Architekturkonzepte für interorganisationales Fehlermanagement

Dissertation

an der

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

vorgelegt von

Gabriela-Patricia Marcu

Tag der Einreichung: 18. April 2011
Tag der mündlichen Prüfung: 18. Mai 2011

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

Danksagung

Diese Arbeit ist im Rahmen meiner Tätigkeit als wissenschaftliche Mitarbeiterin am Leibniz-Rechenzentrum der Bayerischen Akademie der Wissenschaften (LRZ) entstanden. Sie wurde vom DFN-Verein und dem europäischen Projekt Géant gefördert.

Mein besonderer Dank richtet sich an meinen Doktorvater, Prof. Heinz-Gerd Hegering. Dieser nahm mich noch als seine ,,allerletzte" Doktorandin zur Betreuung an. Seine Unterstützung und Ratschläge, sowohl in wissenschaftlichen als auch in menschlichen Angelegenheiten, erwiesen sich sehr oft als wertvoll. Seine hartnäckige Art und Weise: ,,dran zu bleiben" und mich dabei sowohl konstruktiv zu kritisieren als auch freundschaftlich zu ermuntern (insbesondere wenn mancher Fortschritt kaum sichtbar war) habe ich insgeheim sehr oft bewundert. Ich bedanke mich auch recht herzlich bei Prof. Johann Schlichter, der als Zweitgutachter diese Arbeit begleitet hat.

Meinen Kollegen, David Schmitz, Matthias Hamm, Mark Yampolskiy, Wolfgang Fritz, Andreas Hanemann sowie den vielen anderen Mitarbeitern des DFN-Vereins und der anderen europäischen Forschungsnetze, mit denen ich im Rahmen des Géant Projektes zusammengearbeitet habe, ebenfalls ein herzliches: Dankeschön! Ich bedanke mich auch bei meinen Vorgesetzten am LRZ, die da sind: Helmut Reiser, Wolfgang Hommel und Victor Apostolescu, welche mir die Gelegenheit gegeben haben meine wissenschaftliche Tätigkeit mit meiner Projektarbeit zu verbinden. Meinen Rechtschreibprüfern: Felix von Eye, Wolfgang Fritz und insbesondere Matthias Hamm (der keine Zeile ,,unversehrt" ließ) tiefsten Dank!

Ganz besonderes möchte ich mich beim MNM-Team für die zahlreichen Anregungen und vielen wissenschaftlichen Diskussionen, die man dabei laufend im Auge behalten musste, bedanken. Vielen Dank auch an meine Paten: Michael Schiffers, Wolfgang Hommel und Helmut Reiser, die mich während der Entstehung meiner Arbeit konstruktiv begleitet haben; an Vitalian Danciu, Nils gentschen Felde und Silvia Knittl, die mir beim Endspurt des Rigorosums zur Seite standen. Ebenso an Prof. Dieter Kranzlmüller und die anderen Kolleginnen und Kollegen des MNM-Teams.

Für die Gelegenheit meinen wissenschaftlichen Horizont in einem der berühmtesten IT-Forschungszentren zu erweitern, bedanke ich mich stellvertretend bei Larisa Shwartz und Chris Ward vom IBM T.J. Watson Research Center, Hawthorne, NY.

Nicht zuletzt möchte ich mich bei meinen Eltern und meiner Schwester bedanken, die ich, obwohl so weit, trotzdem sehr nahe gefühlt habe. Insbesondere danke ich hiermit meinem Mann Horst, der mich mit seiner Geduld und Ruhe immer wieder unterstützt hat und mich den berühmten Spruch: ,,failure is not an option", aus dem Film Apollo 13, nie vergessen hat lassen.

München, im Juni 2011

d

Abstract

High costs for their infrastructure force IT service providers to move from delivering IT services fully themselves to collaborating with other partners. In this case each party involved delivers a part of the service functionality. IT Service Management - as the totality of the measures to deliver IT services according to agreed quality of service targets - represents a challenge perticularly with regard to inter-organizational environments. In this context fault management is an indispensable part of the management both technical as well as on the service layer. Fault management aims at finding and restoring erroneous states of a service. Thus, fault management is a central part of IT Service Management also as of integrated network management.

In the collaboration of providers usually a hierarchical service delivery is realized. Nevertheless, especially network services often are delivered in a horizontal service chain of several. That means a service is delivered in collaboration of service providers whose parts are chained on the same functional layer. According to a term from organizational theory coined by Hedlund, this is referenced as heterarchical service delivery.

This work aims to establish the so called ioFMA - a management architecture to support the inter-organizational fault management independent of the modality of service delivery. A basis for this work is to gather notions and concepts concerning IT management architectures and inter-organizational service delivery as well as faults, fault classification and fault management.

First, an extensive requirement analysis is done. Therefore, two real scenarios for inter-organizational fault management were taken into account. Based on these, an abstract scenario is deduced. Relying on these three scenarios roles and use cases are defined and analysed and requirements are retrieved. A set of use cases are described and analysed with respect to the modes of inter-organizational service delivery in order to deduce general and specific requirements. In this way functionalities and requirements on the management architecture are retrieved and a requirements catalogue for the ioFMA is concluded.

Based on the derived functionalities and requirements the management architecture is developed stepwise. A platform independent model as well as a platform specific model are established. By using a methodology based on the Model Driven Architecture (MDA) used in software engineering four sub models are realized: organizational model, functional model, information model and communication model. The added value of this thesis consists of the ability of the ioFMA to be instantiated for different service delivery modes. A transformation of the management architecture on the system layer follows. The result is a management platform for supporting inter-organizational fault management. The platform specific model of the platform prototypically implemented for the fault management of the Large Hadron Collider Optical Private Network (LHCOPN).

Zusammenfassung

Die hohen Kosten für die benötigte Infrastruktur motivieren Service Provider, komplexe IT-Dienste nicht mehr vollständig selbst zu erbringen, sondern mit anderen Provider zusammenzuarbeiten. Diese decken jeweils einen Teil der benötigten Dienstfunktionalität ab. Das IT Service Management als Gesamtheit aller Maßnahmen zur Erbringung von IT-Diensten mit dem Ziel der Aufrechterhaltung einer definierten Dienstqualität ist in diesem interorganisationalen Kontext eine Herausforderung. Ein unverzichtbarer Teil des Managements sowohl auf technischer als auch auf Dienstebene ist das Fehlermanagement. Das Ziel des Fehlermanagements ist, eine abweichende Situation vom Normalzustand eines Systems oder eines Dienstes zu finden und sie zu beseitigen. Hiermit kommt dem Fehlermanagement eine zentrale Rolle im IT-Service Management, aber auch im integrierten Netzmanagement zu.

In der Zusammenarbeit zwischen Providern werden üblicherweise hierarchische Formen der Diensterbringung vorausgesetzt. Insbesondere Netzdienste werden jedoch oftmals in Form einer horizontalen Dienstkette erbracht: Dabei wird ein Dienst von mehreren Providern durch eine Kette gleichartiger Teildienste auf derselben funktionalen Schicht realisiert. In Anlehnung an einer Begriff aus der Organisationstheorie wird dies heterarchische Form der Diensterbringung genannt.

Das Ziel dieser Arbeit ist, unabhängig von der Form der interorganisationalen Zusammenarbeit, eine vollständige Managementarchitektur zur Unterstützung des interorganisationale Fehlermanagements mit allen Teilmodellen zu realisieren. Grundlage für die in dieser Arbeit präsentierte Architektur ioFMA ist die Erarbeitung einer Basis an Begriffen und Konzepten bzgl. IT-Managementarchitekturen, interorganisationalen Formen der Diensterbringung, Grundkonzepten im Bereich Fehlermanagement sowie spezieller Begriffe, bezogen auf Fehler, Fehlerklassifikation und Fehlermanagementarchitekturen.

In dieser Arbeit wird zuerst eine tiefgehende Anforderungsanlyse durchgeführt. Für die Ermittlung der Anforderungen werden zwei reale Szenarien für interorganisationales Fehlermanagement herangezogen und daraus ein abstraktes Szenario abgeleitet. Daraufhin werden Rollen und Anwendungsfälle (*use cases*) definiert und analysiert, was zur Ableitung der Anforderungen führt. Bei der Beschreibung der Anwendungsfälle werden parallel die verschiedenen Formen der interorganisationalen Diensterbringung betrachtet, um gezielt auf gemeinsame bzw. spezifische Anforderungen zu schließen. So werden Funktionalitäten und Anforderungen an eine Managementarchitektur abgeleitet, die in einem Anforderungskatalog für eine ioFMA zusammengefasst werden.

Aufgrund der abgeleiteten Funktionalitäten und Anforderungen wird schrittweise eine Managementarchitektur für interorganisationales Fehlermanagement realisiert. Diese wird sowohl als plattformunabhängiges als auch als plattformspezifisches Modell beschrieben. Als Entwurfsmethodik wird die Model Driven Architecture (MDA) aus der Softwareentwicklung herangezogen. Es werden vier Teilmodelle entwickelt: Informationsmodell, Organisationsmodell, Kommunikationsmodell und Funktionsmodell. Ein Mehrwert dieser Arbeit ist

der Vergleich zwischen den unterschiedlichen Formen der Diensterbringung in Bezug auf die entwickelte ioFMA. Es folgt die Transformation der Managementarchitektur auf die Systemsicht: eine Managementplattform zur Unterstützung von interorganisationalen Fehlermanagementprozessen wird realisiert, die der vorher entwickelten Managementarchitektur entspricht. Das plattformspezifische Modell wird beispielhaft in Form eines Tools für das Fehlermanagement im Large Hadron Collider Optical Private Network (LHCOPN) prototypisch implementiert.

h

Inhaltsverzeichnis

1 Einleitung

Inhalt des Kapitels

Während der letzten Jahrzehnte wurde die Informations- und Kommunikationstechnologie von der Problematik der betriebswirtschaflichen Effizienz geprägt. Damit ist hauptsächlich mehr Leistung bei niedrigeren Kosten gemeint. Dieses Ziel zu erreichen und aufrechtzuerhalten ist in der heutigen IT-Welt eine Herausforderung, da die IT-Landschaften selten statisch bleiben, sondern sich durch ständige Veränderungen auszeichnen. Die Beziehungen zwischen Kunden und Providern von IT-Diensten ändern sich häufig, was immer neue Anforderungen mit sich bringt. Die Technologie selbst unterliegt raschen Veränderungen und nicht zuletzt nimmt die räumliche Verteilung der IT-Komponenten immer stärker zu. In dieser Arbeit liegt der Fokus auf dem letzten Punkt, speziell auf den interorganisationalen Eigenschaften einer IT-Landschaft und der Dienste, die darin angeboten werden. Viele Forschungsprojekte in den letzten Jahren konzentrieren sich auf interorganisationale IT-Umgebungen [Geant11, DEISA, EGEE] und deren technologischer Aufbau, die übergreifende Kommunikation zwischen den Partnern, die Überwachung der Netze und nicht zuletzt das Management. Das Thema Management vernetzter Systeme hat seinen Anfang in den 90er Jahren, und spätestens mit der Etablierung des integrierten Netz- und Systemmanagements [HAN 99] wurde die Effizienz und Effektivität der erbrachten IT-Dienste ein Ziel jedes Betreibers von IT-Diensten.

Integriertes Netz- und Systemmanagement

Die Aufgabenbereiche des Netz- und Systemmanagements sind als so genannte Managementfunktionen Bausteine des Gesamtkomplexes „IT-Management". Die Managementfunktionen

FCAPS

werden in der Praxis nach unterschiedlichen Kriterien klassifiziert Das Funktionsmodell der OSI-Managementarchitektur [OSI 89] mit seinen fünf Funktionsbereichen: Fehlermanagement, Konfigurationsmanagement, Leistungsmanagement, Sicherheitsmanagement und Abrechnungsmanagement hat heute breite Akzeptanz gefunden.

Fehlermanagement

Das Fehlermanagement als eigenständige Managementfunktion ist zuständig für das Entdecken, Eingrenzen und Beheben von Fehlern, d.h. Abweichungen vom normalen Betrieb von Ressourcen [HAN 99]. Die Fehlerursachen in einem verteilten System sind vielfältig [OGP 03]. Trotz unterschiedlichster Fehlerquellen ist das Ziel des Fehlermanagements: die Verfügbarkeit der Systeme und seiner Dienste zu gewährleisten [PaHa 07]. Dieses Ziel wird durch das Entdecken, Eingrenzen und Beheben von Störungen realisiert [Cand 03]. Diese Teilaufgaben des Fehlermanagements umfassen mehrere wichtige Unteraufgaben, u. A.: die Überwachung der Dienste, die Bearbeitung von Alarmen, die Diagnose der Fehlerursachen [CQM$^+$ 09] sowie die Korrelation von unterschiedlichen Fehlern. In diesem Kontext spielen Trouble Ticket Systeme als unterstützende Werkzeuge des Fehlermanagements eine wichtige Rolle [GPM 08a].

IT Service Management

Neben der Definition der Funktionsbereiche des IT-Managements nimmt in den letzten Jahren die Dienstorientierung in der IT-Branche einen immer höheren Stellenwert ein. IT-Service-Management (ITSM) bezeichnet die Gesamtheit der Maßnahmen zur Erbringung der IT-Dienste mit dem Ziel der Aufrechterhaltung einer definierten Dienstqualität [HAN 99]. In den letzten Jahren wurden mehrere Rahmenwerke für das ITSM entwickelt. Unterschiedliche Ansätze in diesem Forschungsgebiet wurden realisiert: manche mit einem höheren Bekanntheitsgrad; manche, die nicht lange Zeit standhalten konnten. Die wichtigsten davon sind: die IT Infrastructure Library (ITIL) [OGC 07a], [OGC 07b], das Control Objectives for Information and Related Technology (COBIT) Framework [COBIT], die Enhanced Telecom Operations Map (eTOM) [eTOM], sowie die ISO/IEC 20000 der International Organization of Standardization (ISO) und der International Electrotechnical Commission (IEC) [ISO 20000:1].

Prozessorientierte Ansätze

In Ergänzung zu den Funktionsbereichen sind die ITSM Frameworks durchgängig prozessorientiert. Das vom Office of Government Commerce (OGC) herausgegebene „Best Practice“ Referenzframework ITIL sowie eTOM, herausgegeben vom TeleManagementForum (TMF), definieren Managementprozesse für unterschiedliche Funktionsbereiche des Dienstmanagements. COBIT hingegen ist ein betriebswirtschaftlich orientiertes Framework und umfasst Anforderungen und Modelle zur Bewertung des Reifegrades von Service Management Prozessen. Der auf ITIL beruhende ISO/IEC 20000 ist der erste formelle Standard im Bereich des IT Service Managements [ISO 20000:1].

Incident und Problem Management

Die Incident- und Problem-Management-Prozesse sind zwei der Service-Operation-Prozesse der ITIL. Sie entsprechen dem Funktionsbereich Fehlermanagement in der OSIManagementarchitektur. Diese Prozesse beinhalten die Erkennung (engl. recognizing), Protokollierung (engl. logging), Eingrenzung (engl. isolate) und Behebung (engl. correct) von Störungen, d.h. Einschränkungen oder Unterbrechungen der Dienstfunktionalität.

1.1. Motivation

Dieser Arbeit konzentriert sich auf einen Funktionsbereich des ITSM, nämlich das Fehlermanagement. Dabei soll ein bislang kaum systematisch bearbeiteter Aspekt analysiert werden, nämlich das organisationsübergreifende Fehlermanagement. Insbesondere soll hierfür, wie weiter unten ausgeführt, eine Managementarchitektur entworfen werden.

Outsourcing

In den letzten Jahren haben viele Organisationen den Betrieb von Diensten wie E-Mail, Web-Hosting usw. mehr und mehr zu externen Anbietern ausgelagert. Das Vorgehen, eine Menge von Prozessen und Diensten, die vorher intern betrieben wurden, nach außen zu einer anderen Organisation zu verlagern, nennt man *Outsourcing*. Die Fragen, die sich in diesem Kontext stellen, sind: Was treibt eine Organisation dazu ihre Dienste outzusourcen? Gleicht sich der Gewinn, den die Organisation durch den eigenen Betrieb der Dienste erzielt, mit dem Aufwand bzw. den Kosten, um diese Dienste zu managen, aus? ITIL v3 in [OGC 07a] beschreibt den Übergang vom value-chain-Modell als bekanntes hierarchisches Diensterbringungsmodell zum Modell eines value-network, auch horizontale kollaborative Beziehungen beinhaltet. Dabei werden mehrere Sourcing-Strategien beschrieben. In eTOM wird die Problematik mit dem Konzept des e-business aufgegriffen und hinsichtlich der Interaktion zwischen den unterschiedlichen Geschäftspartnern definiert. Dies wird zu einem späteren Zeitpunkt in dieser Arbeit ausführlich behandelt. Bemerkenswert ist, dass sich die Problematik des Outsourcings in vielen Prozessen widerspiegelt und insbesondere auch einen signifikanten Einfluss auf das Fehlermanagement hat.

