

Buyya u. a. 2000
Buyya, Rajkumar ; Chapin, Steve J. ; DiNucci, David C.: Architectural Models for Resource Management in the Grid. In: *GRID '00: Proceedings of the First IEEE/ACM International Workshop on Grid Computing*. Springer-Verlag. – ISBN 3–540–41403–7, 18–35

Camarinha-Matos 2005
Camarinha-Matos, Luis M.: ICT Infrastructures for VO. In: Camarinha-Matos, Luis M. (Hrsg.) ; Afsarmanesh, Hamideh (Hrsg.) ; Ollus, Martin (Hrsg.): *Virtual Organizations - Systems and Practices*, Springer Science and Business Media, ISBN 0-387-23755-0, 2005, S. 83–104

Camarinha-Matos u. a. 2005
Camarinha-Matos, Luis M. (Hrsg.) ; Afsarmanesh, Hamideh (Hrsg.) ; Ollus, Martin (Hrsg.): *Virtual Organizations – Systems and Practices*. Springer Science and Business Media, ISBN 0-387-23755-0, 2005

Carley 1995
Carley, Kathleen M.: Computational and Mathematical Organization Theory: Perspective and Directions. In: *Computational & Mathematical Organization Theory* 1 (1995), Oktober, Nr. 1, S. 39–56

Cohen u. a. 2007
Cohen, Josh ; Davis, Doug ; Kreger, Heather ; Tewar, Vijay ; Walker, Martin: *Management Harmonization Overview*. Präsentation im Rahmen der OGF 19, Chapel Hill, USA, Januar 2007

Cojocaru 2007
Cojocaru, Natalia: *Realisierung Virtueller Organisationen als Grid-Ressourcen*, Fakultät für Informatik der Technischen Universität München, Diplomarbeit, September 2007

Crnkovic u. Larsson 2002
Crnkovic, Ivica (Hrsg.) ; Larsson, Magnus (Hrsg.): *Building Reliable Component-Based Software Systems*. Artech House Publishers, ISBN 1580533272, 2002

Czajkowski u. a. 2005
Czajkowski, K. ; Ferguson, D. ; Foster, I. ; Frey, J. ; Graham, S. ; Sedukhin, I. ; Snelling, D. ; Tuecke, S. ; Vambenenepe, W.: *The WS-Resource Framework*. Jan 2005

Czajkowski u. a. 2004
Czajkowski, Karl ; Foster, Ian ; Kesselman, Carl: Resource and Service Management. In: Foster, Ian (Hrsg.) ; Carl, Kesselman (Hrsg.): *The Grid 2 – Blueprint for a New Computing Infrastructure*, Morgan Kaufmann Publishers, ISBN 1-55860-933-4, 2004 (Series in Grid Computing), S. 37–65

Czajkowski u. a. 2002
Czajkowski, Karl ; Foster, Ian ; Kesselman, Carl ; Sander, Volker ; Tuecke, Steven: SNAP: A Protocol for Negotiating Service Level Agreements and Coordinating Resource Management in Distributed Systems. In: *Proceedings of the 8th Workshop on Job Scheduling Strategies for Parallel Processing*. Edinburgh : Springer, Juli 2002 (LNCS 2537), S. 153–183

D-Grid-Initiative 2004
D-Grid-Initiative: *e-Science in Deutschland: F&E-Rahmenprogramm 2005 bis 2009*. Version: Juli 2004. `http://grid.desy.de/d-grid/RahmenprogrammEndfassung.pdf`. Bericht der D-Grid-Initiative. – Online–Ressource, Abruf: 24. 2. 2007

Daft 2006
Daft, Richard L.: *Organization Theory and Design*. South-Western College Publishing, 2006. – ISBN 0324405421

Danciu 2007
Danciu, Vitalian A.: *Application of policy-based techniques to process-oriented IT service management*, Ludwig–Maximilians–Universität München, Dissertation, 2007

Dantas u. a. 2006
Dantas, A. ; Santos, F. ; Germoglio, G. ; Oliveira, M. I. ; Cirne, W. ; Brasileiro, F. ; Rafaeli, S. ; Saikoski, K. ; Milojicic, D.: An Initial Assessment of CDDLM. In: *Proceedings of the 13th Workshop of the HP OpenView University Association: HP-OVUA'06, Cote d'Azur, Frankreich*, 2006

Davidow u. Malone 1993
Davidow, William H. ; Malone, Michael S.: *Das virtuelle Unternehmen. Der Kunde als Co- Produzent*. Campus Fachbuch, 1993. – ISBN 3593349477

DEISA 2006
DEISA: *Distributed European Infrastructure for Supercomputing Applications*. Version: 2006. `http://www.deisa.org/`. Homepage der Distributed European Infrastructure for Supercomputing Applications (DEISA). – Online–Ressource, Abruf: 21. Dez. 2006

DEISA Konsortium 2005
DEISA Konsortium: *DEISA Integrated Infrastructure Initiative, Annex I - Description of Work*. Sixth Framework Programme (Research Infrastructures, Communication network Development), Contract FP6 - 508830, Dezember 2005

Dimitrakos u. a. 2004
Dimitrakos, T. ; Golby, D. ; Kearney, P.: Towards a Trust and Contract Management Framework for Dynamic Virtual Organisations. In: *Proceedings eChallenges 2004*. Vienna, Austria, Oktober 2004

DMTF 2006a
DMTF: *The CIM Tutorial*. Version: 2006. `http://www.wbemsolutions.com/tutorials/DMTF/dmtftutorial.pdf`. – Online–Ressource, Abruf: 9.11.2006

DMTF 2006b
DMTF: *WS-CIM Mapping Specification*. Version: Juli 2006. `http://www.dmtf.org/standards/published_documents/DSP0230.pdf`. Spezifikation DSP0230, Version 1.0.0c. – Online–Ressource, Abruf: 9.11.2006

DMTF 2007
DMTF: Common Information Model (CIM) Core Model. 2007. – Spezifikation, Version 2.15

Dreo Rodošek u. Hegering 2004
Dreo Rodošek, Gabi ; Hegering, Heinz-Gerd: IT–Dienstmanagement: Herausforderungen und Lösungsansätze. In: *Praxis der Informationsverarbeitung und Kommunikation (PIK)* 2004 (2004), Februar, Nr. 02/04, S. 85–92

Dreo Rodošek u. a. 2005
Dreo Rodošek, Gabi ; Hegering, Heinz-Gerd ; Stiller, Burkhard: *Dynamic Virtual Organizations as Enablers for Managed Invisible Grids*. 2005. – eingereicht für 10th IEEE/IFIP Network Operations and Management Symposium (NOMS 2006), Vancouver, Canada

Dreo Rodošek 2002
Dreo Rodošek, Gabrijela: *A Framework for IT Service Management*. München, Ludwig-Maximilians-Universität, Institut für Informatik, Habilitationsschrift, 2002

Dugmore 2006
Dugmore, Jenny: *Achieving ISO/IEC 20000 - The Differences Between BS 15000 and ISO/IEC 20000*. BSI Standards, 2006. – ISBN 0580473481

D.W.Chadwick u. a. 2006
D.W.Chadwick ; A. Novikov ; A. Otenko: GridShib and PERMIS Integration. In: *Campus-Wide Information Systems* 23 (2006), October, Nr. 4, 297-308. `http://www.cs.kent.ac.uk/pubs/2006/2530`. – ISSN 1065–0741

Eckert 2006
Eckert, Claudia: *IT-Sicherheit. Konzepte - Verfahren - Protokolle, 4. Auflage*. Oldenbourg, 2006. – ISBN 3486578510

Eftichidis 2005
Eftichidis, George: *MEDIGRID: Mediterranean Grid of Multi-Risk Data and Models*. Version: Juni 2005. `http://www.vfdb.de/riskcon/presentations/MEDIGRID.ppt`. Präsentation im Rahmen der International Conference on Risk and Emergency Management - Research and Policy Perspectives. – Online–Ressource, Abruf: 27. Okt. 2005

Enterprise Grid Alliance 2006
Enterprise Grid Alliance: *Reference Model and Use Cases Part 1 of 2, Version 1.5*. Version: März 2006. `http://www.ogf.org/documents/06321r00EGA_RefMod-Reference-Model.pdf`. – Online–Ressource, Abruf: 20. 5. 2007

Erl 2005
Erl, Thomas: *Service-Oriented Architecture Concepts, Technology, and Design*. Prentice Hall PTR, ISBN 0131858580, 2005