Heterogenität und Autonomie

Die Verschiedenartigkeit der Organisationsformen in der Zusammenarbeit von IT-Providern erschweren ein effizientes und effektives interorganisationales Fehlermanagement. Die bekannteste Form der Zusammenarbeit zwischen Service Providern ist die Hierarchie. Das heißt, ein Provider bietet einem Kunden einen komplexen Dienst an, für dessen Erbringung mehrere Subdienste benötigt werden. Viele diese Dienste können nicht in-house angeboten werden, also werden diese von anderen Provider zugekauft. Der Vorteil im Fall der Hierarchie ist, dass gegenüber den Kunden der Service Provider Ansprechspartner für den erbrachten Dienst ist, und dieser sich um Verträge mit weiteren Providern kümmert. Damit entsteht eine Diensthierarchie. Eine andere Form der interorganisationalen Diensterbringung ist die Heterarchie [Hedl 05]. Ein Kunde benutzt einen komplexen Dienst, der sich durch eine horizontale Dienstkette kennzeichnet. Das heißt der erbrachte Dienst besteht aus einer Kette von Teildiensten, die zu einem gemeinsamen Dienst verbunden sind. In diesem Fall findet zwischen den Providern eine kooperative Zusammenarbeit statt. Auf diesen zwei Organisationsformen in der Zusammenarbeit basieren die interorganisationalen Formen der Diensterbringung. Der gemeinsame Nenner der Organisationsmodelle ist die Heterogenität und Autonomie, was sich insbesondere widerspiegelt in den eingesetzten unterschiedlichen Systemen und der Werkzeugunterstützung. In dieser Hinsicht ist es eine Herausforderung, trotz hoher Heterogenität und Autonomie Konzepte zu entwerfen, die auch organisationsübergreifend auf dem Gebiet des Fehlermanagements eingesetzt werden können.

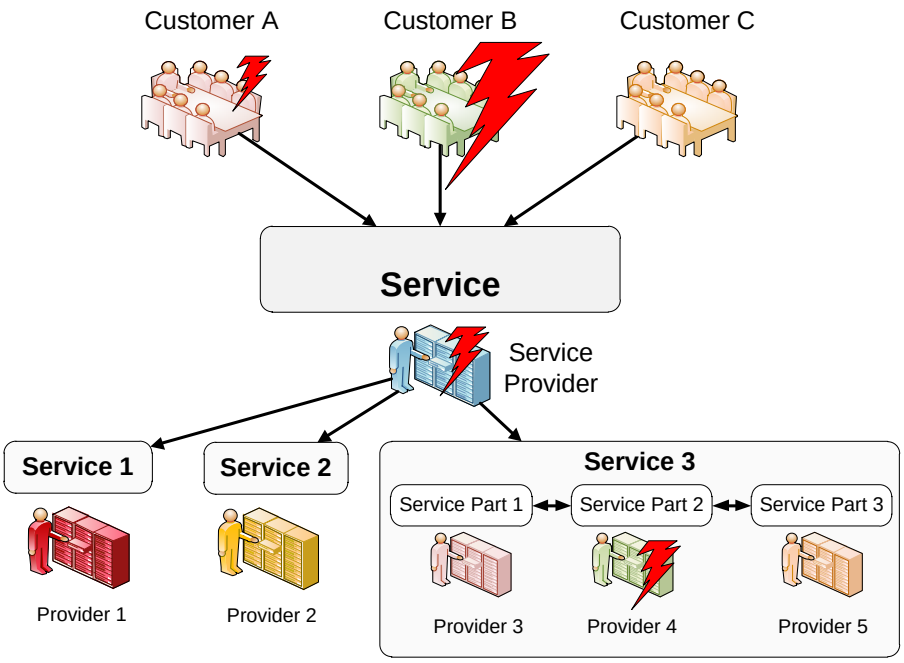


Abbildung 1.1.: Fehlerpropagation in interorganisationale Umgebungen

Unterschiedliche Formen der Diensterbringung

Es gibt vielfältige Unterschiede in der Diensterbringung, in der Prozessführung, in der Kommunikation usw. Referenzprozesse für das Fehlermanagement für verschiedene Organisationsformen wurden in Vorarbeiten wie z.B [OGC 07d] und [eTOM] für hierarchische und in [Hamm 09] für kooperative Diensterbringung definiert. Auf dieser Basis soll in der vorliegenden Arbeit eine Managementarchitektur für das interorganisationale Fehlermanagement entwickelt werden.

Abbildung 1.1 zeigt schematisch ein Szenario interorganisationaler Diensterbringung. Ein Dienstleister (Service Provider) erbringt einen Service für seine Kunden (Customer A, Customer B, Customer C) in unterschiedlichen Ausprägungen. Der Dienstleister hat drei der von ihm benötigten Dienste (Service 1, Service 2 und Service 3) zu anderen Service Providern verlagert (Outsourcing). Bis hierher liegt eine hierarchische Form der Diensterbringung vor, eine sogenannte vertikale Dienstkette. Die beiden Dienste (Service 1 und Service 2) werden von jeweils einem einzigen Dienstleister erbracht (Provider 1 bzw. Provider 2). Im Gegensatz zu diesen wird Service 3 von mehreren gleichberechtigten Providern (Provider 3, Provider 4 und Provider 5) erbracht. Diese "gleichwertigen" Provider (sie befinden sich alle funktional betrachtet in der gleichen Dienstschicht) liefern Teildienste (Service Part 1, Service Part 2 bzw Service Part 3), die zur Erbringung eines einzigen, in der gleichen Dienstschicht konkatenierten, Dienstes führen. Dies wird durch eine horizontale Dienstkette (heterarchische Form der Diensterbringung) realisiert. Die Beteiligten, hier Service Provider 1-5 und Customer A-C, befinden sich selten in einer homogenen Umgebung, insbesondere, da alle Beteiligten

autonom sind. In den meisten Fällen folgt jede Organisation ihren eigenen Anforderungen, Abläufen und Prozessen und benutzt unterschiedliche Infrastrukturen, Systeme und Software. Die Werkzeugunterstützung wird ebenfalls von jeder Organisation auf unterschiedliche Weise realisiert. Das Szenario ist somit sehr heterogen.

Ein Fehler (unabhängig von der Ursache) innerhalb einer Organisation, z. B. der des Providers 4, trägt dazu bei, dass dieser einen Teil des Dienstes Service 3 (Service Part 3) nicht erbringen kann. Dieser Fehler wirkt sich bei dem Service Provider aus, der seinen Service fehlerhaft (oder gar nicht) seinen Kunden erbringt. Je nach der kundenspezifischen Anpassung des Dienstes kann der Fehler größere oder kleinere quantitative und qualitative Folgen haben. Jedoch ist es in solchen Situationen sehr schwierig, Fehler zu finden, miteinander zu korrelieren und deren Bearbeitung bzw. Behebung zu verfolgen. Mögliche Probleme, die bei der Erbringung in einem interorganisationalen Umfeld, auftreten können, sind: der Verlust von Störungsmeldungen, die Einschränkung des Austauschs der benötigten Informationen bzgl. der Störungen oder fehlende Ansprechpartner für Störungen und Fehlerbehebung. Auch wenn es auf intraorganisationaler Ebene bereits etablierte Ansätze und bekannte best-practice-Vorgehen gibt, so existieren auf interorganisationaler Ebene in dieser Hinsicht bisher nur Versuche und keine systematischen Lösungsuntersuchungen.

interorganisationales Fehlermanagement

Für sehr große IT Service Provider, die multinational organisiert sind und die aus mehreren kleineren Organisationen bestehen oder die ihre IT Dienste ausgelagert haben, ist die Durchführung von ITSM Prozessen äußerst komplex. Wenn man die Outsourcing- Problematik aus einem prozessorientierten Gesichtspunkt betrachtet, kann man die oben genannte Heterogenität sowohl mit einer steigenden Komplexität auf Prozesssicht als auch auf Systemsicht verbinden. Unter diesen Umständen ist ein geeignetes interorganisationales Fehlermanagement notwendig und unverzichtbar.

1.2. Fragestellungen

Aus dem Obigen folgt die Notwendigkeit der Ausweitung der Konzepte des Fehlermanagements auf die interorganisationale Ebene, um das Erkennen, Eingrenzen und Beheben von Fehlern auch in diesen komplexen Szenarien zu ermöglichen. Dazu stellen sich die folgenden Fragen:

Formen der interorganisationalen Diensterbringung

Ein zentraler Punkt, dem auf interorganisationaler Ebene Aufmerksamkeit geschenkt werden muss, ist die Art der Diensterbringung. Ob die Provider in der Diensterbringung eine hierarchische (vertikale Kette von Providern) oder eine heterarchische (horizontale Kette von Providern) Struktur verfolgen ist für den Prozessablauf (Workflow) und das IT-Service-Management wesentlich. Die unterschiedlichen Arten der Diensterbringung haben insbesondere einen Einfluss auf das interoganisationale Fehlermanagement. Hiermit lautet die erste zentrale Fragestellung dieser Arbeit: *Welche Auswirkungen haben die unterschiedlichen Arten der Diensterbringung auf das interorganisationale Fehlermanagement?* Um

diese zentrale Frage beantworten zu können, werden in dieser Arbeit folgende Teilfragen untersucht:

Q1: Lebenszyklus eines Fehlers und Formen der Diensterbringung Grundsätzlich gibt es im Lebenszyklus einer Fehlerbehandlung drei wichtige Phasen: das Entdecken, das Eingrenzen und das Beheben der Störung. Mehrmals wurden diese in der Literatur für intraorganisationale Sachverhalte untersucht, aber auf interorganisationaler Ebene wurden sie bisher vernachlässigt [Lewi 95]. Insbesondere in den ersten zwei Phasen ist es notwendig zwischen den unterschiedlichen Formen der Diensterbringung zu unterscheiden. Ein Fehler, der bei Kunden aufgetreten ist, wird auf (teilweise) unterschiedlichen Wegen bei den Service Providern oder deren Provider entdeckt und eingegrenzt, abhängig von der Diensterbringungsform dieses Dienstes. Es stellt sich daher folgende Frage:
Was ist spezifisch im Lebenszyklus eines Fehlers in Bezug auf die unterschiedlichen Formen der Diensterbringung?

Q2: Rollen Wie unterschiedliche Organisationen die Zusammenarbeit untereinander organisieren, ist ein wichtiger Faktor für den interorganisationalen Fehlermanagementprozess. Die Identifizierung von beteiligten Rollen und deren Differenzierung, abhängig von den unterschiedlichen Formen der Diensterbringung, ist dabei entscheidend. Auf dieser Grundlage ergibt sich folgende Problematik:
Welche Rollen sind im Rahmen eines interorganisationalen Fehlermanagements von Bedeutung?
Welche Interaktionen sind relevant für den hierarchischen bzw. heterarchischen Fall? Wie sind die Rollen am Lebenszyklus einer Störung beteiligt?

Q3: Information und Kommunikation In heterogenen interorganisationalen Umgebungen kann a priori kein standardisierter Informationsaustausch vorausgesetzt werden. Die Notwendigkeit eines spezifischen Informationsmodells als auch eines Kommunikationsmodells ergibt sich daraus, dass der Fehlermanagementprozess auf interorganisationaler Ebene ohne einheitliche, von allen Beteiligten anerkannte Datenformate nicht geführt werden kann. Das Informationsformat kann teilweise Unterschiede aufweisen, abhängig von der Form der Diensterbringung. Wichtige Fragestellungen sind dabei:
Welche Informationen werden auf welchen Kommunikationswegen für die erfolgreiche Durchführung des interorganisationalen Fehlermanagements benötigt?
Welche Unterschiede ergeben sich dabei in Bezug auf die unterschiedlichen Formen der Diensterbringung?

Q4: Funktionalitäten Unterschiedliche Funktionalitäten bzw. Funktionsbausteine werden benötigt. Dabei sind die Aspekte für das interorganisationale Fehlermanagement in Betracht zu ziehen. Der Hauptpunkt ist dabei die Identifizierung und Definition von flexibel konfigurierbaren Funktionalitäten, die für interorganisationale Umgebungen skalieren sollten. Des Weiteren sind die möglichen Unterschiede, bezogen auf unterschiedliche Formen der Diensterbringung, für die weitere Anwendung der Architektur zu untersuchen.

Welche spezifischen Funktionalitäten sind für ein interorganisationales Fehlermanagement notwendig und wie können diese realisiert werden?
Wie unterscheiden sich diese für die unterschiedlichen Formen der Diensterbringung?

Werkzeugorientierung

Ein weiterer zentraler Schwerpunkt dieser Arbeit ist ein methodisches Vorgehen zur Operationalisierung des interorganisationalen Fehlermanagements. Abhängig von der Lösung der ersten zentralen Fragestellung werden im Rahmen des zweiten Schwerpunktes die Ergebnisse der Teilfragen Q1 bis Q4 zu einer interorganisationalen Fehlermanagementarchitektur (ioFMA) zusammengeführt. Dies ist ein wichtiger Punkt, da hiermit die praktische Anwendbarkeit der ioFMA gezeigt werden kann. Hierbei sind folgende Fragen zu betrachten:

Q5: Abbildung der ioFMA auf Werkzeugebene Um die Konzepte der ioFMA in einer realen Umgebung zu implementieren, muss diese auf die Systemsicht (die reele IT-System Ebene) abgebildet werden.

Wie kann die ioFMA auf Systemsicht abgebildet werden? Was sind dabei die Besonderheiten im Hinsicht auf unterschiedliche Formen der interorganisationalen Diensterbringung?

Q6: Werkzeugorientierung Das Fehlermanagement wird organisationsintern durch Werkzeuge (z. B. Trouble-Ticket-Systeme) unterstützt. Da unterschiedliche Organisationen unterschiedlichste Werkzeuge zur Entdeckung, Eingrenzung und Behebung von Fehlen benutzen, ist die Heterogenität der Werkzeuge ein Problem. Hiermit ergibt sich bei der Realisierung eines interorganisationalen Fehlermanagements auch die Problematik der Koppelung von Werkzeugen unterschiedlicher Organisationen. In diesem Zusammenhang ist auch das Deployment zu betrachten, d. h. die Einführung der ioFMA in einer bestimmten Umgebung. Das Deployment wird grundsätzlich in mehreren Schritten realisiert. Es stellt sich daher die Frage:

Wie kann dieses ,,interorganisationale Werkzeug" mit anderen Werkzeugen (wie z.B. Trouble-Ticket-Systeme) interagieren? Wie kann man unterschiedliche Werkzeuge miteinander koppeln? Welche Deployment-Schritte werden benötigt um die ioFMA in einer interorganisationalen Umgebung einzuführen?

1.3. Einordnung der Themenstellung

Matthias Hamm untersucht in seiner Arbeit [Hamm 09] Betriebsprozesse Verketteter Dienste; das sind horizontale Dienstketten, die in einer Kooperation von Providern erbracht werden. Er entwickelt einen Ordnungsrahmen zur Provider-Koordination und beschreibt Referenzprozesse, die als Ansatz für diese Arbeit genutzt werden können. Ergänzend zu den in der Arbeit von Hamm beschriebenen Referenzprozessen als auch den hierarchischen

Aspekten auf Prozessebene, wird in dieser Arbeit eine Architektur zum interorganisationalen Fehlermanagement beschrieben. Zur genaueren Abgrenzung der Arbeiten vgl. Kapitel 4.

Mark Yampolskiy entwirft in seiner Dissertation [Yamp 09] ein Managementkonzept für das Service Level Management (SLM) Verketteter Dienste. Er fokussiert sich auf die Frage, wie die Dienstgüte in Providerheterarchien bei End-to-End Diensten zugesichert werden kann. Das Fehlermanagement weist Schnittstellen zum SLM auf; diese werden in der vorliegenden Arbeit entsprechend berücksichtigt. Dieses steht sehr eng in Zusammenhang mit dem Fehlermanagement, so dass in der Realisierung der hier beschriebenen Architektur und insbesondere bei der Anwendung für heterarchischen Organisationsstrukturen Bezug auf die Quality of Service (QoS) genommen wird.

Silvia Knittl [Knit 10] konzipiert in ihrer Dissertation ein Werkzeug, welches das IT-Service-Management organisationsübergreifender Dienste unterstützt; hierbei führt sie eine interorganisationale CMDB (ioCMDB) ein. Diese ioCMDB ist ähnlich einer Configuration Management Database (CMDB) im intraorganisationalen Umfeld, die ITSM-Prozesse gemäß ITIL unterstützt. In dieser Arbeit wird vorausgesetzt, dass eine ioCMDB benutzt wird. Es wird angenommen, dass dieses Werkzeug dem interorganisationalen Fehlermanagement die notwendigen Informationen, wie z.B. Dienstzusammensetzung, Verantwortlichkeiten etc., liefert.

Feng Liu [Liu 11] konzentriert sich in seiner Dissertation darauf, ein intelligentes Recovery-Framework unter Verwendung von KI-basierte Techniken zur automatischen Planung (automated planning) zu entwickeln. Dieses soll Administratoren bei der Erstellung von Recoveryplans unterstützen. Die Idee dahinter ist, die Stärken der KI-Techniken bei der automatischen Suche und Auswahl von plausiblen Recoveryplänen beim Auftreten von Fehlern zu nutzen. Die hier entwickelte Managementarchitektur konzentriert sich hauptsächlich auf das Monitoring, Finden und die Eingrenzung von Fehlern, ergänzt also Lius Arbeit.

Andreas Hanemann und David Schmitz gehen in ihren Dissertationen das Thema Fehlermanagement auf zwei Weisen an: Top-Down (Root-Cause-Analysis) und Bottom-up (Impact-Analysis). Hanemann etwickelt in [Hane 07] ein Framework für dienstorientierte Ereigniskorrelation. Die von Dienstnutzern gemeldeten Störungen werden dabei korreliert und kombiniert mit Störungsmeldungen aus den unterliegenden Schichten, um die mögliche Fehlerursache (root cause) zu identifizieren. Schmitz hingegen geht in seiner Arbeit [Schm 08] von bereits bekannten Fehlerursachen in einem System aus. Das von ihm entwickelte Framework kann für die Analyse der Auswirkungen dieser gegebenen Fehler auf Dienste und Anwender hinsichtlich der vereinbarten SLAs eingesetzt werden. Diese Arbeiten ergänzen sich gegenseitig, und gleichzeitig sind sie für die vorliegende Arbeit grundlegend.

Michael Langer und Michael Nerb In ihren Dissertationen wird eine umfassende Customer-Service-Managementarchitektur sowie eine Methodik zur Anwendung dieser Architektur entworfen [Lang 01] und für das interorganisationale Dienstmanagement mit Hilfe eines allgemeinen und umfassenden Anforderungskatalogs [Nerb01] eingesetzt. Langer und Nerb nutzen ein eingeschränktes Dienstmodell (ausschließlich bilaterale Customer-Provider-Beziehungen). Zusätzlich zu den Aspekten, die Langer und Nerb beschreiben, wird in der vorliegenden Arbeit der Schwerpunkt auf das Fehlermanagement und den Vergleich der unterschiedlichen Konzepte gesetzt, die bei der Realisierung einer Managementarchitektur in Hinsicht auf die interorganisationalen Formen der Diensterbringung benutzt werden.

Michael Schiffers untersucht in seiner geplanten Habilitationsschrift [Schif 11] Verlässlichkeitsaspekte in Grids. Dazu definiert er entsprechende Metriken und Algorithmen, um globale Verlässlichkeitsaussagen aus lokalen Beobachtungen zu aggregieren. Eine auf das Fehlermanagement fokussierende Managementarchitektur ist jedoch nicht Gegenstand der Arbeit. Beide Arbeiten ergänzen sich deshalb auf natürliche Weise.

1.4. Vorgehensmodell

Abbildung 1.2 zeigt die Zusammenhänge zwischen den verschiedenen Teilen dieser Arbeit.

Grundlegende Begriffe

In Kapitel 2 werden grundlegende Begriffe für diese Arbeit definiert. Es werden allgemeine Begriffe in Zusammenhang mit IT-Managementarchitekturen definiert. Unterschiedliche interorganisationale Formen der Diensterbringung werden hier betrachtet, sowie Grundkonzepte im Bereich Fehlermanagement. Die in dieser Arbeit verwendete Begrifflichkeit bezogen auf Fehler, Fehlerklassifikation und Fehlermanagementarchitekturen, wird definiert.

Anforderungsanalyse

Kapitel 3 konzentriert sich hauptsächlich auf die Aufstellung der Anforderungen an eine interorganisationale Fehlermanagementarchitektur. Grundlegend für die Ermittlung der Anforderungen sind zwei Szenarien für interorganisationales Fehlermanagement, die im Detail beschrieben werden. Aufgrund dessen werden die beteiligten Rollen für das interorganisationale Fehlermanagement festgelegt und beschrieben. Des Weiteren wird die Analyse mehrerer Anwendungsfälle (*use cases*) durchgeführt, was zur Ableitung von Anforderungen führt. Bei der Beschreibung der Anwendungsfälle werden betrachtet parallel die beiden Formen der interorganisationalen Diensterbringung, um gezielt gemeinsame bzw. spezifische Anforderungen abzuleiten. Ziel dieses Kapitels ist die Erstellung eines Anforderungskataloges für eine interorganisationale Fehlermanagementarchitektur (ioFMA).

Verwandte Arbeiten

Kapitel 4 beschreibt Vorarbeiten, die zur Realisierung einer ioFMA herangezogen werden können. Die Erfüllbarkeit der bereits ermittelten Anforderungen durch die bestehenden Ansätze wird auch evaluiert.

Entwurf Managementarchitektur

Im Kernkapitel (Kapitel 5) wird schrittweise eine Managementarchitektur für interorganisationales Fehlermanagement entworfen. Als Entwurfsmethodik wird die Model Driven Architecture (MDA) herangezogen. Grundlage sind die im Kapitel 3 beschriebenen Anwendungfälle und daraus ermittelten Anforderungen. Auf deren Basis werden unter der Verwendung von UML vier Teilmodelle entwickelt: Informationsmodell, Organisationsmodell, Kommunikationsmodell und Funktionsmodell. Die Reihenfolge des Entwufs der Teilmodelle hat ihren Ursprung ebenfalls in der Anforderungsanalyse. Es wird zunächst das Organisationsmodell realisiert, gefolgt vom auf den Anwendungsfällen basierenden Funktionsmodell. Daraus werden die für das Informationsmodell wichtigen Entitäten und Informationen extrahiert. Das Kommunikationsmodell baut darauf auf. Ein Mehrwert dieser Arbeit stellt der Vergleich zwischen den unterschiedlichen Diensterbringungsmodellen im Bezug auf die entwickelte Managementarchitektur dar.

Plattformspezifische Transformation

Kapitel 6 beschäftigt sich mit der Transformation der Managementarchitektur auf die Systemsicht. Es werden hier Transformationsregeln erstellt, die die ioFMA von einer allgemeingültigen Managementarchitektur auf eine plattformspezifische ioFMA abbilden. Das plattformspezifische Modell wird beispielhaft auf der Basis von Web-Services realisiert. Es werden dabei zwei Teile betrachtet: die Metamodelltransformation (von UML- auf WSDL) und die Transformation der ioFMA selbst.

Deployment

Das plattformspezifische Modell wird im Kapitel 7 beispielhaft auf das LHC Compute Grid (LCG) Szenario angewendet.

Die Arbeit schließt im Kapitel 8 mit einer Zusammenfassung der präsentierten Inhalte, der erreichten Ziele und der noch offenen Fragen.

1.5. Ergebnisse der Arbeit

Auf Basis der adressierten Fragestellungen liefert diese Arbeit folgende Ergebnisse:

1. Einen umfangreichen Anforderungskatalog für das interorganisationale Fehlermanagement. Der Katalog wird anhand der Untersuchung realer Szenarien zusammengestellt, er adressiert differenziert unterschiedliche Formen der Diensterbringung.
2. Eine vollständige Managementarchitektur zur Unterstützung von interorganisationalem Fehlermanagement (ioFMA), mit allen Teilmodellen. Diese Architektur weist für unterschiedliche Formen der Diensterbringung unterschiedliche Charakteristiken auf.
3. Eine Deployment-Strategie, die die Nutzung der Managementarchitektur auf Systemsicht ermöglicht.
4. Eine prototypische Instantiierung einer Managementplattform für interorganisationales Fehlermanagement für das LHC Optical Private Network (LHCOPN).

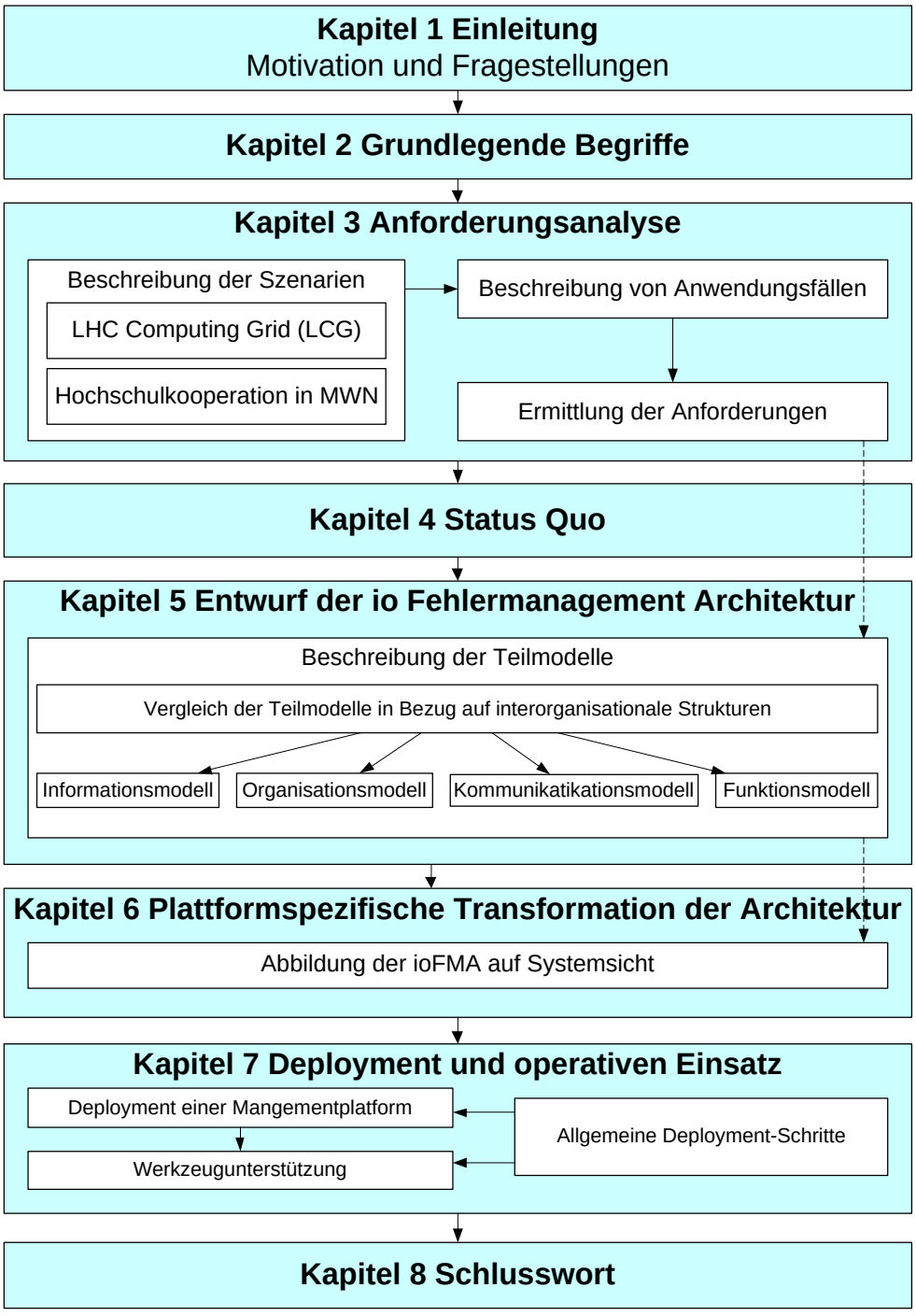


Abbildung 1.2.: Vorgehensmodell: Struktur dieser Arbeit

2

Grundlegende Begriffe und Methodik der Arbeit

Inhalt des Kapitels

2.1. Grundlegende Begriffe

Da in dieser Arbeit einem werkzeugorientierten Ansatz gefolgt wird, führt dieses Kapitel zunächst IT-Managementarchitekturen und ihre Teilmodelle ein (Abschnitt 2.1.1). In den folgenden Abschnitten wird der Fokus auf Fehler und deren Klassifikation (Abschnitt 2.1.2) gerichtet. Im Abschnitt 2.1.3 werden Formen der interorganisationalen Diensterbringung eingeführt. Anhand dieser werden im Laufe dieser Arbeit die interorganisationalen Fehlermanagementkonzepte verglichen.

2.1.1. IT-Managementarchitekturen

Eine Managementarchitektur, wie sie in [HAN 99] definiert wird, ist eine spezifische Ausprägung aller notwendigen Systemübergreifenden und herstellerunabhängigen Festlegungen, die zur Realisierung eines integrierten Managements in verteilten heterogenen Umgebungen führen. Bei der folgenden Beschreibung von Managementarchitekturen werden zwei Facetten betrachtet: die Realisierung der Managementlösung (Abschnitt 2.1.1.1) und die Teilmodelle der Managementarchitektur (Abschnitt 2.1.1.3).

2.1.1.1. Ansätze zur Realisierung von Managementlösungen

Wenn man sich mit werkzeugunterstützten Managementlösungen beschäftigt, kann man drei Ansätze betrachten: den *isolierten*, den *koordinierten* und den *integrierten Ansatz* (vgl. Abbildung 2.1).

isolierter Ansatz

Der isolierte Ansatz ist gekennzeichnet durch unabhängige Werkzeuge, die isoliert voneinander für unterschiedliche Managementprobleme zuständig sind. Werkzeuglandschaften, die unabhängig voneinander arbeiten, tragen zu einem isolierten Management (im Bezug auf Hersteller, Funktionsbereiche, Standards usw.) bei. Auch wenn es Versuche zur Integration dieser Werkzeuge in eine einheitliche Bedieneroberfläche gibt, so handelt es sich trotzdem um einen isolierten Ansatz. Der isolierte Ansatz gerade in komplexen, heterogenen, verteilten, multi-domänen Systemen wegen der Heterogenität der technischen und organisatorischen Gegebenheiten oft anzutreffen.

Nachteil für das interorganisationale Fehlermanagement: viele unterschiedliche Werkzeuge in unterschiedlichen Organisationen oder Domänen erschweren ein gesamtheitlicher organisationsübergreifendes Fehlermanagement.

koordinierter Ansatz

Bei dem koordinierten Ansatz koexistieren die Werkzeuge isoliert weiterhin voneinander, aber diese werden in einer koordinierten Weise eingesetzt. Das heißt, dass diese Werkzeuge sich gegenseitig ergänzen und dass eine gewisse Koordination zwischen den Werkzeugen über proprietäre, herstellerspezifische Schnittstellen eingerichtet wird. So kann z.B. die von

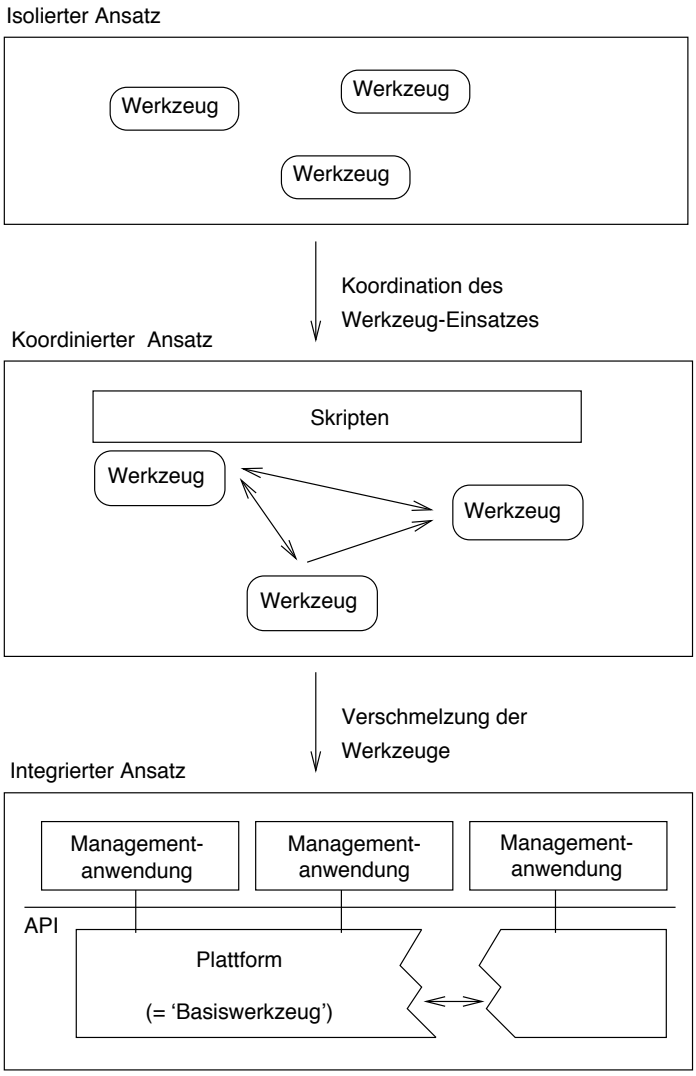


Abbildung 2.1.: Werkzeugunterstützte Managementlösungen nach Hegering et al. [HAN 99]

einem Werkzeug erzeugte Ausgabe als Input für ein oder mehrere andere Werkzeuge zur Verfügung stehen. Dies kann zusätzlich auch durch eine gemeinsame graphische Oberfläche unterstützt werden. Diese vereinigt z. B. durch ein koordiniertes Skript verschiedene Managementanwendungen und Werkzeuge.