Farley u. a. 2005
Farley, Jim ; Crawford, William ; Malani, Prakash: *Java Enterprise in a Nutshell*. O'Reilly Media, ISBN 0596101422, 2005 (3. Auflage)

Ferraiolo u. Kuhn 1992
Ferraiolo, David ; Kuhn, Richard: Role-Based Access Control. In: *Proceedings of the 15th National Computer Security Conference*, 1992

Ferro u. a. 2005
Ferro, Enrico ; Fanzago, Federica ; Ciaschini, Vincenzo ; Spataro, Fabio: *Integration of VOMS and LCAS/LCMAPS*. Version: Februar 2005. `http://grid-it.cnaf.infn.it/fileadmin/sysadm/voms-integration/voms-integration.html`. – Online–Ressource, Abruf: 16. 1. 2007

Foster u. a. 2004
Foster, I. ; Gannon, D. ; Kishimoto, H. ; Reich, Jeffrin J. V.: *Open Grid Services Architecture Use Cases*. Version: Oktober 2004. `http://www.gridforum.org/documents/GFD.29.pdf`. Report Global Grid Forum GFD-I.029. – Online–Ressource, Abruf: 12. August 2006

Foster u. a. 2006
Foster, I. ; Kishimoto, H. ; Savva, A. ; Berry, D. ; Djaoui, A. ; Grimshaw, A. ; Horn, B. ; Maciel, F. ; Siebenlist, F. ; Subramaniam, R. ; Treadwell, J. ; Reich, J. von: *The Open Grid Services Architecture, Version 1.5*. `https://forge.gridforum.org/projects/ogsa-wg/document/draft-ggf-ogsa-spec-1.5/en/8`. Version: März 2006

Foster 2005
Foster, Ian: Globus Toolkit Version 4: Software for Service-Oriented Systems. In: *Proceedings of the IFIP International Conference on Network and Parallel Computing* Bd. 3779, Springer Verlag, 2-13

Foster u. a. 2003
Foster, Ian ; Carl, Kesselman ; Nick, Jeffrey M. ; Tuecke, Stevens: The Physiology of the Grid. In: Berman, Fran (Hrsg.) ; Fox, Geoffrey (Hrsg.) ; Hey, Tony (Hrsg.): *Grid Computing - Making the Global Infrastructure a Reality*, J. Wiley & Sons, ISBN 0-470-85319-0, 2003 (Series in Communications Networking & Distributed Systems), S. 217–250

Foster u. Childers 2005
Foster, Ian ; Childers, Lisa: *Introduction to GT4*. Version: September 2005. `http://www.globustoolkit.org/toolkit/tutorials/BAS/APAC/APACGlobusIntro.pdf`. Tutorial at the APAC Conference and Exhibition on Advanced Computing, Grid Applications and eResearch (APAC'05). – Online–Ressource, Abruf: 20. Okt. 2005

Foster u. Kesselman 2004a
Foster, Ian ; Kesselman, Carl: Concepts and Architecture. In: Foster, Ian

(Hrsg.) ; Carl, Kesselman (Hrsg.): *The Grid 2 – Blueprint for a New Computing Infrastructure*, Morgan Kaufmann Publishers, ISBN 1-55860-933-4, 2004 (Series in Grid Computing), S. 37–65

Foster u. Kesselman 2004b
Foster, Ian (Hrsg.) ; Kesselman, Carl (Hrsg.): *The Grid 2*. San Francisco : Morgan Kaufmann Publishers, ISBN 1-55860-933-4, 2004

Foster u. a. 2001
Foster, Ian ; Kesselman, Carl ; Tuecke, Steven: The Anatomy of the Grid: Enabling Scalable Virtual Organizations. In: *International Journal of High Performance Computing Applications* 15 (2001), Nr. 3, S. 200–222

Fowler 1996
Fowler, Martin: *Analysis Patterns: Reusable Object Models*. Addison-Wesley Professional, 1996 (The Addison-Wesley Object Technology Series). – ISBN 0201895420

Fox u. Walker 2003
Fox, Geoffrey ; Walker, David: *e-Science Gap Analysis*. Version: Jun 2003. `http://www.grid2002.org/ukescience/gapresources/GapAnalysis30June03.pdf`. – Online–Ressource, Abruf: 18. Okt. 2005

Frankel 2003
Frankel, David S.: *Model Driven Architecture: Applying MDA to Enterprise Computing*. Wiley, 2003. – ISBN 0471319201

Fulk 1993
Fulk, J.: Social Construction of Communication Technology. In: *Academy of Management Journal* 36 (1993), Nr. 5, S. 921–950

Gamma u. a. 1995
Gamma, Erich ; Helm, Richard ; Johnson, Ralph ; Vlissides, John: *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, 1995. – ISBN 0201633612

Garschhammer u. a. 2001a
Garschhammer, M. ; Hauck, R. ; Hegering, H.-G. ; Kempter, B. ; Langer, M. ; Nerb, M. ; Radisic, I. ; Roelle, H. ; Schmidt, H.: Towards generic Service Management Concepts — A Service Model Based Approach. In: *Proceedings of the 7th International IFIP/IEEE Symposium on Integrated Management (IM 2001)*. IEEE Publishing, 719–732

Garschhammer u. a. 2002
Garschhammer, M. ; Hauck, R. ; Hegering, H.-G. ; Kempter, B. ; Radisic, I. ; Roelle, H. ; Schmidt, H.: A Case–Driven Methodology for Applying the MNM Service Model. In: Stadler, R. (Hrsg.) ; Ulema, M. (Hrsg.): *Proceedings of the 8th International IFIP/IEEE Network Operations and Management Symposium (NOMS 2002)*. IEEE Publishing, 697–710

Garschhammer u. a. 2001b
Garschhammer, M. ; Hauck, R. ; Kempter, B. ; Radisic, I. ; Roelle, H. ; Schmidt, H.: The MNM Service Model — Refined Views on Generic Service Management. In: *Journal of Communications and Networks* 3 (2001), Dezember, Nr. 4, 297–306. `http://wwwmnmteam.informatik.uni-muenchen.de/_php-bin/pub/show_pub.php?key=ghkr01`

Garschhammer u. a. 2003
Garschhammer, M. ; Hegering, H.-G. ; Schiffers, M. ; Broy, M. ; Picot, A.: *Kommunikations- und Informationstechnik 2010+3.* Bonn : Bundesamt für Sicherheit in der Informationstechnik, ISBN 3–922746–48–9, 2003

Garschhammer u. Schiffers 2005
Garschhammer, Markus ; Schiffers, Michael: Integrated IT-Management in Large-Scale, Dynamic, and Multi-Organizational Environments. In: *Proceedings of the 12th Workshop of the HP OpenView University Association: HP-OVUA'05, Porto, Portugal*, 2005

Gemmill u. Robinson 2006
Gemmill, Jill ; Robinson, John-Paul: *myVocs and GridShib: Integrated VO Management.* Präsentation im Rahmen des Spring 2006 Internet2 Member Meeting in Arlington, VA (USA). `http://grid.ncsa.uiuc.edu/presentations/i2mm-myvocs-gridshib-april06.ppt`. Version: April 2006

gentschen Felde u. a. 2006
gentschen Felde, N. ; Hegering, H.-G. ; Schiffers, M.: IT-Service Management Across Organizational Boundaries. In: [**Kern u. a., 2006**], S. 147–178. – ISBN 3540341285

Gentzsch 2005
Gentzsch, Wolfgang: *E-Science-Framework für Deutschland.* Version: 2005. `http://www.d-grid.de`. – Online–Ressource, Abruf: 21. 4. 2007. – Homepage der D-Grid Initiative

Gietz u. a. 2007
Gietz, Peter ; Grimm, Christian ; Gröper, Ralf ; Haase, Martin ; Makedanz, Siegfried ; Pfeiffenberger, Hans ; Schiffers, Michael: *Evaluation of International Shibboleth-Based VO Management Projects.* Version: May 2007. `http://www.d-grid.de/fileadmin/user_upload/documents/DGI-FG1-IVOM/AP1-Report-v1-final.pdf`. Report of Work Package 1 of the IVOM Project, Version 1.0. – Online–Ressource

Gietz u. a. 2006
Gietz, Peter ; Lienhard, Jochen ; Makedanz, Siegfried ; Oberknapp, Bernd ; Pfeiffenberger, Hans ; Rauschenbach, Jürgen ; Ruppert, Ato ; Schroeder, Renate: *DFN-AAI: Technische und organisatorische Voraussetzungen (Attribute), Version 0.8.* November 2006