Nachteil für das interorganisationale Fehlermanagement: zwischen den Werkzeugen der einzelnen Domänen müssen spezifische Schnittstellen zur Koordination realisiert werden. Bei jeder Änderung eines Werkzeugs in einer der Domänen muss die Implementierung wieder revidiert oder im schlimmsten Fall die ganze ,,Koordination" neu realisiert werden. Ebenfalls kann es im interorganisationalen Umfeld zu semantischen Widersprüchen zwischen den

Implementierungen unterschiedlicher Domänen kommen.

integrierter Ansatz

Der integrierte Ansatz setzt voraus, dass die Informationen zu den zu managenden Komponenten in einer herstellerunabhängigen Weise interpretiert werden. Dazu müssen diese Informationen über standardisierte Schnittstellen und Protokolle erreichbar sein.

2.1.1.2. Managementplattformen

Die Integration von Managementanwendungen liefert wird durch den Einsatz von Managementplattformen ermöglicht. Managementplattformen führen konzeptionell Netz-, System- und Anwendungsmanagement zusammen, so dass gemeinsame Grundfunktionen aus den verschiedenen Funktionsbereichen, organisatorische Vorgehensweisen, Zugriff auf gemeinsamen Ressourcen oder Datenbestände usw. *zentral* definiert werden. Plattformen sind nach Hegering et al. [HAN 99] *„Trägersysteme für Managementanwendungen"*, die auf andere Werkzeuge, Managementsysteme oder gemeinsame Ressourcen zugreifen können. Managementplattformen eignen sich hervorragend für den interorganisationalen Einsatz; diese Tatsache wird in der vorliegenden Arbeit für das interorganisationale Fehlermanagement genutzt. Im Abbildung 2.2 ist der Aufbau einer Managementplattform dargestellt.

Infrastruktur

Es handelt sich dabei nicht um eine traditionelle Infrastruktur, die zur Verbindung unterschiedlicher Komponenten in verteilten Systemen fungiert, sondern um die Logik der Plattform. Deswegen ist die Infrastruktur eigentlich die Plattform selbst. Sie besteht aus einem Kernsystem, das sowohl auf den Kommunikationsbaustein als auch auf den Baustein der Informationsverwaltung zugreift. Der Kommunikationsbaustein bietet Dienste, um auf entfernte Managementobjekte zugreifen zu können. Der Informationsbaustein hingegen realisiert den Zugriff auf die Datenbasis der eigenen Managementobjekte.

Oberflächenbaustein

Die Oberflächenbaustein ist derjenige Teil, der dem Benutzer die Interaktion mit den Funktionen und Anwendungen bzw. Managementapplikationen der Plattform ermöglicht. Das Graphical User Interface (GUI) beinhaltet Symbole, die die Managementobjekte graphisch darstellen. Die Eigenschaften der MOs (vgl. Abschnitt 2.1.1.3) werden in Form von Attributen festgelegt.

Basisanwendungen

Über eine Programmierschnittstelle Application Programming Interface (API) greifen Basisanwendungen auf das Kernsystem der Plattform-Infrastruktur zu. Zu diesen Basisanwendungen zählen im Falle von Netzmanagementplattformen unter anderem ein Konfigurationsmanager, ein Topologiemanager, ein Leistungsmonitor, ein MIB-Browser, ein Ereignismanager und ein Zustandsmonitor. Es handelt sich hierbei um vergleichsweise rudimentäre Anwendungen, die meistens sich auf eine Visualisierung der Informationen der Managementobjekte beschränken. Die Basisanwendungen reichen aber nicht aus, um ein großes oder komplexes verteiltes System zu managen. Daher setzen auf diese komplexere Managementapplikationen auf.

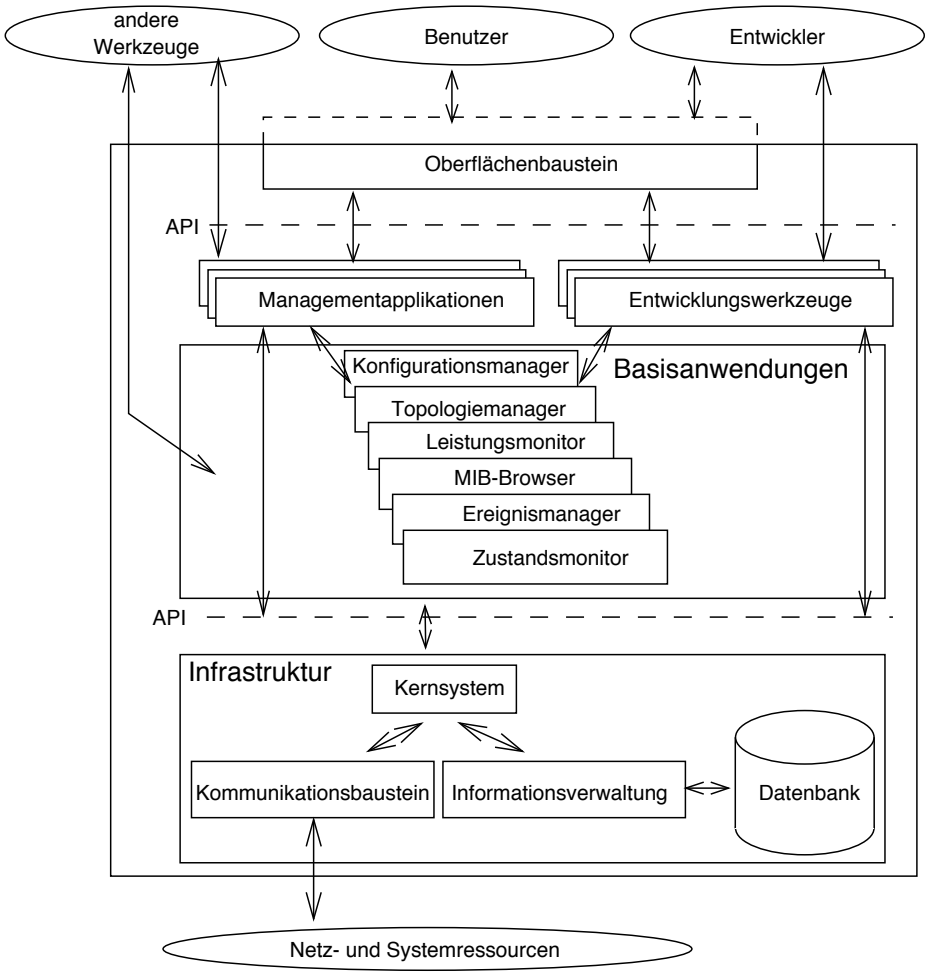


Abbildung 2.2.: Eine Übersicht über Managementplattformen [HAN 99]

Management-anwendungen

Die Managementanwendungen befinden sich im engen Kontakt mit den Basisanwendungen, deren Grundfunktionen sie benutzen. Sie sind aus Sicht des Benutzers des Systems der präsenteste Teil der Plattform, da diese näher am Anwender angesiedelt sind. In ihnen werden Funktionen realisiert und gebündelt, die sich aus den Anwendungsfällen des jeweiligen Managementbereichs ergeben (basierend auf dem Funktionsmodell) und über die graphische Oberfläche dem Benutzer angeboten werden.

Management-architektur

Für den Entwurf einer Managementplattform ist eine Managementarchitektur die Voraussetzung. Eine Managementplattform ist für ein integriertes Management in einer heterogenen Umgebung, wenn die Managementarchitektur in einer herstellerunabhängigen Weise definiert wird nur geeignet. Um eine Managementarchitektur realisieren zu können, müssen folgende Komponenten vorhanden sein: Informations-, das Organisations-, das Kommunikations- und das Funktionsmodell. Diese werden im nächsten Abschnitt ausführlicher beschrieben.

Nutzen für das io Fehlermanagement

Eine Managementplattform in einer interorganisationalen Umgebung ist ein Trägersystem für die Managementanwendungen, die zugleich auf andere Managementsysteme, Werkzeuge und Ressourcen zugreifen. Im Sinne der Realisierung einer integrierten Managementlösung konzentriert sich diese Arbeit auf die Entwicklung einer IT-Managementarchitektur bzw. einer Managementplattform, die für das interorganisationale Fehlermanagement eingesetzt werden sollen.

2.1.1.3. Teilmodelle

Da in der Realität sich verteilte Systeme sehr stark unterscheiden hinsichtlich ihres Aufbaus, der Größe und der Zielsetzung, kann es nicht nur eine Lösung für alle vernetzten Systeme geben. Daher ist ein Ziel einer allgemeiner IT-Managementarchitektur, eine optimale Managementlösung zu realisieren durch das Definieren eines Rahmens mit unterschiedlichen Modulen, die *„einzelne Managementproblembereiche bearbeiten und möglichst flexibel und interoperabel zusammengesetzt werden können"* [HAN 99].

Die vier zentralen Aspekte, die in Bezug auf IT-Managementarchitekturen betrachtet werden müssen, sind:

- Die Beschreibung von Managementobjekten (MOs) mit Hilfe eines **Informationsmodells**
- Die Beschreibung von Organisationsaspekten, Rollen und Kooperationsformen mittels eines **Organisationsmodells**
- Die Beschreibung von Kommunikationsvorgängen zu Managementzwecken durch ein **Kommunikationsmodell**
- Die Strukturierung der Managementfunktionalität innerhalb eines **Funktionsmodells**

Diese Teilaspekte werden in den nächsten Paragraphen erläutert.

Das Informationsmodell Managementobjekte sind Abstraktionen von Ressourcen, die aus Managementsicht relevant sind. Das Informationsmodell stellt einen Beschreibungsrahmen für Managementobjekte bereit. Dieser wird durch Festlegung eines einheitlichen Formats für Managementinformationen (Informationen, die zu Managementzwecken auszutauschen sind) realisiert. Die Gesamtheit der Managementobjekte wird in einer sogenannten Managementinformationsbasis (MIB) zusammengefasst. Die Struktur der MIB ist durch das Informationsmodell definiert. Eine MIB umfasst alle Managementschnittstellen, die ein Agent einem Manager zur Verfügung stellt. Abbildung 2.3 zeigt, wie nach [HAN 99] das Management durch MIB-Zugriffe realisiert werden kann. Es gibt unterschiedliche Quellen für die Festlegung von Managementinformationen: Ressourcen, Verfahren, Hersteller, Betreiber, Kunden. Bei der Gewinnung von relevanten Managementinformationen können zwei Vorgehen verfolgt werden: ein *Bottom-up-* und ein *Top-down-Vorgehen.*

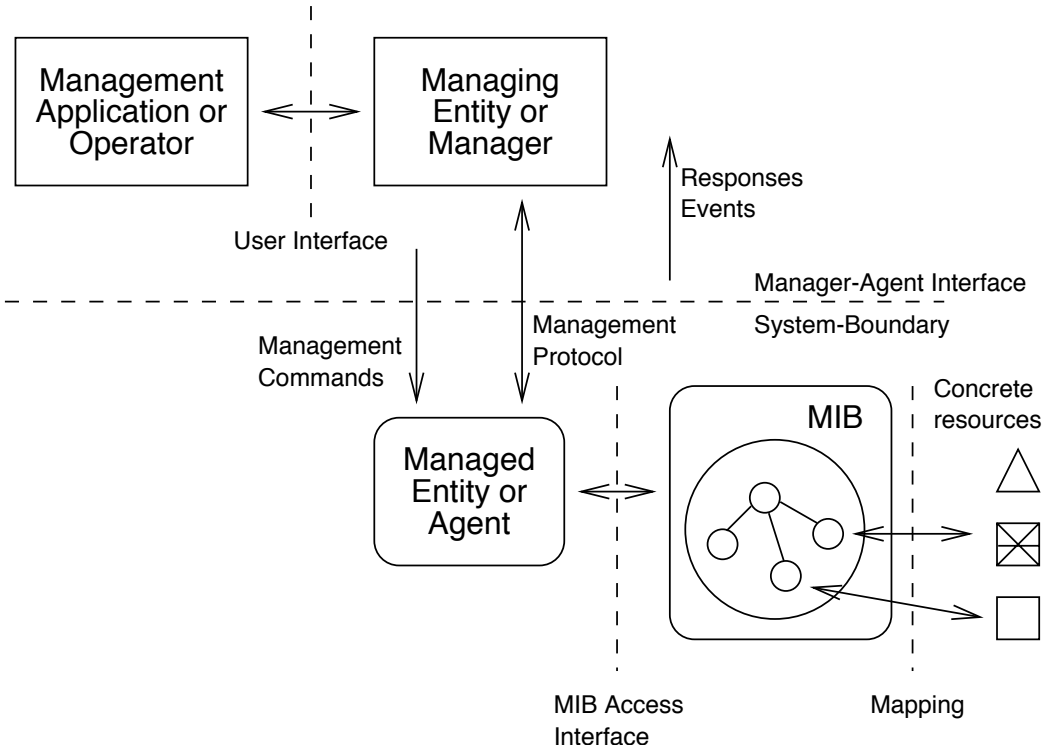


Abbildung 2.3.: Management durch MIB-Zugriff [HAN 99]

Bei einem Bottom-up-Vorgehen geht man von den Informationen, die zur Verfügung stehen, aus; die relevanten Managementinformationen werden aus vorgegebenen Protokollen, Komponenten und Produktvorgaben gewonnen. Bei einem Top-down-Vorgehen werden Informationen aus den Anforderungen von Managementanwendungen (z. B. Algorithmen, Prozessen, Verfahren), Nutzern (z.B. Informationsbedarf, SLAs) oder Betreibern (z.B. organisatorische Information, Prozessworkflows) abgeleitet. Es handelt sich dabei um relevante Managementinformationen, die benötigt werden, um die MOs zu definieren.

Diese Arbeit folgt einem Top-down-Vorgehen zur Gewinnung der relevanten Managementinformationen und Definition von MOs. Es werden dabei die Referenzprozesse für das Incident Management aus [OGC 07d] und [Hamm 09] aus Prozesssicht betrachtet und daraus die für das Informationsmodell relevanten Abstraktionen abgeleitet.

Das Organisationsmodell

Das Organisationsmodell definiert Rollen, Zuständigkeitsbereiche (Domänen), Gruppenbildungen und Kooperationsformen, die am Managementprozess beteiligt sind. Eng mit dem Begriff der Domäne hängt der Begriff der *Policy* zusammen. Policies werden aus übergeordneten Zielen bzw. Prozessen als Vorgaben für das technische Management abgeleitet und sind hiermit auf verschiedenen Ebenen des IT-Managements zu finden.

Systeme können aktive Rollen übernehmen und andere Systeme als *Manager* bzw. *Managementsystem* steuern, oder sie können eine passive Rolle haben, indem sie von anderen Systemen administriert werden als *Agenten* bzw. *Agententensysteme*.

IT-Managementarchitekturen können, für asymmetrische (hierarchische) oder symmetrischen Kooperationsformen, aufgestellt werden [HAN 99]. Im Abschnitt 2.1.3 wird vertieft auf die Unterschiede in den Kooperationsformen eingegangen.

Das Kommunikationsmodell

Das Kommunikationsmodell spezifiziert Prinzipien und Konzepte zum Austausch von Managementinformationen zwischen den im Organisationsmodell definierten Rollen. Das Kommunikationsmodell muss folgende Aspekte betrachten:

- welche Partner zur Kommunikation berechtigt sind
- welche Kommunikationsmechanismen zum Austausch relevanter Managementinformationen verwendet werden (Protokoll- und Dienstspezifikation)
- welche Syntax und Semantik für die Protokoll-Datenstrukturen benutzt werden

Der Informationsaustausch in Managementarchitekturen kann im *Pull-Modus*, im *Push-Modus* oder in einer hybriden Form erfolgen. Im *Pull-Modus* werden die Informationen seitens des Managementsystems beim Agenten abgefragt. Im *Push-Modus* werden die Informationen von den Agenten selbständig gesendet ohne dass eine vorherige Aufforderung seitens des Managementsystems notwendig ist.

Das Funktionsmodell

Das Funktionsmodell teilt die Gesamtaufgaben des Managements in Management-Funktionsbereiche ein und legt allgemeine Managementfunktionen fest. Die Funktionsbereiche sind gemäß OSI: Fehler-, Konfigurations-, Abrechnungs-, Leistungs-, und Sicherheitsmanagement (aus dem Englischen: Fault, Configuration, Accounting, Performance, Security Management (FCAPS)).

Fehlermanagement

Das *Fehlermanagement* ist der Managementfunktionsbereich, der sich mit dem Finden, Eingrenzen und Beheben von Fehlern in einem zu managenden System beschäftigt. Eine ausführliche Klassifikation von Fehlern folgt im nächsten Abschnitt. Insbesondere in verteilten, vernetzten Systemen wird die Fehlererfassung und Fehlerverfolgung durch die große Anzahl an Komponenten, die weite räumliche Verteilung der Ressourcen sowie die Heterogenität der Hardware- und Softwarekomponenten erschwert.

Die Hauptaufgabe des Fehlermanagements besteht darin, eine hohe Verfügbarkeit eines Systems und der bereitgestellten Dienste durch schnelle Entdeckung und Behebung von Fehlern zu gewährleisten. Aus dieser Aufgabe ergeben sich u.a. folgende Teilaufgaben:

- Überwachung des Netz- bzw. Systemzustandes
- Aufnahme und Verarbeiten von Alarmen

- Diagnose der Fehlerursachen
- Erkennen von Propagation von Fehlern
- Einleiten und Überprüfen von Fehlerbehebungsmaßnahmen
- Führen eines Trouble-Ticket-Systems
- Hilfestellungen für den Benutzer

Die hier genannten Aufgaben beziehen sich in dieser Arbeit hauptsächlich auf vernetzte, verteilte Organisationen. Das Fehlermanagement in interorganisationalen Umgebungen stellt den Kern dieser Arbeit dar. Im Gegensatz zum intraorganisationalen Fehlermanagement sind weitere Problemfelder zu betrachten, wie unterschiedliche intraorganisatorische Abläufe zur Fehlererfassung und -verfolgung, unterschiedliche Definitionen von Fehlern, unterschiedliche Werkzeuge für das Fehlermanagement, die verschiedenen Kooperationsformen zwischen den Provider der unterschiedlichen Organisationen als auch unterschiedliche Formen der interorganisationalen Diensterbringung. Diese Arbeit erweitert die intraorganisationalen Prinzipien des Fehlermanagements auf interorganisationale Umgebungen.

Konfigurationsmanagement

Das *Konfigurationsmanagement* umfasst die Beschreibung eines vernetzten Systems mit seinen Komponenten, die Konfiguration als Aktivität in Bezug auf die Änderung der Struktur des Systems als auch das Ergebnis dieser Aktivität. Damit umfasst das Konfigurationsmanagement das Setzen von Parametern, das Festlegen von Schwellwerten und Filtern, die Dokumentation des Gesamtsystems und seiner Evolution sowie das aktive Ändern der Konfiguration.

Abrechnungsmanagement

Das *Abrechnungsmanagement* ist zuständig für die Umlage der Kosten für bereitgestellte Kommunikations- oder sonstige Dienste, die in der Regel von den Verursachern zu tragen sind. Die Aufteilung erfolgt nach Abrechnungs-Policies. Teilaufgaben des Abrechnungsmanagements sind die Festlegung von Abrechnungsdaten, die Führung von Abrechnungskonten, die Zuordnung von Nutzungskennzahlen Kosten zu Konten, die Führung von Verbrauchsstatistiken sowie die Festlegung von Abrechnungs-Policies und die Aushandlung sowie Tarifen.

Leistungsmanagement

Das *Leistungsmanagement* kann als konsequente Weiterführung des Fehlermanagements gesehen werden. Zusätzlich zum Fehlermanagement, das für den störungsfreien Betrieb des Systems verantwortlich ist, ist das Leistungsmanagement für die Dienstqualität zuständig. Die Dienstgüte (engl. Quality of Service (QoS)) wird durch Dienstgüteparameter zwischen Provider und Kunden, in Form vom Service Level Agreements (SLAs), festgelegt. Teilaufgaben des Leistungsmanagements sind die Bestimmung von QoS-Parametern und Metriken zur Überwachung der Dienstgüte, das Monitoring von Ressourcen, die Durchführung von Messungen, die Aufzeichnung von Systemprotokollen, die Aufbereitung und Aggregation von Messdaten sowie die Durchführung von Leistungs- und Kapazitätsplanungen sowie von Anpassungen.

Sicherheits-management

Das *Sicherheitsmanagement* bezeichnet das Management der Sicherheit in einem System und umfasst alle Maßnahmen zur Gewährleistung eines sicheren und geschützten verteilten Systems, indem die schützenswerten Ressourcen einer Organisation (Infrastrukturen, Informationen, Dienstleistungen) vor unautorisierten Zugriffen abgeschirmt werden.

2.1.2. Fehler und Klassifikation von Fehlern

Ein Fehler ist nach [HAN 99] definiert als eine Abweichung von gesetzten Betriebszielen, Systemfunktionen oder Diensten. Fehler werden von überwachten Komponenten oder von den Benutzern der Systeme gemeldet.

Laprie et al. definiert Fehler in [ALR 01] und erweitert die Begriffsbildung in [ALRL 04] als Bedrohungen (Threats) der Verlässlichkeit (Dependability) von IT-Systemen. Laprie teilt diese in mehrere Fehlerklassen ein. In diesem Kontext wird ein System als Einheit, die mit anderen Systemen, Hardware, Software, Menschen und der physischen Welt interagiert, definiert. Rechen- und Kommunikationssysteme besitzen grundlegende Eigenschaften: Funktionalität, Leistung, Verlässlichkeit, Sicherheit und Kosten. Ein Dienst, der von einem System geliefert wird, wird als das Verhalten des Systems definiert, so wie es die Nutzer wahrnehmen. Laut diesem Ansatz implementiert ein System mehrere Funktionen und stellt mehrere Dienste bereit. Funktionen und Dienste können als zusammengesetzte Gebilde angesehen werden, die aus Funktionsbausteinen bzw. Dienstbausteinen ,,aufgebaut" sind.

In Abschnitt 2.1.2.1 werden Bedrohungen beschrieben, es folgen in Abschnitt 2.1.2.2 die Attribute der Verlässlichkeit und in Abschnitt 2.1.2.3 eine Aufzählung der Maßnahmen zur Sicherung der Verlässlichkeit. Anschließend wird zusammengefasst, was bezüglich der Klassifikation von Fehlern, in dieser Arbeit betrachtet wird.

2.1.2.1. Bedrohungen der Verlässlichkeit

Man spricht von einem funktionsfähigen Dienst (*engl. correct service*), wenn er die Systemfunktionen entsprechend der Spezifikation der Dienste implementiert. Dieser Abschnitt beschäftigt sich mit Abweichungen von der Spezifikation während des Betriebes. Fehler werden laut [ALRL 04] in Failures, Errors und Faults eingeteilt.

Failures

Ein Dienstausfall (*engl. service failure*) oder kurz ein Failure ist ein Ereignis als Folge einer Abweichung des Dienstes von seiner Spezifikation. Ein Dienstausfall ist der Übergang von einem funktionsfähigen Dienst zu einen nicht-funktionsfähigen Dienst.

Da wir uns in dieser Arbeit hauptsächlich auf Failures (Ausfälle) konzentrieren werden, wird eine ausführlichere Beschreibung bzw. Klassifikation von Failures benötigt.

Es gibt drei Kategorien von Ausfällen:

A. Störungen (service failures)

Wie oben beschrieben, repräsentieren Störungen Abweichungen vom funktionsfähigen Dienst. Eine Abweichung vom funktionsfähigen Dienst kann mehrere Formen einnehmen (*service failure modes*) und jede Form kann eine oder mehrere Ernsthaftigkeitsgrade (*service failure severity*) haben. Die Störungsformen können aus folgenden Perspektiven beschrieben

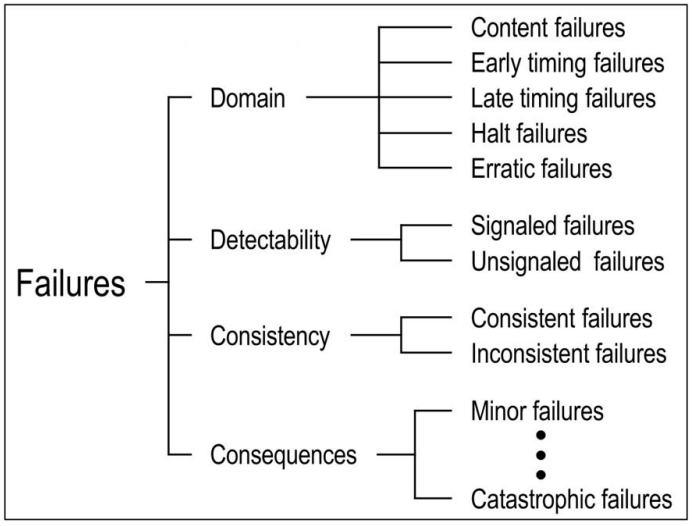


Abbildung 2.4.: Störungsformen (*service failure modes*) nach [ALRL 04]

werden:

Störungsbereich (failure domain) Bezüglich deren Wirkungsbereich ergeben sich folgende Klassen von Störungen:

- Inhaltliche Störungen (*content failures*): Der Inhalt der den Nutzern bereitgestellten Information weicht von der Spezifikation der Systemfunktionalität ab.
- Zeitbezogene Störungen (*timing failures*): Der Zeitpunkt der Bereitstellung einer Dienstfunktion weicht von der Spezifikation des Dienstes ab.
- Dienstausfälle (*halt failures*): Die Bereitstellung der Dienstfunktionalität endet.
- Fehlverhalten (*erratic failures*): Der Dienst ist bereitgestellt, aber er ist fehlerhaft.

Die Störungserkennung (detectability of failures) bezieht sich auf die Benachrichtigung (Alarmierung) des Nutzers bezüglich einer Störung. Hierbei existieren zwei Klassen von Störungen:

- Signalisierte Störungen (*signaled failures*): wenn die darunterliegende Degradierung des Dienstes entdeckt und zum Nutzer weitergeleitet wird

- Nicht-signalisierte Störungen (*unsignaled failures*): wenn es keine Signalisierung bzgl. des gestörten Dienstes gibt

Die Störungskonsistenz (consistency of failures) Hier gibt es zwei Klassen von Störungen:

- Konsistente Ausfälle (consistent failures): Alle Nutzer nehmen die Störung auf dieselbe Art und Weise wahr.
- Inkonsistente Ausfälle (inconsistent failures): Die Nutzer haben unterschiedliche Wahrnehmungen der Störung. Für manche Nutzer gibt es u. U. sogar keine Störung.

Die Störungstragweite (consequences of failures) kann für die Umgebung bzw. für den Nutzer unterschiedlich sein. Es werden Ernsthaftigkeitsgrade eingeführt, die die Auswirkungen und die Dringlichkeit einer Störung berücksichtigen. Man kann daher die Störungen in unterschiedliche Klassen einteilen von minimalen Ausfällen (*minor failures*) bis hin zu katastrophalen Ausfällen (*catastrophic failures*)

Abbildung 2.4 stellt die vier Formen der Dienstausfälle sowie deren Unterklassen dar.

B. Scheitern der Entwicklung (development failures)

Entwicklungsfehler (development faults) werden in der Entwicklungsphase durch ein Fehlverhalten der Beteiligten, Umgebungsbedingungen, Entwicklungswerkzeuge und andere Ursachen eingeführt. Als Folge davon können partielle oder totale Entwicklungsausfälle auftreten. Ein totaler Entwicklungsausfall trägt dazu bei, dass der Entwicklungsprozess frühzeitig beendet wird, schon bevor das System überhaupt produktiv ist. Partielle Entwicklungsausfälle beziehen sich auf eine fehlerhafte finanzielle und/oder zeitliche Planung als auch auf Dienste, die zwar produktiv sind, aber nur mit einer eingeschränkten Funktionalität.

C. Verlässlichkeits- und Sicherheitsstörungen (dependability and security failures)

Verlässlichkeits- und Sicherheitsstörungen treten auf, wenn das System sehr häufig oder schwerwiegende Störungen erlebt. Diese Kategorie von Störungen ist eine ,,Oberklasse" für die obengenannten.

Errors Bei einer Störung weichen ein oder mehrere Zustände von der Spezifikation des funktionsfähigen Dienstes ab. Diese Abweichung wird als Error bezeichnet.

Faults Die mögliche oder bekannte Ursache eines Errors ist ein Fault. Faults können systemintern oder -extern auftreten. Bevor ein Fault auftritt, existiert immer eine sogenannte Schwäche im System. In den meisten Fällen verursacht ein Fault zunächst einen Error im Zustand einer Komponente (also einen Teil des internen Zustandes des Systems) und der externe Zustand des Systems ist nicht sofort beeinträchtigt.

Faults und Errors werden in mehrere Klassen klassifiziert, die aber hier nicht mehr betrachtet werden, da diese nicht als zentraler Punkt dieser Arbeit betrachtet werden.

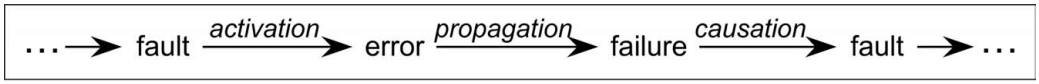


Abbildung 2.5.: Die Kette der Verlässlichkeit (*chain of dependability*) nach [ALRL 04]

Kette der Bedrohungen

Viele Errors werden nie außerhalb des Systems im Form von Failures wahrgenommen. Ein Fault ist erst dann *aktiv*, wenn er einen Error provoziert, ansonsten ist er *inaktiv* (*dormant*). Die Abbildung 2.5 zeigt die Kette der Verlässlichkeit (*chain of dependability*). Die Aktivierung eines Faults (*fault activation*) ist die jenige Aktion, die verursacht, dass ein Fault in den Zustand *aktiv* geht und zu einem Error führt. Die Error-Verbreitung (*engl. error propagation*) kann intern (innerhalb einer Komponente) oder extern (von einer Komponente zu einer anderen) erfolgen. Ein Failure tritt ein, wenn ein Error bis an die Dienstschnittstelle verbreitet wird, und provoziert eine Abweichung vom funktionsfähigen Zustand. Dieser Failure kann wiederum einen permanenten oder vorübergehenden Fault verursachen.

2.1.2.2. Attribute der Verlässlichkeit

Verlässlichkeit ist definiert als die Fähigkeit des Systems, aufgetretene Störungen zu vermeiden. Verlässlichkeit weist folgende Attribute auf:

- Verfügbarkeit (availability): die Eigenschaft des funktionsfähigen Dienstes erreichbar (verfügbar) zu sein.
- Ausfallsicherheit (reliability): die Eigenschaft der Kontinuität des funktionsfähigen Dienstes. Es ist eine qualitative Eigenschaft; ein Dienst kann z.B. verfügbar sein, aber mit schlechter Qualität; in diesem Fall ist der Dienst nicht ausfallsicher.
- Vertraulichkeit (confidentiality): die Eigenschaft, Information nur für Befugte zugänglich zu machen. Unbefugte haben dagegen keinen Zugang zu einer übertragenen Nachricht oder gespeicherten Information.
- Sicherheit (safety): Abwesenheit von Schadwirkungen des Systems gegen die Nutzer oder die Umgebung.
- Integrität (integrity): Integrität umfasst den Schutz vor Verlust und die Fälschungssicherheit der Daten.
- Wartbarkeit (maintainability): Fähigkeit des Systems geändert oder gewartet zu werden.

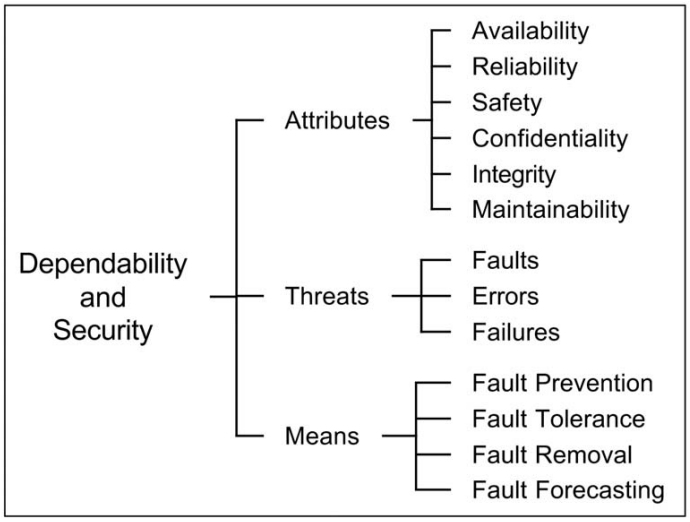


Abbildung 2.6.: *Dependability and Security Tree* nach [ALRL 04]

2.1.2.3. Maßnahmen zum Erhalt der Verlässlichkeit

Unter Fehlervorbeugung (*engl. fault prevention*) werden Maßnahmen zur Vorbeugung gegen das Auftreten und Einführen von Fehlern verstanden.

Fehlertoleranz (*engl. fault tolerance*) ist die Eigenschaft eines Systems, seine Funktionsweise auch in Anwesenheit von Fehlern aufrechtzuerhalten.

Fehlerbehebung (*engl. fault removal*) beinhaltet Maßnahmen zur Senkung der Anzahl und der Ernsthaftigkeit der Faults.