Girod u. a. 2005
Girod, Bernd ; Rabenstein, Rudolf ; Stenger, Alexander: *Einführung in die Systemtheorie*. Teubner, 2005. – ISBN 3519261944

Global Grid Forum 2005
Global Grid Forum: *Global Grid Forum, History and Background*. Version: 2005. `http://www.ggf.org/L_About/hist&back.htm`. Homepage des Global Grid Forums. – Online–Ressource, Abruf: 1. Nov. 2005

Globus Alliance 2005
Globus Alliance: *Globus Alliance, Background*. Version: 2005. `http://www.globus.org/alliance`. Homepage der Globus Alliance. – Online–Ressource, Abruf: 1. Nov. 2005

Gmür 1993
Gmür, Markus: *Organisationstheorien: Entwicklungslinien - Systematik - Kritik*. Version: 1993. `http://www.ub.uni-konstanz.de/kops/volltexte/1999/336/`. Schriftenreihe Management Forschung und Praxis der Universität Konstanz, Band 7. – Online–Ressource, Abruf: 20.11. 2006

Goyal 2002
Goyal, Brajesh: *Oracle and the Grid*. Version: Nov 2002. `http://www.oracle.com/technology/products/oracle9i/grid_computing/OracleGridWP.pdf`. Technical White Paper. – Online–Ressource, Abruf: 18. Okt. 2005

Greif 1990
Greif, Irene (Hrsg.): *Computer-Supported Cooperative Work: A Book of Readings*. Elsevier Science and Technology Books, ISBN 0934613575, November 1990

GridCoord 2005
GridCoord: *ERA Pilot on a co-ordinated Europe-wide Initiative in Grid Research, Survey of Funding Bodies and Industry*. Version: Juni 2005. `http://www.gridcoord.org/grid/portal/information/public/D.3.1.1_V.2.2_221105.doc`. Deliverable 3.1.1, WP3 - Task 3.1.1, Report IST-2003-511618. – Online–Ressource, Abruf: 2. Jan. 2006

Grimm u. Pattloch 2006
Grimm, Christian ; Pattloch, Marcus: *Analyse von AA-Infrastrukturen in Grid-Middleware*. Version: März 2006. `https://www.d-grid.de/fileadmin/user_upload/documents/DGI-FG3-4/Analyse-AAI_v1_1.pdf`. Report des Fachgebietes 3-4 - Aufbau einer AA-Infrastruktur für das D-Grid, Version 1.1. – Online–Ressource, Abruf: 12. 3. 2007

GriPhyN 2005
GriPhyN: *Grid Physics Network*. Version: 2005. `http://www.griphyn.org/`. Homepage des Grid Physics Network (GriPhyN). – Online–Ressource, Abruf: 21. Dez. 2005

Groep 2005
Groep, David: *International Grid Trust Federation.* Version: Oktober 2005. `http://www.gridpma.org/docs/IGTF-Federation-Constitution.pdf`. – Online–Ressource, Abruf: 23. 5. 2007

Gruhn u. a. 2006
Gruhn, Volker ; Pieper, Daniel ; Röttgers, Carsten: *MDA – Effektives Software-Engineering mit UML2 und Eclipse.* Springer, Berlin, 2006. – ISBN 3540287442

Göhner u. a. 2006
Göhner, Matthias ; Waldburger, Martin ; Gubler, Fabian ; Dreo Rodošek, Gabi ; Stiller, Burkhard: An Accounting Model for Dynamic Virtual Organizations / Universität Zürich, Institut für Informatik. Zürich, November 2006 (2006.11). – Technical Report

Hanemann u. a. 2004
Hanemann, A. ; Sailer, M. ; Schmitz, D.: Variety of QoS — the MNM Service Model Applied to Web Hosting Services. In: *11th International Workshop of the HP OpenView University Association (HPOVUA 2004), Paris* Bd. 2004

Hauck 2001
Hauck, R.: *Architektur für die Automation der Managementinstrumentierung bausteinbasierter Anwendungen*, Institut für Informatik der Ludwig-Maximilians-Universität München, Dissertation, Juli 2001. `http://wwwmnmteam.informatik.uni-muenchen.de/_php-bin/pub/show_pub.php?key=hauc01`. – Elektronische Ressource

Hegering u. a. 1999
Hegering, H.-G. ; Abeck, S. ; Neumair, B.: *Integrated Management of Networked Systems – Concepts, Architectures and their Operational Application.* Morgan Kaufmann Publishers, ISBN 1-55860-571-1, 1999

Hegering 2005a
Hegering, Heinz-Gerd: Grids als Plattform für e-Science. In: *Proceedings "Kbowlege eXtended"*, 2005 (Schriften des Forschungszentrums Jülich, Band 14, ISBN 3-89336-409-0), S. 317–324

Hegering 2005b
Hegering, Heinz-Gerd: Management-Herausforderungen bei Grids. In: *Wissenschaftsmanagement - Zeitschrift für Innovation* (2005), Nr. 1, S. 8–9

Hellwig 2007
Hellwig, Jens: *Bereitstellung verlässlicher Datendienste in Grids unter Berücksichtigung der Dynamik Virtueller Organisationen*, Institut für Informatik der Ludwig-Maximilians Universität München, Diplomarbeit, September 2007

Hennicker 2006
Hennicker, Rolf: *Objektorientierte Software-Entwicklung.* Vorlesungsskriptum

Wintersemester 2005/2006, Institut für Informatik an der Ludwig-Maximilians-Universität München. `http://www.pst.informatik.uni-muenchen.de/lehre/WS0506/oose/`. Version: 2006

Hill u. a. 1994
Hill, Wilhelm ; Fehlbaum, Raymond ; Ulrich, Peter: *Organisationslehre 1.* UTB, Stuttgart, 1994. – ISBN 3825202593

Hippner u. a. 2001
Hippner, Hajo ; Martin, Stephan ; Wilde, Klaus D.: CRM-Systeme - Eine Marktübersicht. In: *HMD - Praxis der Wirtschaftsinformatik* 221 (2001), Oktober

Hommel u. Schiffers 2006
Hommel, W. ; Schiffers, M.: Supporting Virtual Organization Life Cycle Management by Dynamic Federated User Provisioning. In: *Proceedings of the 13th Workshop of the HP OpenView University Association (HP–OVUA), Cote d'Azur, Frankreich* Bd. 2006, Hewlett–Packard Corporation, Mai 2006

Hommel 2007
Hommel, Wolfgang: *Architektur- und Werkzeugkonzept für föderiertes Identitäts-Management*, Fakultät für Mathematik, Informatik und Statistik der Ludwig-Maximilians-Universität München, Dissertation, 2007

Hommel u. Reiser 2005
Hommel, Wolfgang ; Reiser, Helmut: Federated Identity Management: Shortcomings of Existing Standards. In: *Proceedings of the 9th IFIP/IEEE International Symposium on Integrated Management (IM 2005).* Nice, France : IEEE Press, Mai 2005

IBM 2007
IBM: *WSDM/WS-Man Reconciliation An Overview and Migration Guide, Version 2.0.* Version: Mai 2007. `http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-wsdmmgmt/wsdmmgmt_v2.pdf`. – Online–Ressource, Abruf: 1. 7. 2007

Internet2 2003
Internet2: *EduPerson Specification (200312).* Report des Internet2 Middleware Architecture Committee for Education, Directory Working Group (MACE-Dir), Dezember 2003

IPG 2005
IPG: *NASA Information Power Grid.* Version: 2005. `http://www.ipg.nasa.gov/`. Homepage des NASA Information Power Grids (IPG). – Online–Ressource, Abruf: 21. Dez. 2005

IT Governance Institute 2005
IT Governance Institute: *CobiT 4.0, deutsche Ausgabe.* Version: 2005. `http://www.isaca.at/Ressourcen/CobiT%204.0%20Deutsch.pdf`. – Online–Ressource, Abruf: 24. 5. 2007

Jacobson u. a. 1993
Jacobson, Ivar ; Christensen, Magnus ; Jonsson, Patrik ; Övergaard, Gunnar: *Object-oriented Software Engineering: A Use Case Driven Approach (Revised Fourth Printing)*. ACM Press, 1993. – ISBN 0201544350