Unter Fehlervorhersage (*engl. fault forecasting*) werden Maßnahmen zur Schätzung der gegenwärtigen Anzahl, des zukünftigen Auftretens und möglicher Folgen von Faults verstanden.

Laut Laprie et. al kann zusammenfassend zu den oben genannten Eigenschaften ein sogenannter *Dependability and Security-Baum* dargestellt werden (siehe Abbildung 2.6) mit Attributen, Bedrohungen und Maßnahmen zur Erhalt der Verlässlichkeit.

2.1.2.4. Lebenszyklus eines Fehlers

In Bezug auf das Fehlermanagement in den untersuchten Szenarien (vgl. Abschnitt 3.1) wird der Lebenszyklus eines Fehlers betrachtet. In [HAN 99] ist dieser Lebenszyklus auf folgende Phasen zusammengefasst: Entdecken, Eingrenzen und Beheben. In [PaHa 07] wird eine Taxonomie von Fehlern in drahtlosen Netze definiert und dabei der Lebenszyklus eines Fehlers

mit folgenden Phasen in Betracht gezogen: Vorbeugen, Erkennen, Eingrenzen, Identifizieren und Wiederherstellen. Candea definiert in [Cand 03] die Phasen eines Lebenszyklus wie folgt: Erkennen, Diagnose, Eingrenzen, Reparieren und Wiederherstellen. In Laprie's Ansatz [ALRL 04] gibt es auch sogenannte Maßnahmen zum Erhalt der Verlässlichkeit (siehe Abschnitt 2.1.2.3), die in etwa dem Lebenszyklus eines Fehlers entsprechen: Fehlervorbeugung, Fehlertoleranz, Fehlerbehebung und Fehlervorhersage.

2.1.3. Formen der interorganisationalen Diensterbringung

Im Kontext der interorganisationalen Diensterbringung werden in diesem Abschnitt zwei zentrale Aspekte betrachtet: die Dienstbeziehungen im Abschnitt 2.1.3.1 sowie die Kooperation und die Koordination zwischen den Organisationen (bzw. Providern), ausgeführt in Abschnitt 2.1.3.2.

2.1.3.1. Dienstbeziehungen

Für die interorganisationale Diensterbringung sind die Dienstbeziehungen zwischen den unterschiedlichen Organisationen von großer Wichtigkeit. Ein Dienst ist eine Zusammensetzung von Teildiensten, die zu einen „Gesamtdienst" führen.

In ihrer Arbeit analysiert Dreo [Dreo 02] eine Unterscheidung zwischen zwei Formen von Dienstkomposition (siehe Abbildung 2.7), der vertikalen und der horizontalen Dienstkomposition.

Die Bereitstellung von Diensten wird sehr oft durch mehrere Subdienste realisiert. In der Abbildung 2.7 ist gezeigt, wie ein Web-Dienst auf einem IP-Dienst basiert. Die Dienstkomposition in diesem Fall entsteht dadurch, dass der Dienst (*Web*) und seine Subdienste (*IP*) sich auf unterschiedlichen Funktionsebenen befinden. Eine Diensthierarchie entsteht dabei durch mehrfache Anwendung der Dienst-Subdienst-Technik [GHH+ 02]. Die Integration erfolgt über Service Access Points (SAPs) der Subdienste.

Vertikale Dienstkomposition

Ein Dienst kann auch durch die Verkettung verschiedener Teildienste auf der gleichen Funktionsebene zusammengesetzt werden. Wie in Abbildung 2.7 dargestellt, wird der *IP-Dienst* durch Verkettung von mehreren *IP-Teildiensten* mehrerer Provider realisiert. Durch mehrfache Anwendung dieser Technik entsteht eine horizontale Dienstkette.

Horizontale Dienstkomposition

In der Praxis werden selten reine Formen dieser Dienstkompositionen gefunden, sondern zumeist eine Kombination von Sub- und Teildiensten. Schiffers bezeichnet dies als diagonale Dienstkomposition [Schif 07].

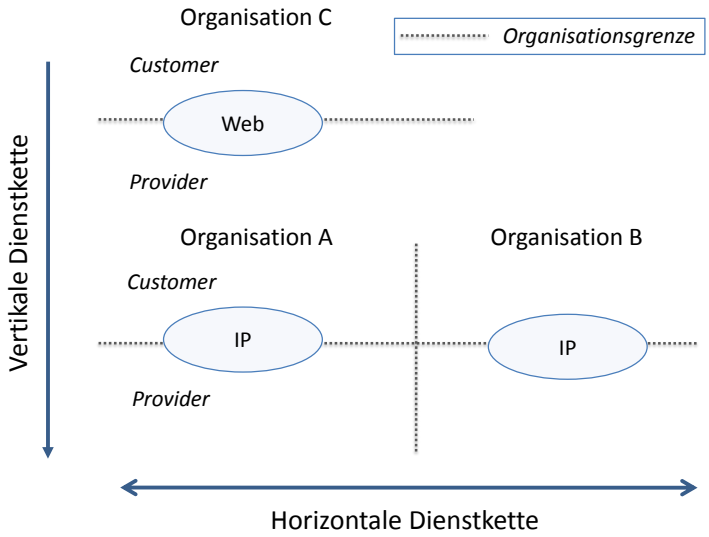


Abbildung 2.7.: Dienstbeziehungen nach [Dreo 02]

Mehrere Subdienste können innerhalb einer Organisation erbracht werden. Im Fall von komplexen IT-Diensten greifen die Provider auf Dienste anderer Provider zurück und stellen eigene Dienste als eine Kombination von Sub- und/oder Teildiensten anderer Provider bereit.

2.1.3.2. Koordination

Bei der interorganisationalen Diensterbringung ist die Koordination zwischen den Organisationen eine grundlegende Notwendigkeit. Die Darstellung der organisatorischen Aspekte in der Erbringung von IT-Diensten in diesen sowie in den nächsten zwei Abschnitten ist weitgehend [Hamm 09] entnommen.

Hamm beschreibt die vielfältigen Formen der Koordination [Hamm 09] mit dem sogenannten Koordinationswürfel (vgl. Abbildung 2.8). Der Koordinationswürfel basiert auf drei Dimensionen: Steuerung, Kommunikation und Aufgabenzuordnung. Es ergeben sich dabei Koordinationsmuster, die zwischen den zwei extremen Formen der Hierarchie und Heterarchie angesiedelt sind. Hamm benutzt den Koordinationswürfel als Mittel zur Kategorisierung der organisatorischen Randbedingungen der Providerzusammenarbeit. Für jeden ITSM-Prozess kann i. A. genau ein Koordinationsmuster bestimmt werden.

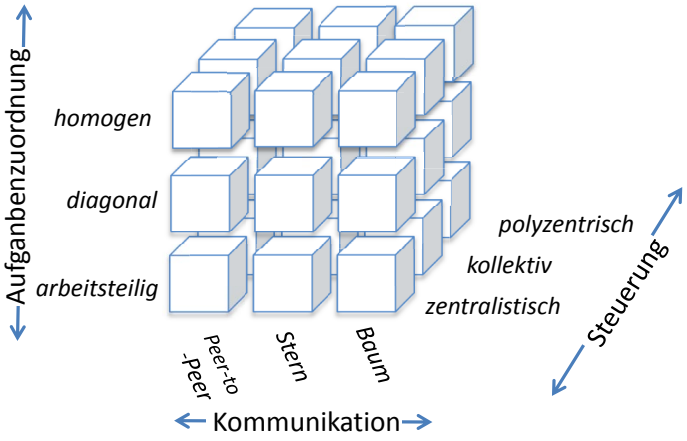


Abbildung 2.8.: Der Koordinationswürfel nach [Hamm 09]

Dimensionen der Koordination

Laut einer Definition aus der Organisationstheorie ist die Koordination die Abstimmung von Einzelaktivitäten über mehrere Organisationseinheiten oder Organisationen im Hinblick auf ein oder mehrere übergeordnete Gesamtziele [Vahs 05]. Auf der Grundlage der Arbeiten von Malone zur Koordinationstheorie [Malo 87] und Hedlund zu Heterarchien [Hedl 05] definiert Hamm drei Dimensionen zur strukturellen Analyse von Koordinationsstrukturen.

Steuerung erfasst den Aspekt, Entscheidungen im Rahmen der Zusammenarbeit zu treffen und durchzusetzen. Die Steuerung kann zentralistisch, kollektiv oder polyzentrisch sein. Eine zentralistische Steuerung liegt vor, wenn nur eine Organisation die Steuerung übernimmt. Im Fall einer kollektiven Steuerung wird diese mittels Teamwork, Aufgabeneinteilung und Rotation der Aufgaben (auch der Koordinatorenrollen) organisiert und basiert auf der Anwendung demokratischer Verfahren (z.B. Abstimmungen). Bei einer polyzentrischen Steuerung gibt es keine gemeinsame Entscheidungsfindung, sondern die beteiligten Organisationen besitzen völlige Entscheidungsautonomie.

Aufgabenzuordnung bezieht sich auf die Aufteilung einzelner Aktivitäten auf die beteiligten Organisationen. Eine arbeitsteilige Aufgabenzuordnung liegt vor, wenn eine funktionale Dekomposition und Zuordnung einzelner Aktivitäten auf die Organisationen existiert. Wenn mehrere Organisationen dieselben Einzelaktivitäten durchführen, spricht man von einer homogenen Zuordnung. Eine Kombination der beiden wird als diagonale Aufgabenzuordnung bezeichnet.

Kommunikation ermöglicht die Verteilung der Informationen in den beteiligten Organisationen und ist bei der Umsetzung der Entscheidungen als auch bei der Durchführung von Aufgaben unabdingbar. Es wird zwischen drei Grundformen von Kommunikationsstrukturen unterschieden:

Baum-Struktur In einem Baum gibt es eine ,,Wurzel"-Organisation, die mit einer oder mehreren Organisationen kommuniziert. Diese kommuniziert wiederum mit einer oder mehreren Organisationen. Zyklen sind im Baum nicht zulässig.

Stern-Struktur In einer Stern-Struktur ist eine Organisation das Kommunikationszentrum und kommuniziert mit allen anderen Organisationen.

Peer-to-Peer-Struktur Im Fall einer Peer-to-Peer-Struktur kann jede Organisation mit jeder anderen Organisation kommunizieren.

Koordinationsmuster Ein Koordinationsmuster ist ein Tripel der obengenannten Dimensionen der Koordination. Koordinationsmuster sind ein Mittel, um die zahlreichen Formen von Koordination, die in der Praxis der interorganisationalen Zusammenarbeit auftreten können, anhand weniger struktureller Eigenschaften zu kategorisieren. Das mögliche Spektrum der Koordinationsmuster breitet sich zwischen den zwei abstrakten Kooordinationsmustern Hierarchie und Heterarchie aus.

2.1.3.3. Hierarchie

Die grundlegende Eigenschaft einer Hierarchie ist die baumartige Organisationsstruktur, wie in Abbildung 2.9 dargestellt. Diese Struktur prägt alle Koordinationsdimensionen, insbesondere die Steuerung: In einer Hierarchie liegt zentralistische Steuerung vor. Die Organisation an der Wurzel des Hierarchiebaumes steuert alle untergeordneten Organisationen oder Organisationseinheiten. Diese steuern wiederum jeweils eine Reihe von untergeordneten Organisationen oder Organisationseinheiten. Also wird trotz der zentralistischen Steuerung eine Partitionierung der Autorität realisiert, um eine effektive Steuerung zu erreichen.

zentralistische Steuerung

Eine Hierarchie weist eine arbeitsteilige Aufgabenzuordnung auf, die üblicherweise mit der Struktur der Steuerung korrespondiert.

Baum Struktur Die Kommunikation zwischen den Organisationen wird üblicherweise auch in einer baumartigen Struktur (vorgegeben durch die Steuerung) realisiert. Die steuernde Organisation steht in direktem Kontakt zu den untergeordneten Organisationen oder Organisationseinheiten. Die Organisationen weisen aber im Allgemeinen keine zusätzlichen Kommunikationsbeziehungen auf.

vertikale Dienstkomposition Im Fall der Hierarchie liegt eine vertikale Dienstkomposition vor, wobei die Subdienste von den untergeordneten Organisationen bereitgestellt werden (siehe Abbildung 2.9).

2.1.3.4. Heterarchie

Bei einer Heterarchie liegt eine Organisationsstruktur vor, in der die beteiligten Organisationen gleichberechtigt sind. Im Gegensatz zur Hierarchie gibt es keine bestimmte einheitliche Ausprägung einer heterarchischen Organisation. Hedlund [Hedl 05] beschreibt eine

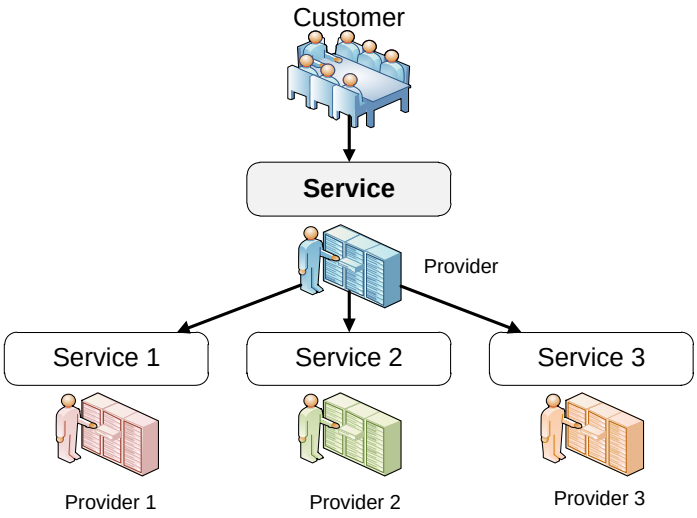


Abbildung 2.9.: Hierarchische Form der Diensterbringung

Reihe von Eigenschaften, die relevant für die heterarchische Form der Diensterbringung sind.

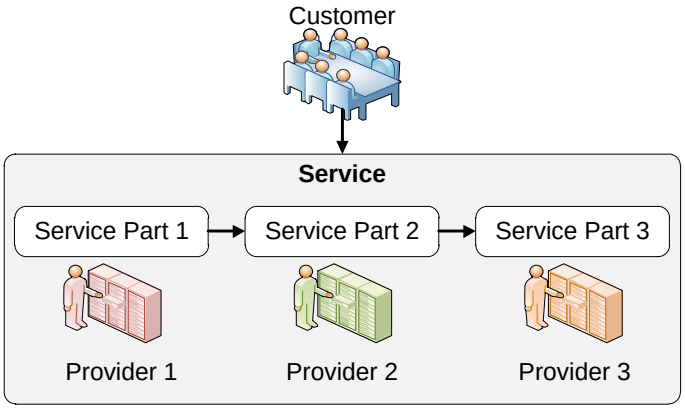


Abbildung 2.10.: Heterarchische Form der Diensterbringung

Asymetrie

Bezüglich der Steuerung und Kommunikation in einer Heterarchie gibt es keine einheitliche Ordnung. Es gibt vielfältige Kommunikationsbeziehungen zwischen den Organisationen, die unabhängig von der Dimension der Steuerung sind.

Zirkularität

Die Steuerungs- und Kommunikationsstruktur ist nicht zwingend transitiv wie in einer Hierarchie. Auch zirkuläre Strukturen sind im Extremfall möglich.

dynamische Ordnung

Die Ordnung der Organisationen in einer Heterarchie ist nicht statisch, sondern dyna-

misch und kann sich zu unterschiedlichen Zeitpunkten an veränderte Umstände anpassen.

gemeinsame Ziele

In einer Heterarchie herrscht nicht wie in einer Hierarchie eine zentralistische Steuerung; dennoch verfolgen die Mitgliedsorganisationen in einer Heterarchie gemeinsame Ziele und suchen nach einer gemeinsamen Entscheidungsfindung. Hiermit ist die Steuerung in Heterarchien kollektiv.

horizontale Dienstkomposition

In Heterarchien werden Dienste meist als horizontale Dienstketten realisiert. Dabei besteht ein Dienst aus mehreren Teildiensten, wie in [Dreo 02] erläutert. Ein zusammenfassende Darstellung für die Heterarchie wird in der Abbildung 2.10 gezeigt.

2.2. Präzisierung der Zielsetzung

Die vorliegende Arbeit beschreibt eine integrierte Managementlösung in Form einer Managementarchitektur zur Unterstützung des interorganisationalen Fehlermanagements. Es wurde schon zuvor die Vielfältigkeit der Aspekte, die daran beteiligt sind, beschrieben und aufgrund dessen der Anwendungsbereich der vorgeschlagenen Lösung wie folgt eingeschränkt.

Entsprechend dem Fokus dieser Arbeit, wird hier der Funktionsbereich Fehlermanagement (nach [HAN 99]) betrachtet.

Angelehnt an den *Dependability and Security-Baum*, erstellt von Laprie et. al in [ALRL 04], werden folgende Dimensionen betrachtet:

- Attribute: Verfügbarkeit (*availability*), Ausfallsicherheit (*reliability*), Vertraulichkeit (*confidentiality*), Integrität (integrity) und Wartbarkeit (*maintainability*)
- Bedrohungen: Ausfälle (*failures*) und davon nur die *Störungen* (*service failures*)
- Maßnahmen: Fehlertoleranz (*fault tolerance*), Fehlerbehebung (*fault removal*) und Fehlervorhersage (*fault forecasting*)

Es werden in Betracht gezogen die interorganisationalen Formen der Diensterbringung: Hierarchie und Heterarchie. Dabei werden zwei Hauptaspekte berücksichtigt: Dienstbeziehung (nach [Dreo 02]) und Koordination (nach [Hamm 09]).

Grundlegend für die Durchführung der Anforderungsanalyse und den Entwurf der Managementarchitektur sind die Incident-Management-Referenzprozesse für die hierarchische [OGC 07d] und heterarchische [Hamm 09] Form der Diensterbringung.

2.3. Methodik der Arbeit

Dieser Abschnitt besteht aus zwei Teilen: Im ersten Teil (Abschnitt 2.3.1) wird ein Übersicht über die Model Driven Architecture (MDA), die in der vorliegenden Arbeit die Vorgehensweise prägt, gegeben. Der zweite Teil fasst die grundlegenden Konzepte bezüglich der ioFMA Entwurfs zusammen (Abschnitt 2.3.2).

2.3.1. Model Driven Architecture

Das Vorgehen dieser Arbeit orientiert sich am Ansatz der von der Object Management Group (OMG) entwickelten modellgetriebenen Architektur (engl. MDA [OMG 01], [OMG 03]. Das schrittweise Vorgehen dieser Arbeit ist in Abbildung 2.11 dargestellt, die auch die Einordnung der MDA zeigt.

Mit ihrem Ursprung in der Softwareentwicklung beinhaltet MDA drei Modelle: CIM, PIM und PSM. Das Computation Independent Model (CIM) als das berechnungsunabhängige Modell beschreibt das gesamte System sowie seine Umgebung. Das Platform Independent Model (PIM) als plattformunabhängiges Modell ist ein technologie- und herstellerunabhängiges Modell. Das Platform Specific Model (PSM) oder das plattformspezifische Modell, im Gegensatz zu CIM oder PIM, berücksichtigt die Plattform, auf der eine Anwendung läuft, ist also ein technologieabhängiges Modell. Diese Modelle folgen einander von einem höheren (CIM) zu einem niedrigeren (PSM Abstraktionsgrad.

Analog zum CIM in der MDA werden in dieser Arbeit Szenarien beschrieben und daraus ein verallgemeinertes Szenario abgeleitet. Daraus folgen allgemeine Anforderungen an das Fehlermanagement in interorganisationalen Umgebungen und an eine Managementarchitektur. In der Entwicklungsphase wird in dieser Arbeit die Lösung von der Prozesssicht über die Architektursicht zur Systemsicht schrittweise aufgebaut. Auf der Prozesssicht sind die Referenzprozesse für das Incident Management (konform [OGC 07a] für die hierarchische Form der Diensterbringung und [Hamm 09] für die heterarchische) bereits gegeben. Die Notwendigkeit eines allgemeinen übergreifenden Referenzprozesses ist ebenso gegeben.

Die Architektursicht, entsprechend dem PIM im MDA-Ansatz, beinhaltet eine interorganisationale Fehlermanagementarchitektur (ioFMA). Die Referenzprozesse werden auf diese Sicht abgebildet.

Auf der Systemsicht wird auf der Basis der entwickelten Managementarchitektur ein plattformspezifisches Modell realisiert. An dieser Stelle werden Implementierungsaspekte für eine adäquate Managementplattform betrachtet.

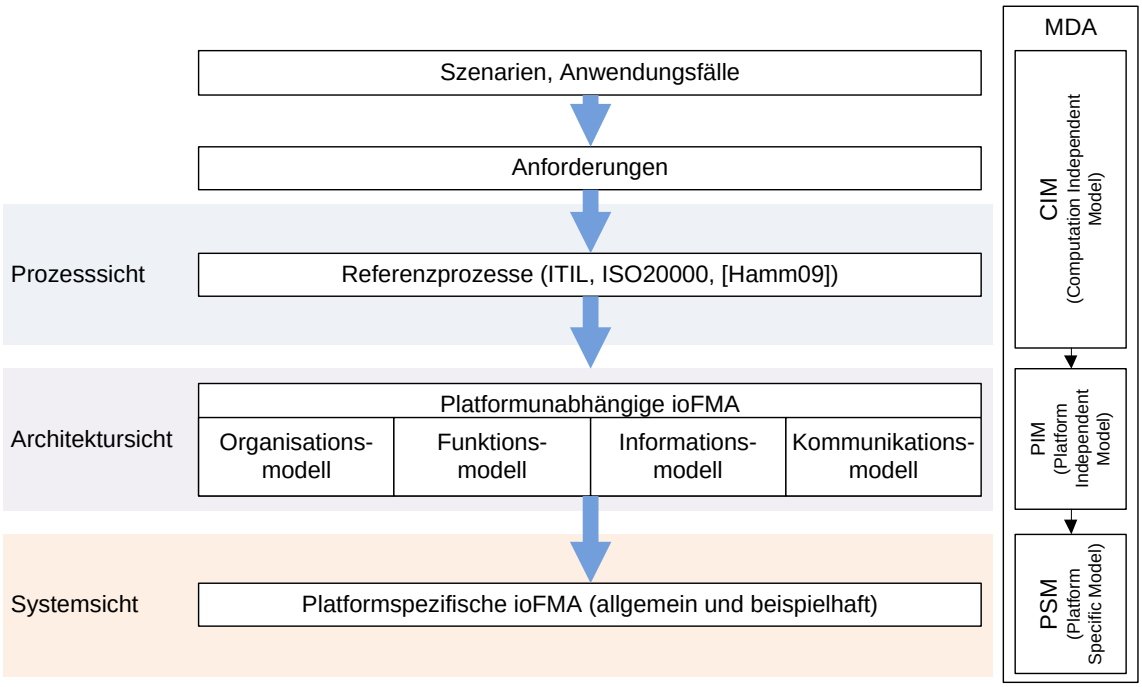


Abbildung 2.11.: Vorgehen dieser Arbeit im Kontext des MDA-Ansatzes

2.3.2. Systematik des Architekturentwurfs

Der Architekturentwurf besteht im Wesentlichen aus zwei wichtigen Bestandteilen, die als Vorgehen in dieser Arbeit betrachtet werden:

- Anforderungsanalyse: Wie werden die Anforderungen an die Architektur ermittelt?
- Modellentwurf: Wie wird die ioFMA auf der Architektursicht plattformunabhängig entworfen und instantiiert sowie danach plattformspezifisch auf Systemsicht transformiert?

In den folgenden Abschnitten werden die Vorgehensweisen der Anforderungsanalyse bzw. des Modellentwurfs kurz dargestellt.

Systematik der Anforderungsanalyse Abbildung 2.12 zeigt im oberen Teil die Systematik der Anforderungsanalyse. Aus zwei repräsentativen Szenarien (MWN und LCG) wird zunächst ein verallgemeinertes Szenario abstrahiert. Daraus werden allgemeine Anforderungen an das interorganisationale Fehlermanagement abgeleitet. In der Gesamtheit bilden diese das abzudeckende Funktionsspektrum. Daraus werden die Anwendungsfälle unter Berücksichtigung der vier Teilmodelle einer allgemeinen Managementarchitektur (vgl. Abschnitt 2.1.1) abgeleitet und beschrieben. Es werden dabei die Funktions- , Organisations-,

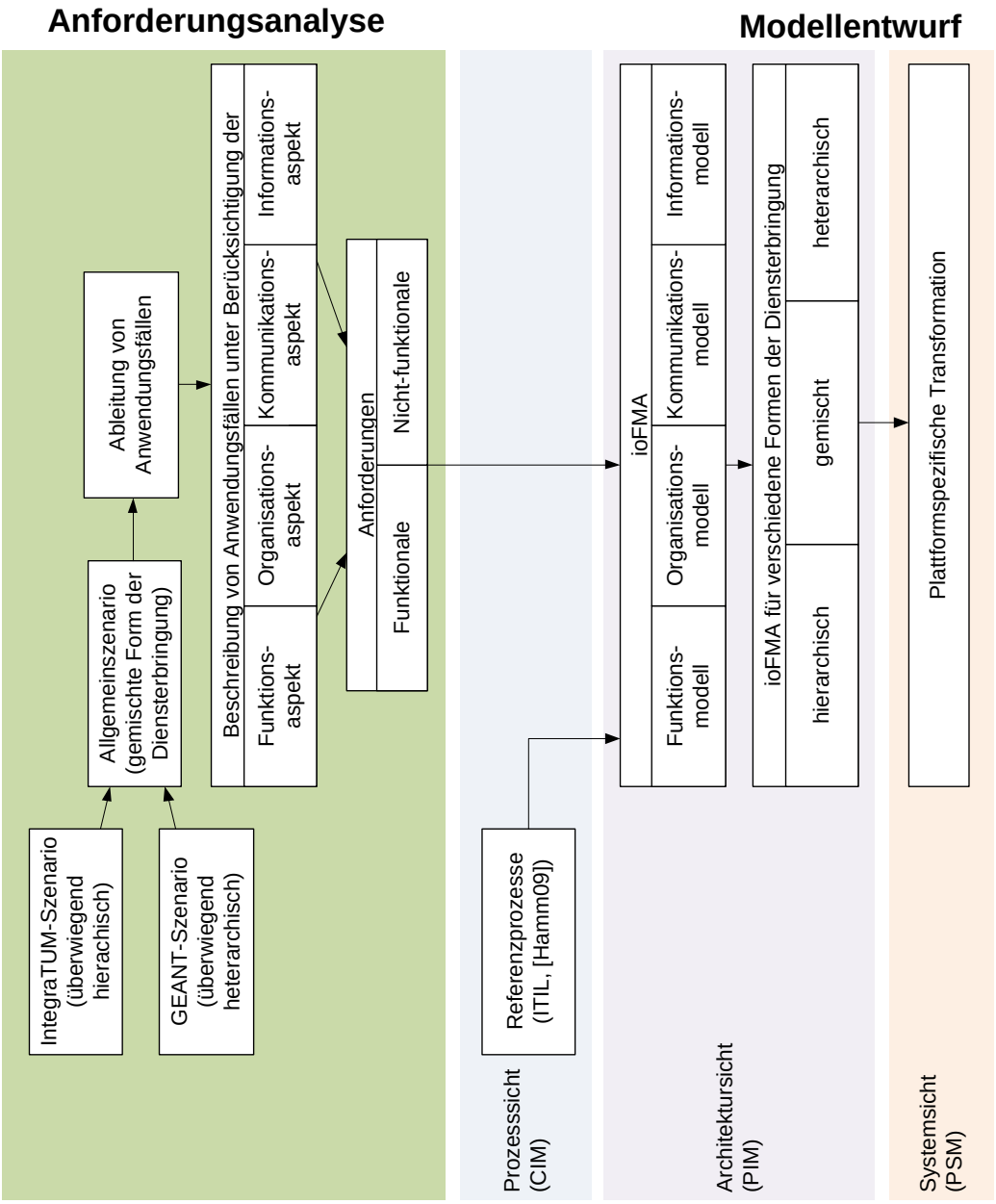


Abbildung 2.12.: Systematik der Anforderungsanalyse und des Modellentwurfs

Kommunikations- und Informationsaspekte der Anwendungsfälle betrachtet. Daraus werden funktionale und nichtfunktionale Anforderungen an die ioFMA abgeleitet, die den Modellentwurf leiten.

Systematik des Modellentwurfs Auf Basis der existierenden Referenzprozesse für das Incident Management und Problem Management zusammen mit den Anforderungen aus den Anwendungsfällen werden die Teilmodelle in folgender Reihenfolge entwickelt:

1. Funktionsmodell: Welche Funktionen sind für das Fehlermanagement notwendig?
2. Organisationsmodell: Welche Rollen und Verantwortlichkeiten werden im Umfeld benötigt?
3. Kommunikationsmodell: Wie werden Informationen im Umfeld des Fehlermanagements ausgetauscht?
4. Informationsmodell: Welche Managementinformationen sind für das Fehlermanagement notwendig?

Die ioFMA ist laut MDA-Ansatz ein plattformunabhängiges Modell, welches im nächsten Schritt für hierarchische, heterarchische und gemischte Formen der Diensterbringung instantiiert wird.

Diese drei Instantiierungen auf PIM-Ebene werden dann weiter plattformspezifisch instantiiert.

Der Gesamtablauf des Modellentwurfs nach dem MDA-Ansatz ist in den zwei unteren Kästen der Abbildung 2.12 dargestellt.

3 Anforderungsanalyse

Inhalt des Kapitels

3.1. Szenarien für interorganisationales Fehlermanagement

Die Ermittlung der Anforderungen an eine interorganisationale Fehlermanagementarchitektur (ioFMA) erfolgt in diesem Kapitel anhand von praxisbezogenen Szenarien. Die Szenarien müssen dabei zwei grundlegende Eigenschaften besitzen: Sie müssen eine interorganisationale Natur haben und unterschiedliche Formen der interorganisationalen Diensterbringung aufweisen. Im Abschnitt 3.1.1 wird der morphologische Kasten für die Einordnung der Szenarien beschrieben. Zwei Szenarien werden im Detail analysiert: Die Hochschulkooperation im Münchner Wissenschaftsnetz (MWN) im Rahmen des IntegraTUM-Projekts (Abschnitt 3.1.2) sowie ein Szenario aus dem Europäischen Umfeld – Géant End-to-End (E2E)-Links für das Large Hadron Collider (LHC) Projekt (Abschnitt 3.1.3). Im Abschnitt 3.1.4 wird ein verallgemeinertes Szenario für interorganisationales Fehlemanagement eingeführt, das die unterschiedlichen Eigenschaften der zwei vorher beschriebenen Szenarien verbindet.

3.1.1. Grundlegendes zur Einordnung der Szenarien

In diesem Abschnitt werden die Merkmale zur Einstufung der Szenarien dargestellt. Diese basieren auf den Definitionen aus im Abschnitt 2.1.3 und noch anderen zusätzlichen Sachverhalten, die für die Einordnung der Szenarien in dem interorganisationalen Umfeld von Belang sind. In Abschnitt 2.2 wurde die Zielsetzung dieser Arbeit zusammengefasst; die Bedeutung der Szenarien ist auf die für die Zielsetzung relevanten Eigenschaften fokussiert.

Diese Eigenschaften (Merkmale) und deren Ausprägungen bilden wie in Abildung 3.1 zusammengefasst einen morphologischen Kasten, anhand dessen eine morphologische Analyse bzw. Einordnung der einzelnen Szenarien stattfinden kann. Anhand der hier definierten Merkmale werden die Szenarien, die in den Abschnitten 3.1.2 und 3.1.3 folgen, verglichen und voneinander abgegrenzt. Das Ziel der Nutzung dieses morphologischen Kastens ist die Demonstration der Vollständigkeit der zwei Szenarien bezüglich des Fehlermanagements der interorganisationalen Umgebungen. Dies ist notwendig, da mit dieser Arbeit

Merkmal	Ausprägung		
Dienstbeziehung	vertikal	horizontal	
Steuerung	zentralistisch	kollektiv	polyzentrisch
Aufgabenzuordnung	arbeitsteilig	homogen	
Kommunikation	Baum	Stern	Peer-to-Peer
Dynamik	statisch	dynamisch	
Rollenvergabe	statisch	dynamisch	
Werkzeugorientierung	keine (manuell)	teilweise	
Verantwortliche SP	1	n	

Tabelle 3.1.: Morphologischer Kasten zur Einordnung der Szenarien

nicht nur der Entwurf einer allgemeingültigen Managementarchitektur für io FM verfolgt wird, sondern auch ein Vergleich bezüglich der unterschiedlichen io-Formen der Diensterbringung.

Die ersten vier Merkmale beziehen sich direkt auf die identifizierten Eigenschaften aus Abschnitt 2.1.3. Zusätzlich werden noch folgende Merkmale eingeführt:

Dynamik Oft bilden die Service Provider aus organisatorischen Gesichtspunkten ein statisches Provider-Netzwerk, das sich durch relativ große Stabilität und Langlebigkeit auszeichnet. Ebenfalls ist es aber möglich, dass dieses Providernetzwerk dynamisch ist, sich also häufige organisatorische Änderungen innerhalb dessen ergeben können.