Jagatheesan u. a. 2002
Jagatheesan, Arun ; Moore, Reagan ; Rajasekar, Arcot ; Zhu, Bing: Virtual Services in Data Grids. In: *Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing HPDC-11 2002 (HPDC'02)*. Washington, DC, USA : IEEE Computer Society, 2002. – ISBN 0–7695–1686–6

Janssen 2003
Janssen, Doris: Kopplung von Web Services - State of the Art Recherche / Fraunhofer IAO. Version: Dezember 2003. `http://www.webservice-kompass.de/fileadmin/publikationen/KopplungVonWebServices-StateOfTheArt_2003-12-19_v1-0.pdf#search=%22orchestrierung%20choreographie%22`. Stuttgart, Dezember 2003. – Forschungsbericht. – Elektronische Ressource

Johnston 2003
Johnston, William E.: Implementing Production Grids. In: Berman, Fran (Hrsg.) ; Fox, Geoffrey (Hrsg.) ; Hey, Tony (Hrsg.): *Grid Computing - Making the Global Infrastructure a Reality*, J. Wiley & Sons, ISBN 0-470-85319-0, 2003 (Series in Communications Networking & Distributed Systems), S. 117–167

Joseph u. Fellenstein 2004
Joseph, Joshy ; Fellenstein, Craig: *Grid Computing*. Indianapolis, USA : Pearson Education, IBM Press, ISBN 0131456601, 2004

Jungert u. a. 2006
Jungert, Erland ; Hallberg, Niklas ; Hunstad, Amund: A Service-Based Command and Control Systems Architecture for Crisis Management. In: *International Journal of Emergency Management (IJEM)* 3 (2006), Nr. 2/3, S. 131–148

Juric 2006
Juric, Matjaz B.: *Business Process Execution Language for Web Services BPEL and BPEL4WS (2. Auflage)*. Packt Publishing, 2006. – ISBN 1904811817

Katzy u. Löh 2003
Katzy, Bernhard ; Löh, Hermann: *Virtual Enterprise Research State of the Art and Ways Forward*. Proceedings of the Center for Technology and Innovation Management (CeTIM). `http://portal.cetim.org/file/1/63/104_Katzy_Loehprint.pdf`. Version: 2003

Katzy u. a. 2005
Katzy, Bernhard ; Zhang, Chunyan ; Löh, Hermann: Reference Models for Virtual Organizations. In: Camarinha-Matos, Luis M. (Hrsg.) ; Afsarmanesh, Hamideh (Hrsg.) ; Ollus, Martin (Hrsg.): *Virtual Organizations - Systems and Practices*, Springer Science and Business Media, ISBN 0-387-23755-0, 2005, S. 45–58

Kaye 2003
Kaye, Doug: *Loosely Coupled: The Missing Pieces of Web Services*. Rds Associates, 2003. – ISBN 1881378241

KBSt 2006
KBSt, Koordinierungs-und Beratungsstelle der Bundesregierung für Informationstechnik in der Bundesverwaltung im Bundesministerium des I.: *V-Modell XT, Release 1.2*. Version: 2006. `http://ftp.uni-kl.de/pub/v-modell-xt/Release-1.2/Dokumentation/Aenderungsvergleich/Vergleich.pdf`. – Online–Ressource, Abruf: 18.10.2006

Keen u. a. 2004
Keen, Martin ; Acharya, Amit ; Bishop, Susan ; Hopkins, Alan ; Milinski, Sven ; Nott, Chris ; Robinson, Rick ; Adams, Jonathan ; Verschueren, Paul: *Patterns: Implementing an SOA Using an Enterprise Service Bus*. IBM International Technical Support Organization (IBM Redbook SG24-6346-00). `http://www.redbooks.ibm.com/redpieces/pdfs/sg246346.pdf`

Keller u. a. 2001
Keller, A. ; Kreger, H. ; Schopmeyer, K.: Towards a CIM Schema for RunTime Application Management. In: *Proceedings of the 12th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM 2001)*. Nancy, Frankreich, Oktober 2001

Keller 1998
Keller, Alexander: *CORBA-basiertes Enterprise Management: Interoperabilität und Managementinstrumentierung verteilter kooperativer Managementsysteme in heterogener Umgebung*, Lehrstuhl für Technische Informatik – Rechnernetze der Technischen Universität München, Dissertation, Dezember 1998

Kempter 2004
Kempter, Bernhard: *Konfliktbehandlung im policy-basierten Management mittels a priori Modellierung*, Fakultät für Mathematik, Informatik und Statistik der Ludwig-Maximilians-Universität München, Dissertation, 2004

Kern u. a. 2006
Kern, Eva-Maria ; Hegering, Heinz-Gerd ; Brügge, Bernd: *Managing Development and Application of Digital Technologies: Research Insights in the Munich Center for Digital Technology & Management (CDTM)*. Berlin : Springer-Verlag, 2006. – ISBN 3540341285

Kim u. Lee 2005
Kim, Eunju ; Lee, Youngkon: *Quality Model for Web Services*. Version: September 2005. `http://www.oasis-open.org/committees/download.php/15910/WSQM-ver-2.0.doc`. OASIS Proposal WSQM -2.0. – Online–Ressource, Abruf: 24. 5. 2007

Kleppe u. a. 2003
Kleppe, Anneke ; Warmer, Jos ; Bast, Wim: *MDA Explained: The Model Driven Architecture–Practice and Promise*. Addison-Wesley Professional, 2003. – ISBN 032119442X

Koch 2003
Koch, Jürgen H.: *Unterstützung der Formierung und Analyse von virtuellen Communities*. Peter Lang, Europäischer Verlag der Wissenschaften, Frankfurt/Main, ISBN 3-631-50288-5, 2003 (Europäische Hochschulschriften, Reihe XLI, Informatik, Band 39)

Koch u. a. 2007
Koch, Nora ; Knapp, Alexander ; Zhang, Gefei ; Baumeister, Hubert: UML-based Web Engineering: An Approach Based on Standards. In: Olsina, Luis (Hrsg.) ; Pastor, Oscar (Hrsg.) ; Rossi, Gustavo (Hrsg.) ; Schwabe, Daniel (Hrsg.): *Web Engineering: Modelling and Implementing Web Applications*, Springer-Verlag, 2007 (Volume 12, Chapter 7 of Human-Computer Interaction Series). – to appear

Krauter u. a. 2002
Krauter, Klaus ; Buyya, Rajkumar ; Maheswaran, Muthucumaru: A Taxonomy and Survey of Grid Resource Management Systems for Distributed Computing. In: *Software – Practice and Experience* 32 (2002), 135 - 164. `http://www.gridbus.org/~raj/papers/gridtaxonomy.pdf`

Kurowski u. a. 2004
Kurowski, Krzysztof ; Nabrzyski, Jarek ; Olegsiak, Ariel ; Węglarz, Jan: Multicriteria Aspects of Grid Resource Management. In: Nabrzyski, Jarek (Hrsg.) ; Schopf, Jennifer M. (Hrsg.) ; Węglarz, Jan (Hrsg.): *Grid Resource Management: State of the Art and Future Trends* Bd. 64, Kluwer Academic Publishers, ISBN 1-4020-7575-8, 2004

Kürümlüoglu u. a. 2005
Kürümlüoglu, Mehmet ; Nøstdal, Rita ; Karvonen, Iris: Base Concepts. In: Camarinha-Matos, Luis M. (Hrsg.) ; Afsarmanesh, Hamideh (Hrsg.) ; Ollus, Martin (Hrsg.): *Virtual Organizations - Systems and Practices*, Springer Science and Business Media, ISBN 0-387-23755-0, 2005, S. 11–28

Köhler 2004
Köhler, Peter T.: *ITIL*. Springer-Verlag, ISBN 3540228934, 2004

Langer 2001
Langer, Michael: *Konzeption und Anwendung einer Customer Service Management Architektur*, Lehrstuhl für Technische Informatik – Rechnernetze der Technischen Universität München, Dissertation, 2001

Larman 2001
Larman, Craig: *Applying UML and Patterns: An Introduction to Object-Oriented*

Analysis and Design and the Unified Process (2nd Edition). Prentice Hall Ptr, 2001. – ISBN 0130925691

von Laszewski u. Wagstrom 2004
Laszewski, G. von ; Wagstrom, P.: Gestalt of the Grid. In: Hariri, Salim (Hrsg.) ; Parashar, Manish (Hrsg.): *Tools and Environments for Parallel and Distributed Computing*, John Wiley and Sons (Wiley Series on Parallel and Distributed Computing), 149-187

LCG 2005
LCG: *Large Hadron Collider Grid.* Version: 2005. `http://lcg.web.cern.ch/LCG/`. Homepage des Large Hadron Collider Grids (LCG). – Online–Ressource, Abruf: 21. Dez. 2005

Lewis 1999
Lewis, Lundy: *Service Level Management for Enterprise Networks.* Norwood, USA : Artech House Telecommunications Library, ISBN 1-58053-016-8, 1999

Lillesand u. a. 2003
Lillesand, Thomas M. ; Kiefer, Ralph W. ; Chipman, Jonathan W.: *Remote Sensing and Image Interpretation.* Wiley, 2003. – ISBN 0471152277

Lin u. a. 2006
Lin, Aizhong ; Vullings, Erik ; Dalziel, James: A Trust-based Access Control Model for Virtual Organizations. In: *Fifth International Conference on Grid and Cooperative Computing Workshops* 0 (2006), S. 557–564. `http://dx.doi.org/http://doi.ieeecomputersociety.org/10.1109/GCCW.2006.14`. – DOI http://doi.ieeecomputersociety.org/10.1109/GCCW.2006.14. ISBN 0–7695–2695–0

Lindner u. Schier 2004
Lindner, Markus ; Schier, Michael: *Web Services Resource Framework.* Vortrag im Rahmen des Seminars „Computing on the Grid“ der Universität Innsbruck, 2004

List 2004
List, Thomas F.: *Nachvollziehbarkeit und Überwachung als Vertrauensdienste auf elektronischen Marktplätzen.* Aachen, Rheinisch-Westfälische Technische Hochschule, Dissertation, Juli 2004

Lupu u. Sloman 1997
Lupu, E. ; Sloman, M.: A Policy-Based Role Object Model. In: *Proceedings of the 1 st IEEE Enterprise Distributed Object Computing Workshop (EDOC'97).* Gold Coast, Australia, Oktober 1997, 36–47

Maciel 2005
Maciel, Frederico B.: *Resource Management in OGSA.* Version: März 2005. `http://www.ggf.org/documents/GFD.45.pdf`. Report GFD-I.045 der Open Grid Service Common Management Model (CMM) Working Group des Global Grid Forums (GGF). – Online–Ressource, Abruf: 21. Dez. 2005

Makedanz u. Pfeiffenberger 2006
Makedanz, Siegfried ; Pfeiffenberger, Hans: *Virtuelle Organisationen und Geschäftsprozesse im Grid*. Workshop Identitäten in elektronischen Geschäftsprozessen, Fraunhofer-Institut für Sichere Informations-Technologie, Darmstadt, September 2006

McGovern u. a. 2003
McGovern, James ; Ambler, Scott W. ; Stevens, Michael E. ; Linn, James ; Jo, Elias K. ; Sharan, Vikas: *The Practical Guide to Enterprise Architecture*. Prentice Hall PTR, 2003. – ISBN 0131412752

Mellor u. Shlaer 1991
Mellor, Stephen J. ; Shlaer, Sally: *Object Life Cycles: Modeling the World In States*. Prentice Hall PTR, 1991 (Yourdon Press Computing Series). – ISBN 0136299407

Mertens 1994
Mertens, Peter: Virtuelle Unternehmen. In: *Wirtschaftsinformatik* 36 (1994), Nr. 2, S. 169–172.

Micacchi u. Cohen 2006
Micacchi, C. ; Cohen, R.: A Multi-Agent Architecture for Robotic Systems in Real-Time Environments. In: *Int. J. Robot. Autom.* 21 (2006), Nr. 2, S. 82–90. – ISSN 0826–8185

Microsoft Corporation 2004
Microsoft Corporation: *Integration Patterns (Patterns and Practices)*. Microsoft Press, 2004. – ISBN 073561850X

Milke u. a. 2006a
Milke, J.-M. ; Schiffers, M. ; Ziegler, W.: Virtuelle Organisationen in Grids: Charakterisierung und Management. In: *Praxis der Informationsverarbeitung und Kommunikation (PIK)* 29 (2006), Juli-September, Nr. 3, S. 165–170

Milke u. a. 2006b
Milke, Jens-Michael ; Schiffers, Michael ; Ziegler, Wolfgang: *Rahmenkonzept für das Management Virtueller Organisationen im D-Grid*. Projektbericht des Arbeitspaketes FG 1-10 des D-Grid Integrationsprojektes (DGI), Version 0.9a, August 2006

Miura 2006
Miura, Kenichi: Overview of Japanese Science Grid Project NAREGI. In: *Progress in Informatics* (2006), Nr. 3, 67-75. `http://www.nii.ac.jp/pi/n3/3_67.pdf`

Mowshowitz 1997
Mowshowitz, Abbe: Virtual Organization. In: *Communications of the ACM* 40 (1997), September, Nr. 9, S. 30–37

Müller 1997
Müller, Thomas: *Virtuelle Organisaton: Konzept, Theoriebasis, Möglichkeiten und Grenzen*. Version: 1997. `http://w3.ub.uni-konstanz.de/v13/volltexte/1999/293//pdf/293_1.pdf`. Schriftenreihe Management Forschung und Praxis der Universität Konstanz, Band 21. – Online–Ressource, Abruf: 20.11. 2006

Nabrzyski u. a. 2004
Nabrzyski, Jarek (Hrsg.) ; Schopf, Jennifer M. (Hrsg.) ; Węglarz, Jan (Hrsg.): *Grid Resource Management: State of the Art and Future Trends*. Kluwer Academic Publishers, ISBN 1-4020-7575-8, 2004 (International Series on Operations Research and Management Science 64)

Nelson 2004
Nelson, David B.: *Grand Challenges: Science, Engineering, and Societal Advances Requiring Networking and Information Technology Research and Development*. Version: März 2004. `http://www.nitrd.gov/pubs/200311_grand_challenges.pdf`. Report of the Interagency Working Group of the National Coordination Office for Information Technology Research and Developmenton (NITRD). – Online–Ressource, Abruf: 1. Nov. 2005

Nerb 2001
Nerb, Michael: *Customer Service Management als Basis für interorganisationales Dienstmanagement*, Lehrstuhl für Technische Informatik – Rechnernetze der Technischen Universität München, Dissertation, 2001

Newcomer 2002
Newcomer, Eric: *Understanding Web Services*. Boston : Addison-Wesley, 2002. – ISBN 0–201–75081–3

NGG2 Expert Group 2004
NGG2 Expert Group: *Next Generation Grid 2: Requirements and Options for European Grids Research 2005-2010 and Beyond*. Version: Jul 2004. `ftp://ftp.cordis.lu/pub/ist/docs/ngg2_eg_final.pdf`. Final Report. – Online–Ressource, Abruf: 26. Okt. 2005

Nguyen-Tuong u. a. 2004
Nguyen-Tuong, Anh ; Grimshaw, Andrew S. ; Wasson, Glenn ; Humphrey, Marty ; Knight, John C.: Towards Dependable Grids / University of Virginia, Department of Computer Science. Charlottesville, USA, März 2004 (Technical Report CS-2004-11). – Forschungsbericht

Norman 2006
Norman, Mark: *What is a VO? Towards a Definition*. Version: 2006. `http://wiki.oucs.ox.ac.uk/esp-grid/VODefinition`. ESP Grid Wiki. – Online–Ressource, Abruf: 29. Aug. 2006

Novotny u. a. 2001
Novotny, J. ; Tuecke, S. ; Welch, V.: An Online Credential Repository for

the Grid: MyProxy. In: *Proceedings of the Tenth International Symposium on High Performance Distributed Computing (HPDC-10)*, IEEE Press, August 2001

OASIS 2004
OASIS: *UDDI Version 3.0.2.* Version: Oktober 2004. `http://uddi.org/pubs/uddi_v3.htm`. UDDI Spec Technical Committee Draft, Dated 20041019. – Online–Ressource, Abruf: 23. Mai. 2007

OASIS 2005
OASIS: *SAML V2.0 Executive Overview.* Version: April 2005. `http://www.oasis-open.org/committees/download.php/13525/sstc-saml-exec-overview-2.0-cd-01-2col.pdf`. Committee Draft 01, 12 April 2005. – Online–Ressource, Abruf: 23. Feb. 2007

OASIS 2006
OASIS: *Electronic Business Using eXtensible Markup Language.* Version: 2006. `http://www.ebxml.org/geninfo.htm`. Homepage der ebXML Initiative. – Online–Ressource, Abruf: 31. Jan. 2006

OASIS 2006a
OASIS: *Web Services Base Faults 1.2 (WS-BaseFaults).* Version: April 2006. `http://docs.oasis-open.org/wsrf/wsrf-ws_base_faults-1.2-spec-os.pdf`. Standard. – Online–Ressource, Abruf: 20. Dez. 2006

OASIS 2006b
OASIS: *Web Services Base Notification 1.3 (WS-BaseNotification).* Version: Oktober 2006. `http://docs.oasis-open.org/wsn/wsn-ws_base_notification-1.3-spec-os.pdf`. Standard. – Online–Ressource, Abruf: 20. Dez. 2006

OASIS 2006c
OASIS: *Web Services Brokered Notification 1.3 (WS-BrokeredNotification).* Version: Oktober 2006. `http://docs.oasis-open.org/wsn/wsn-ws_brokered_notification-1.3-spec-os.pdf`. Standard. – Online–Ressource, Abruf: 20. Dez. 2006

OASIS 2006d
OASIS: *Web Services Distributed Management: Management of Web Services (WSDM-MOWS) Version 1.1.* Version: August 2006. `http://www.oasis-open.org/committees/download.php/20574/wsdm-mows-1.1-spec-os-01.pdf`. OASIS Standard. – Online–Ressource, Abruf: 20. Dez. 2006

OASIS 2006e
OASIS: *Web Services Distributed Management: Management Using Web Services (MUWS 1.1) Part 1.* Version: August 2006. `http://www.oasis-open.org/committees/download.php/20576/wsdm-muws1-1.1-spec-os-01.pdf`. OASIS Standard. – Online–Ressource, Abruf: 20. Dez. 2006

OASIS 2006f
OASIS: *Web Services Distributed Management: Management Using Web Services (MUWS 1.1) Part 2*. Version: August 2006. `http://www.oasis-open.org/committees/download.php/20575/wsdm-muws2-1.1-spec-os-01.pdf`. OASIS Standard. – Online–Ressource, Abruf: 20. Dez. 2006

OASIS 2006g
OASIS: *Web Services Distributed Management: MOWS Primer*. Version: Februar 2006. `http://docs.oasis-open.org/wsdm/wsdm-1.0-mows-primer-cd-01.pdf`. Working Draft 6. – Online–Ressource, Abruf: 20. Dez. 2006

OASIS 2006h
OASIS: *Web Services Distributed Management: MUWS Primer*. Version: Februar 2006. `http://docs.oasis-open.org/wsdm/wsdm-1.0-muws-primer-cd-01.pdf`. Working Draft 6. – Online–Ressource, Abruf: 20. Dez. 2006

OASIS 2006i
OASIS: *Web Services Resource 1.2 (WS-Resource)*. Version: April 2006. `http://docs.oasis-open.org/wsrf/wsrf-ws_resource-1.2-spec-os.pdf`. Standard. – Online–Ressource, Abruf: 20. Dez. 2006

OASIS 2006j
OASIS: *Web Services Resource Framework (WSRF) Primer v1.2*. Version: Mai 2006. `http://docs.oasis-open.org/wsrf/wsrf-primer-1.2-primer-cd-02.pdf`. Committee Draft 02 - 23 May 2006. – Online–Ressource, Abruf: 20. Dez. 2006

OASIS 2006k
OASIS: *Web Services Resource Lifetime 1.2 (WS-ResourceLifetime)*. Version: April 2006. `http://docs.oasis-open.org/wsrf/wsrf-ws_resource_lifetime-1.2-spec-os.pdf`. Standard. – Online–Ressource, Abruf: 20. Dez. 2006

OASIS 2006l
OASIS: *Web Services Resource Properties 1.2 (WS-ResourceProperties)*. Version: April 2006. `http://docs.oasis-open.org/wsrf/wsrf-ws_resource_properties-1.2-spec-os.pdf`. Standard. – Online–Ressource, Abruf: 20. Dez. 2006

OASIS 2006m
OASIS: *Web Services Service Group 1.2 (WS-ServiceGroup)*. Version: April 2006. `http://docs.oasis-open.org/wsrf/wsrf-ws_service_group-1.2-spec-os.pdf`. Standard. – Online–Ressource, Abruf: 20. Dez. 2006

OASIS 2006n
OASIS: *Web Services Topics 1.3 (WS-Topics)*. Version: Oktober 2006. `http://docs.oasis-open.org/wsn/wsn-ws_topics-1.3-spec-os.pdf`. Standard. – Online–Ressource, Abruf: 20. Dez. 2006

OASIS 2007
OASIS: *Reference Model for Service Oriented Architectures.* Version: Februar 2007. http://www.oasis-open.org/committees/download.php/16587/wd-soa-rm-cd1ED.pdf. Committee Draft 1.0, 7. Februar 2007. – Online–Ressource, Abruf: 23. Feb. 2007

Oberhaitzinger u. a. 2004
Oberhaitzinger, Barbara ; Gerloni, Helmar ; Reiser, Helmut ; Plate, Jürgen: *Praxisbuch Sicherheit für Linux-Server und -Netze.* Hanser Fachbuchverlag, 2004. – ISBN 3446226265

Office of Government Commerce 2001
Office of Government Commerce (Hrsg.): *Service Delivery.* Norwich, UK : The Stationary Office, 2001 (IT Infrastructure Library (ITIL))

OGF and DMTF 2006
OGF and DMTF: *OGF / DMTF Work Register Version 11/06/2006.* http://www.dmtf.org/about/register/OGF-DMTFWorkRegister.pdf. Version: November 2006

O'Leary u. a. 1997
O'Leary, Daniel E. ; Kuokka, Daniel ; Plant, Robert: Artificial Intelligence and Virtual Organizations. In: *Commun. ACM* 40 (1997), Nr. 1, S. 52–59. http://dx.doi.org/http://doi.acm.org/10.1145/242857.242871. – DOI http://doi.acm.org/10.1145/242857.242871. – ISSN 0001–0782

OMG 2002
OMG: *UML Profile for CORBA Specification, Version 1.0.* Version: April 2002. http://www.omg.org/docs/formal/02-04-01.pdf. – Online–Ressource, Abruf: 6. 5. 2007

OMG 2003
OMG: *UML 2.0 OCL Specification.* Version: 2003. http://www.omg.org/docs/ptc/03-10-14.pdf. – Online–Ressource, Abruf: 3. 5. 2007

OMG 2004a
OMG: *Metamodel and UML Profile for Java and EJB Specification, Version 1.0.* Version: Februar 2004. http://www.omg.org/docs/formal/04-02-02.pdf. – Online–Ressource, Abruf: 6. 5. 2007

OMG 2004b
OMG: *UML Profile and Interchange Models for Enterprise Application Integration (EAI) Specification, Version 1.0.* Version: März 2004. http://www.omg.org/docs/formal/04-03-26.pdf. – Online–Ressource, Abruf: 6. 5. 2007

OMG 2005
OMG: *MOF QVT Final Adopted Specification.* Version: 2005. http://www.omg.org/docs/ptc/05-11-01.pdf. – Online–Ressource, Abruf: 24. 5. 2007

OMG 2006
OMG: *UML Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms, Version 1.0.* Version: Mai 2006. `http://www.omg.org/docs/formal/06-05-02.pdf`. – Online–Ressource, Abruf: 6. 5. 2007

Otto u. a. 2000
Otto, Boris ; Witzig, Silke ; Fleckstein, Thomas ; Pitsch, Stefan: *Marktstudie Elektronische Marktplätze.* Version: November 2000. `http://www.media-vision.iao.fraunhofer.de/downloads/MS_ElektronischeMarktplaetze_klima.pdf`. Studie des Fraunhofer-Instituts für Arbeitswirtschaft und Organisation (IAO), Stuttgart, und des EMNID-Instituts, Bielefeld. – Online–Ressource, Abruf: 29.12.2006

Parashar u. a. 2002
Parashar, Manish ; Laszewski, Gregor von ; Verma, Snigdha ; Gawor, Jarek ; Keahey, Kate: A CORBA Commodity Grid Kit. In: *Concurrency and Computation: Practice and Experience* 14 (2002), 1057-1074. `http://www.mcs.anl.gov/~gregor/papers/corbacog-ccpe-gce01-final.pdf`

Patel u. a. 2005
Patel, Jigar ; Teacy, W. T. L. ; Jennings, Nicholas R. ; Luck, Michael ; Chalmers, Stuart ; Oren, Nir ; Norman, Timothy J. ; Preece, Alun ; Gray, Peter M. D. ; Shercliff, Gareth ; Stockreisser, Patrick J. ; Shao, Jianhua ; Gray, W. A. ; Fiddian, Nick J. ; Thompson, Simon: Agent-Based Virtual Organisations for the Grid. In: *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS '05).* New York, NY, USA : ACM Press, 2005. – ISBN 1–59593–093–0, S. 1151–1152

Pavlou 1998
Pavlou, George: *OSI Systems Management, Internet SNMP and ODP/OMG CORBA as Technologies for Telecommunications Network Management.* Chapter 2 Telecommunications Network Management: Technologies and Implementations, S. Aidarous, T. Plevyak, eds. pp. 63-109, IEEE Press, 1998. `citeseer.ist.psu.edu/520781.html`. Version: 1998

Petrasch u. Meimberg 2006
Petrasch, Roland ; Meimberg, Oliver: *Model-Driven Architecture. Eine praxisorientierte Einführung in die MDA.* Dpunkt Verlag, 2006. – ISBN 3898643433

Pietryka u. Srivastava 2005
Pietryka, Frank ; Srivastava, Niraj: *Adaptive Enterprise: Grid Computing Services.* Version: 2005. `http://h71028.www7.hp.com/enterprise/downloads/Grid_Services_Whitepaper.pdf`. – Online–Ressource, Abruf: 18. Okt. 2005

Radisic 2003
Radisic, Igor: *Ein prozessorientierter, policy-basierter Ansatz für ein integriertes, dienstorientiertes Abrechnungsmanagement*, Institut für Informatik der Ludwig-Maximilians-Universität München, Dissertation, Januar 2003

Raistrick u. a. 2004
Raistrick, Chris ; Francis, Paul ; Wright, John: *Model Driven Architecture with Executable UML*. Cambridge University Press, 2004. – ISBN 0521537711

Rapley u. a. 2004
Rapley, Chris ; Bell, Robin ; Allison, Ian ; Bindschadler, Robert ; Casassa, Gino ; Chown, Steven ; Duhaime, Gerard ; Kotlyakov, Vladimir ; Kuhn, Michael ; Orheim, Olav ; Pandey, Prem C. ; Petersen, Hanne K. ; Schalke, Henk ; Janoschek, Werner ; Sarukhanian, Eduard ; Zhang, Zhanhai: *A Framework for the International Polar Year 2007-2008*. Version: November 2004. `http://www.icsu.org/Gestion/img/ICSU_DOC_DOWNLOAD/562_DD_FILE_IPY_web_version.pdf`. Report of the ICSU IPY 2007-2008 Planning Group. – Online–Ressource, Abruf: 18.02.2007

Robertson u. Robertson 2006
Robertson, James ; Robertson, Suzanne: *Volere Requirements Specification Template, Edition 11*. Version: Februar 2006. `http://www.volere.co.uk/template.pdf`. – Online–Ressource, Abruf: 23.11.2006

Rolstadås u. Andersen 2000
Rolstadås, Asbjørn ; Andersen, B.: *Enterprise Modeling: Improving Global Industrial Competitiveness*. Springer, 2000 (The International Series in Engineering and Computer Science). – ISBN 0792378741

Rumbaugh u. a. 1993
Rumbaugh, J. ; Blaha, M. ; Premerlani, W. ; Eddy, F. ; Lorensen, W.: *Objektorientiertes Modellieren und Entwerfen*. Hanser Fachbuch, 1993. – ISBN 3446175202

Rupp u. a. 2005
Rupp, Chris ; Hahn, Jürgen ; Queins, Stefan: *UML 2 glasklar. Praxiswissen für die UML -Modellierung und Zertifizierung*. Hanser Fachbuchverlag, 2005. – ISBN 3446229523

Saabeel u. a. 2002
Saabeel, W. ; Verduijn, T.M. ; Hagdorn, L. ; Kumar, K.: A Model of Virtual Organisation: A Structure and Process Perspective. In: *Electronic Journal of Organizational Virtualness* 4 (2002), Nr. 1, 1–16. `http://www.ve-forum.org/Projects/264/Issues/eJOV%20Vol4/Saabeel%20-%202002%20-%20eJOV4,1-1%20-%20A%20Model%20of%20Virtual%20Organisation%20A%20Structure%20and%20Process%20Perspective.pdf`

Saeki u. a. 2006
Saeki, Yuuji ; Pearlman, Laura ; Schopf, Jennfer M. ; Matsuoka, Satoshi: *Multi-Grid Interoperability and Standards in Grid Information Services*. Präsentation im Rahmen des Grid Inteoperation Now (GIN)-Projektes. `http://forge.gridforum.org/sf/docman/do/downloadDocument/projects.`

ogsa-wg/docman.root.design_teams.resource_management.input_materials.gin/doc12801. Version: Februar 2006

Sailer 2007
Sailer, Martin: *Konzeption einer Service-MIB – Analyse und Spezifikation dienstorientierter Managementinformationen*, Fakultät für Mathematik, Informatik und Statistik der Ludwig-Maximilians-Universität München, Dissertation, 2007

Schaad 2003
Schaad, Andreas: *A Framework for Organisational Control Principles*. York, England, Department of Computer Science, The University of York, Dissertation, Juli 2003. http://alloy.mit.edu/papers/st.pdf. – Elektronische Ressource

Schiffers 2004
Schiffers, Michael: Achieving Service Dependability Through Context–Awareness. In: *Proceedings 11th Conference of the HP OVUA, Paris*, Hewlett Packard

Schmidt 2001
Schmidt, Holger: *Entwurf von Service Level Agreements auf der Basis von Dienstprozessen*, Institut für Informatik an der Ludwig-Maximilians-Universität, Dissertation, Juli 2001

Schmidt 2006
Schmidt, Leo: *Technologie als Prozess: Eine empirische Untersuchung organisatorischer Technologiegestaltung am Beispiel von Unternehmenssoftware*, Freie Universität Berlin, Fachbereich Wirtschaftswissenschaft, Dissertation, 2006. http://www.diss.fu-berlin.de/2006/346/. – Elektronische Ressource

Scholz 1994
Scholz, Christian: *Die virtuelle Organisation als Strukturkonzept für die Zukunft?* Diskussionsbeitrag Nr. 30, Lehrstuhl für Betriebswirtschaftslehre, Universität des Saarlandes, Saarbrücken, 1994

Shtub u. a. 2004
Shtub, Avraham ; Bard, Jonathan F. ; Globerson, Shlomo: *Project Management: Processes, Methodologies, and Economics (2nd Edition)*. Prentice Hall, 2004 (Prentice-Hall International Series in Industrial and Systems Engineering). – ISBN 0130413313

Siegel 1996
Siegel, Jon: *CORBA – Fundamentals and Programming*. John Wiley and Sons, ISBN 0-471-12148-7, 1996

Sigletos u. a. 2002
Sigletos, Georgios ; Paliouras, Georgios ; Karkaletsis, Vangelis: Role Identification from Free Text Using Hidden Markov Models. In: *Proceedings of the Second Hellenic Conference on AI (SETN '02)*. London, UK : Springer-Verlag, 2002. – ISBN 3–540–43472–0, S. 167–178

Smith u. a. 2004
Smith, Matthew ; Friese, Thomas ; Freisleben, Bernd: Towards a Service-Oriented Ad Hoc Grid. In: *Proceedings of the Third International Symposium on Parallel and Distributed Computing/Third International Workshop on Algorithms, Models and Tools for Parallel Computing on Heterogeneous Networks (ISPDC/HeteroPar'04)*. IEEE Computer Society. – ISBN 0–7695–2210–6, 201–208

Sotomayor u. Childers 2006
Sotomayor, Borja ; Childers, Lisa: *Globus Toolkit 4 - Programming Java Services*. Morgan Kaufmann Publishers, 2006 (The Elsevier Series in Grid Computing)

Squicciarini u. a. 2007
Squicciarini, Anna C. ; Bertino, Elisa ; Goasguen, Sebastien: Access Control Strategies for Virtualized Environments in Grid Computing Systems. In: *Proceedings of the 11th IEEE International Workshop on Future Trends of Distributed Computing Systems (FTDCS 07)*. Washington, DC, USA : IEEE Computer Society, 2007. – ISBN 0–7695–2810–4, S. 48–54

Stal 2006
Stal, Michael: *Microsoft DCOM*. Version: 2006. `http://www.stal.de/Downloads/DCOM/DCOM.html`. – Online–Ressource, Abruf: 30. Jan. 2006

Strassner 2003
Strassner, John: *Policy-Based Network Management: Solutions for the Next Generation*. Morgan Kaufmann, ISBN 1558608591, 2003 (The Morgan Kaufmann Series in Networking). – ISBN 1558608591

Strong 2007
Strong, Paul: *Report of the OGF Reference Model Working Group*. Präsentation im Rahmen des 20th Open Grid Forum. `http://www.ogf.org/OGF20/materials/755/OGF20-GLUE-joint-session_Paul_Strong.ppt`. Version: Mai 2007

Sun Microsystems 2002
Sun Microsystems: *Core J2EE Patterns - Transfer Object*. Version: 2002. `http://java.sun.com/blueprints/corej2eepatterns/Patterns/TransferObject.html`. – Online–Ressource, Abruf: 3. 5. 2007

Sun Microsystems 2003
Sun Microsystems: *Deploying Grid Computing for Competitive Advantage*. Version: 2003. `http://www.sun.com/solutions/documents/business-cases/grid_advantage_ee.pdf`. – Online–Ressource, Abruf: 18. Okt. 2005

Tan u. a. 2004
Tan, Y.-S. ; Vellanki, V. ; Xing, J. ; Topol, B. ; Dudley, G.: Service Domains. In: *IBM Systems Journal* 43 (2004), Nr. 4, S. 734–755

Tanenbaum u. van Steen 2002
Tanenbaum, Andrew S. ; Steen, Maarten van: *Distributed Systems: Principles and Paradigms*. Prentice Hall, 2002. – ISBN 0130888931

Thompson u. Zald 2003
Thompson, James ; Zald, Mayer: *Organizations in Action: Social Science Bases of Administrative Theory*. Transaction Publishers, 2003 (Classics in Organization and Management Series). – ISBN 0765809915

TMF 2004
TMF: *Shared Information/Data (SID) Model – System View Concepts and Principles, Version 1.1*. Spezifikation GB926 des NGOSS Release 4.0, August 2004

TMF 2007
TMF: *NGOSS Release 7.0 SID Solution Suite (GB922 & GB926)* . Version: Januar 2007. `http://www.tmforum.org/browse.aspx?catID=1696&linkID=32555`. – Online–Ressource, Abruf: 23. 4. 2007

TMF and DMTF 2007
TMF and DMTF: *TeleManagement Forum / DMTF Work Register Version 1.1, Technology and Business (CIM/SID/MTNM) Model Convergence and Coordination*. `http://www.dmtf.org/about/register/DMTF-TMF_Work_Register_v1_1.pdf`. Version: März 2007

Tølle u. Bernus 2003
Tølle, Martin ; Bernus, Peter: Reference Models Supporting Enterprise Networks and Virtual Enterprises. In: *International Journal on Networking and Virtual Organizations* 2 (2003), Nr. 1

Tølle u. a. 2003
Tølle, Martin ; Zwegers, Arian ; Vesterager, Johan: *Virtual Enterprise Reference Architecture and Methodology (VERAM)*. Version: 2003. `http://cic.vtt.fi/projects/globemen/D41_D43_VERAM_Final.zip`. Joint Deliverable D41/D43 des Arbeitspaketes WP4 des Globemen-Projektes. – Online–Ressource, Abruf: 24. Apr. 2006

Treadwell 2006
Treadwell, J.: *Open Grid Services Architecture – Glossary of Terms*. Version: 1.5 vom März 2006. `https://forge.gridforum.org/projects/ogsa-wg/document/draft-ggf-ogsa-glossary-1.5-006/en/6`. – Online–Ressource, Abruf: 12. Mai. 2006

Ulrich 2001
Ulrich, Hans: *Systemorientiertes Management. Studienausgabe*. Haupt-Verlag, St. Gallen, 2001. – ISBN 3258063591

Unicore Forum 2006
Unicore Forum: *UNICORE - A Grid Solution for Your Business*. Version: 2006. `http://www.unicore.org/fileadmin/pdf/Unicore-Brochure-V2.pdf`. Broschüre des UNICORE Forums. – Online–Ressource, Abruf: 8. Feb. 2007

Vaccari u. a. 2006
Vaccari, L. ; Marchese, M. ; Giunchiglia, F. ; McNeill, F. ; Potter, S. ; Tate, A.: *Emergency Response in an Open Information Systems Environment*. Version: Juli 2006. `http://i-x.info/documents/2006/2006-openk-eresponse-deliverable-6.5.pdf`. Deliverable 6.5 of the OpenKnowledge Project. – Online–Ressource, Abruf: 8. 3. 2007

Vahs 2005
Vahs, Dietmar: *Organisation. Einführung in die Organisationstheorie und -praxis*. Verlag Schäffer-Poeschel, 2005. – ISBN 3791023578

Vatle 2005
Vatle, Terje: *Model-based Development of Management Agents*, Universität Karlsruhe, Department of Computer Sciences, Institute of Telematics, Cooperation and Management, Diplomarbeit, September 2005

Vogel 2004
Vogel, Sascha: *Eine Methode zur Entwicklung flexibler Dienste auf der Basis von Interaktionsmustern*, Institut für Informatik der Technischen Universität München, Dissertation, Januar 2004

VOX Project Team 2004
VOX Project Team: *VOMRS User Guide*. Chicago, USA: Fermi National Accelerator Laboratory, Computing Division, 2004. `http://computing.fnal.gov/docs/products/vomrs/pdf/vox.pdf`

Watt u. a. 2006
Watt, J. ; Ajayi, O. ; Jiang, J. ; Koetsier, Jos ; Sinnott, Richard O.: A Shibboleth-Protected Privilege Management Infrastructure for e-Science Education. (2006), S. 357–364. ISBN 0–7695–2585–7

Wedig 2004
Wedig, Arnim: *Formale Modellierung interagierender autonomer und reaktiver Komponenten verteilter Systeme mit I-Systemen : Formale Basis und Beiträge zur Theorie*, Fachbereich Informatik der Universität Dortmund, Dissertation, Jan 2004. `http://ls3-www.cs.uni-dortmund.de/Publications/pdf/wedig.pdf`. – Elektronische Ressource

Weiss 2000
Weiss, Gerhard: *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. The MIT Press, 2000. – ISBN 0262731312

Williamson 1975
Williamson, Oliver E.: *Markets and Hierarchies, Analysis and Antitrust Implications : A Study in the Economics of Internal Organization*. New York : Free Press, ISBN 0029353602, 1975

Xu u. a. 2004
Xu, Ming ; Hu, Zhenhua ; Long, Weihong ; Liu, Wayne: Service Virtualization: Infrastructure and Applications. In: [**Foster u. Kesselman, 2004b**],

Yee 2001
Yee, Ka-Ping: Operating an Emergency Information Service. In: *Commun. ACM* 44 (2001), Nr. 12, S. 25–28. – ISSN 0001–0782

Yu u. Buyya 2005
Yu, Jia ; Buyya, Rajkumar: A Taxonomy of Workflow Management Systems for Grid Computing / Grid Computing and Distributed Systems Laboratory, University of Melbourne. Melbourne, Australien, März 2005 (GRIDS-TR-2005-1). – Technical Report

Zeller 2003
Zeller, Andrew J.: Controlling von Unternehmensnetzwerken: Bestandsaufnahme und Lückenanalyse / Universität Erlangen-Nürnberg. Version: Mai 2003. `http://www.forwin.de/download/berichte/Internet_FWN_2003-002.pdf`. Erlangen-Nürnberg, Mai 2003 (FWN-2003-002). – Bericht des Bayerischen Forschungsverbundes Wirtschaftsinformatik (FORWIN). – Elektronische Ressource

Ziegler u. Grimm 2006
Ziegler, Wolfgang ; Grimm, Christian: *Interoperabilität und Integration der VO-Management Technologien im D-Grid.* Projektantrag im Rahmen des Förderprogramms Service Grids für Forschung und Entwicklung, Juni 2006

Zimmermann u. a. 2005
Zimmermann, Olaf ; Tomlinson, Mark R. ; Peuser, Stefan: *Perspectives on Web Services: Applying SOAP, WSDL and UDDI to Real-World Projects.* Springer, 2005 (Springer Professional Computing). – ISBN 3540009140

Aktoren, Domänen, Anwendungsfälle

Packages, Klassen

Index

Warenzeichen, eingetragene Warenzeichen und Copyrights