Rollenvergabe Wenn mehrere Service Provider an der Erbringung eines Dienstes beteiligt sind, müssen entsprechende Rollen und Zuständigkeiten vergeben werden. Manche Rollen sind statisch vergeben d.h. diese werden einmal festgelegt und für jede Dienstinstanz gleich vergeben. Die Rollen können aber auch dynamisch zugewiesen werden zur Laufzeit eines Dienstes, bedingt durch den beteiligten Service Provider, die organisatorischen Struktur usw.

Werkzeugorientierung Die Prozesse innerhalb einer Organisation oder auch interorganisational können werkzeugunterstützt sein oder nicht. In Abwesenheit von Werkzeugen ist eine manuelle Unterstützung notwendig.

Verantwortliche Service Provider repräsentiert die Anzahl an Service Providern, die für die Erbringung eines Dienstes verantwortlich gegenüber den Kunden sind. Die Ausprägungen sind in diesem Fall 1 und n.

Hiermit stellen die acht Merkmale und deren Ausprägungen einen morphologischen Kasten dar, der als vollständig bezüglich der interorganisationalen Form der Diensterbringung in den Beispielszenarien betrachtet wird.

Zusätzlich zu diesen Merkmalen werden die im Abschnitt 2.1.2.4 beschriebenen Ansätzen bzgl. des Lebenszyklus eines Fehlers folgende Phasen betrachtet: Fehler-Entdecken, Fehler-Eingrenzen, Fehler-Beheben und Fehler-Vorhersagen/Vorbeugen. Damit werden die Anforderungen in Abschnitt 3.2.2 eingeordnet.

3.1.2. Hochschulkooperation im Münchner Wissenschaftsnetz (MWN)

An den Hochschulen, Universitäten und Forschungseinrichtungen im Großraum München studieren und forschen über 100.000 Studenten und Wissenschaftler. Um die IT-Infrastruktur und die Dienste für dieses Areal zu gewährleisten betreibt das Leibniz-Rechenzentrum (LRZ) in Garching bei München das Münchner Wissenschaftsnetz (MWN).

Das MWN verbindet die Standorte der Ludwig-Maximilians-Universität München (LMU), der Technischen Universität München (TUM), der Bayerischen Akademie der Wissenschaften (BAdW), der Hochschule München (HM) und der Hochschule Weihenstephan-Triesdorf (HSWT) miteinander. Diese Standorte sind über die gesamte Münchner Region (i.W. Münchner Stadtgebiet, Garching und Weihenstephan) verteilt, dazu kommen einige weiter entfernte Standorte in Bayern, z.B. in Straubing, Triesdorf, Iffeldorf oder auf der Zugspitze [MWN].

3.1.2.1. Allgemeine Beschreibung des IntegraTUM-Projekts

Im MWN werden viele Projekte gemeinsam von den obengenannten Institutionen durchgeführt. Ein Beispiel dafür ist das IntegraTUM-Projekt [IntegraTUM], das an der TUM angesiedelt war (Laufzeit 2004-2009). Ziel des Projekts war die Schaffung einer benutzerfreundlichen und nahtlosen Infrastruktur für Information und Kommunikation (IuK) an der TUM, die eine Verbesserung der Leistungen in Forschung und Lehre bei gleichzeitiger Kostenoptimierung ermöglicht. *Motto der Neustrukturierung der IuK der TUM ist die Rezentralisierung des Betriebs durch Nutzung modernster Techniken bei Aufrechterhaltung der dezentralen Verantwortlichkeit für Inhalte und Abläufe in Fakultäten und zentralen Einrichtungen. Redundanzen in Technik, Daten und Verantwortlichkeiten werden vermindert, die Qualität der Versorgung wird verbessert. Neue Dienstleistungen schärfen das Profil der TUM und verbessern die Grundlagen für Forschung und Lehre.* [BoBo 10]

Das IntegraTUM-Projekt setzt sich aus neun Teilprojekten zusammen (vgl. Abbildung 3.1). Auf der einen Seite wird die Einrichtung zentraler Fileserver, die Rezentralisierung der eMail Services und einer Neuorganisation der Systemadministration angestrebt. Auf der

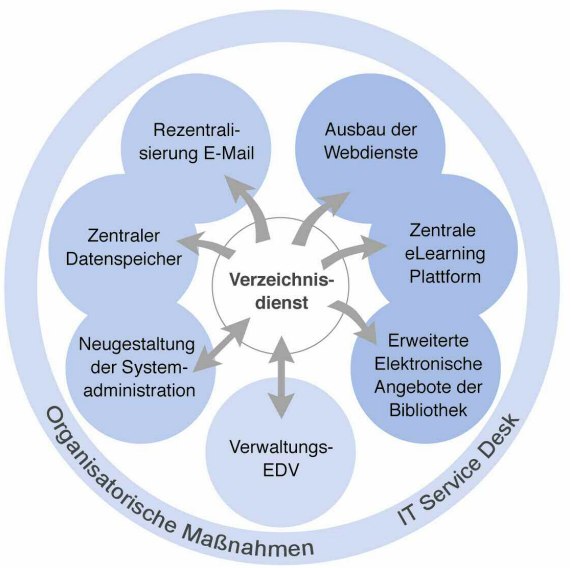


Abbildung 3.1.: Teilprojekte in Rahmen des Projekt IntegraTUM

anderen Seite werden aus einer kundenorientierten Sicht der Ausbau des TUM-Portals, die Einführung einer zentralen eLearning-Plattform und die Ausweitung elektronischer Angebote der Bibliothek realisiert. Alle Teilprojekte greifen auf einen gemeinsamen Verzeichnisdienst zu, dessen Integration mit der Verwaltungs-EDV gewünscht ist. Dieser zentrale Verzeichnisdienst, der die Verbindung zwischen den Teilprojekten realisiert, wird durch das Leibniz-Rechenzentrum (LRZ) betrieben. Auch der zentrale Datenspeicher wird am LRZ betrieben [Biar 10]. Als übergreifendes Teilprojekt ist die Organisationsentwicklung im Bereich IuK an der TUM zu erwähnen.

io ITSM-Prozesse

Um den Betrieb der Dienste, die innerhalb von IntegraTUM betrieben werden, zu gewährleisten wurden einige interorganisationale ITSM-Prozesse etabliert. Diese tragen zusätzlich zu einer Steigerung der Effektivität der Diensterbringung bei. Da diese Arbeit den Schwerpunkt auf das Fehlermanagement setzt, wird im Folgenden der Incident-Management-Prozess beschrieben, so wie er im IntegraTUM-Projekt etabliert ist [HKMY 10]. Das Hauptziel des Incident-Managements besteht darin, nach dem Eintreten einer Störung den normalen Servicebetrieb schnellstmöglich wiederherzustellen [OGC 07d]. Ebenfalls in der Verantwortung dieses Prozesses liegt die Minimierung der Auswirkungen von Störungen auf den Geschäftsbetrieb, um Dienstgüteparameter (Quality of Service), wie etwa die Verfügbarkeit, auf dem mit den Kunden vereinbarten Niveau zu halten.

3.1.2.2. Fehlermanagement für IntegraTUM

io-Fehler-management

Abbildung 3.2 stellt das Prozessmodell des IntegraTUM Incident-Management-Prozesses

dar. Dieser Prozess erstreckt sich über zwei Organisationen – TUM und LRZ –, die die Dienste in IntegraTUM gemeinsam erbringen.

In [HoKn 10] werden die Grundsätze des interorganisationalen Störungsmanagements in IntegraTUM beschrieben. Zuerst werden die beteiligten Rollen definiert: der TUM Service Desk, der für die Aufnahme von Störungen aus der TUM zuständig ist, die LRZ Hotline, die als Service Desk für das LRZ fungiert und die die Anfragen von der TUM-Service-Desk entgegennimmt und sie löst bzw. zu dem entsprechenden qualifizierten Personal weiterleitet. Die LRZ-Spezialisten sind der 2/3 Level Support am LRZ, also sind sie für die Lösung von Störungen, die über die Kompetenz der LRZ-Hotline hinaus gehen, zuständig. Der Prozessablauf (siehe Abbildung 3.2) wurde in [HKMY 10] beschrieben.

Prozessbeschreibung

Eine von einem Benutzer gemeldete Störung wird von einem Mitarbeiter des TUM Service-Desks zunächst entgegengenommen, erfasst und dokumentiert (Aktivität A_1). Dazu wird das Werkzeug Open Ticket Request System (OTRS) verwendet, mit dem ein Ticket geöffnet wird. Anschließend erfolgt eine Klassifizierung (Priorisierung und Kategorisierung) durch den Service-Desk-Mitarbeiter (Aktivität A_2). Hierbei können Dienstleistungsvereinbarungen (Service Level Agreements, SLAs) die notwendigen Inputs liefern. Der Service-Desk bzw. die Spezialisten in der TUM bearbeiten die Störung (Aktivität A_{3a}), schließen das Ticket (A_{10}) und benachrichtigen den Benutzer (Aktivität A_{11}). Zu bemerken ist, dass die TUM durch den Service Desk einen Single-Point-of-Contact (nach [OGC 07d]) für den Anwender (z.B. Studenten) bereitstellt. Dieser tritt immer als Ansprechpartner für die Dienste auf, die erbracht werden, und im Falle einer Störung trägt er die volle Verantwortung für die Weiterleitung an den Spezialisten, Überwachung des Lösungsfortschritts und Verständigung der Kunden.

Nicht in allen Fällen kann die Ermittlung der Störungsursache und die Störungsbehebung direkt durch die Mitarbeiter der TUM durchgeführt werden. Tickets, die der TUM Service-Desk nicht lösen kann und die außerhalb der Zuständigkeit der TUM-Experten liegen, werden fachlich eskaliert und an das LRZ weitergeleitet (Aktivität A_{3b}). Der Prozess wird dann am LRZ weitergeführt. Der TUM Service-Desk fungiert dabei als Ticketowner, ist also für den Gesamtprozess der Ticketauflösung verantwortlich. Es ist daher wichtig, dass die Antworten auf Störungen, die an das LRZ weitergeleitet werden, wieder zurück an den TUM Service-Desk zurückgegeben werden.

Historisch gesehen sind die LRZ- bzw. TUM-Prozesse älter als das interorganisationale IntegraTUM-Werkzeug. Einige Erweiterungen der bestehenden Prozesse mussten vorgenommen werden, um diese an die Gegebenheiten von IntegraTUM anzupassen: Damit z.B. die LRZ-Hotline nicht jede vom TUM Service-Desk weitergeleitete Anfrage manuell kopieren muss, wurde für die Ticket-Informationen ein dediziertes, formales und einfach maschinenlesbares Austauschformat vereinbart; dieses erlaubt die automatische Übernahme von TUM-Tickets als LRZ-Ticket.

Störung ans LRZ weitergeleitet

Störungen, welche von der TUM an das LRZ weitergeleitet werden, werden am LRZ neu

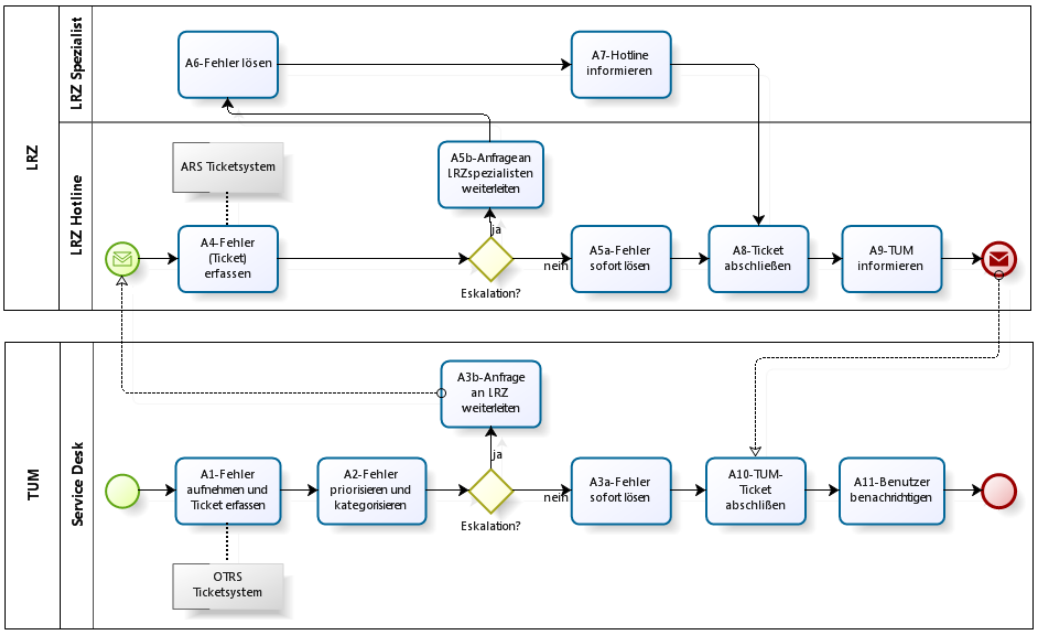


Abbildung 3.2.: IntegraTUM Incident-Management-Prozess [HKMY 10]

erfasst und stellen für das LRZ eine „neue“ Störung dar. Die Störung wird im LRZ-Ticketsystem dokumentiert; ein LRZ-Ticket wird geöffnet (Aktivität A_4). Am LRZ wird Remedy Action Request System (ARS) seit vielen Jahren als Ticket-System benutzt. Mittelfristig wird das ARS am LRZ gegen die ITSM-Suite von iET Solutions ausgetauscht. Das wird im Rahmen der Ausrichtung der ITSM-Prozesse gemäß ISO/IEC 20000 stattfinden.

Einhaltung des Prozesses

Der Prozess folgt am LRZ einem ähnlichen Ablauf wie der an der TUM. Zunächst versucht die LRZ-Hotline selbst, das Ticket zu lösen (Aktivität A_{5a}). Ist dies erfolgreich, schließt die LRZ-Hotline das LRZ-Ticket (Aktivität A_8) und leitet die Antwort an den TUM Service-Desk zurück (Aktivität A_9). Wenn die LRZ-Hotline keine Lösung für die bestehende Störung findet, eskaliert sie diese funktional innerhalb des LRZ an den zuständigen Spezialisten (Aktivität A_{5b}). Nach der Behebung der Störung durch die LRZ-Spezialisten (Aktivität A_6) erfolgt eine Meldung an die LRZ-Hotline (Aktivität A_7), das LRZ-Ticket wird geschlossen (Aktivität A_8) und die Lösung an das TUM Service-Desk gemeldet (Aktivität A_9). Hier werden die Aktivitäten A10 und A11 wie oben beschrieben durchgeführt. Der Prozess muss in jedem Fall eingehalten werden, auch in dem Fall, wenn am LRZ Störungen gemeldet werden, die offensichtlich die TUM betreffen. Hierfür verweist die LRZ-Hotline die Benutzer an den Service-Desk der TUM.

Merkmal	Ausprägung		
Dienstbeziehung	vertikal	horizontal	
Steuerung	zentralistisch	kollektiv	polyzentrisch
Aufgabenzuordnung	arbeitsteilig	homogen	
Kommunikation	Baum	Stern	Peer-to-Peer
Dynamik	statisch	dynamisch	
Rollenvergabe	statisch	dynamisch	
Werkzeugorientierung	keine (manuell)	teilweise	
Verantwortliche SP	1	n	

Tabelle 3.2.: Einordnung des MWN-Szenario im morphologischen Kasten

3.1.2.3. Einordnung des MWN-Szenarios

Bezug nehmend auf den morphologischen Kasten (siehe Abschnitt 3.1.1) wird in Abbildung 3.2 das MWN-Szenario folgendermaßen eingeordnet:

Dieses Szenario ist ein gutes Beispiel für eine hierarchische Form der Diensterbringung (siehe die Abschnitte 2.1.3.3 und 3.1.1): vertikale Dienstbeziehung (die untergeordnete Dienste, die das IntegraTUM-Projekt benötigt, werden vom LRZ bereitgestellt), baumartige Struktur und zentralisierte Steuerung (TUM ist dabei die „Wurzel"-Organisation), sowie eine arbeitsteilige Aufgabenzuordnung.

Die Service Provider sind statisch organisiert und auch die Rollen in einem solchen Providernetzwerk sind statisch vergeben. Das MWN-Szenario wird teilweise werkzeugunterstützt. Die Anzahl zuständigen Service Provider in diesem Fall ist 1.

Die unterschiedlichen Aktivitäten des Fehlermanagementprozesses können auf die Phasen des Lebenszyklus eines Fehlers abgebildet werden: Aktivität A_1 repräsentiert das Entdecken von Fehlern, die Aktivitäten A_2, A_{3b}, A_4, A_{5b} sind Teile des Fehler-Eingrenzen-Phase und die anderen des Fehler-Behebens. Teilweise wird das OTRS System auch für die Realisierung von Fehler-Vorhersagen/Vorbeugung benutzt.

3.1.3. Géant E2E-Links für das LHC Compute Grid (LCG)

Das 1997 gegründete europäische Wissenschaftsnetz Géant [Geant11] verbindet mittlerweile 34 nationale Forschungsnetz-Organisationen – engl. National Research and Educational

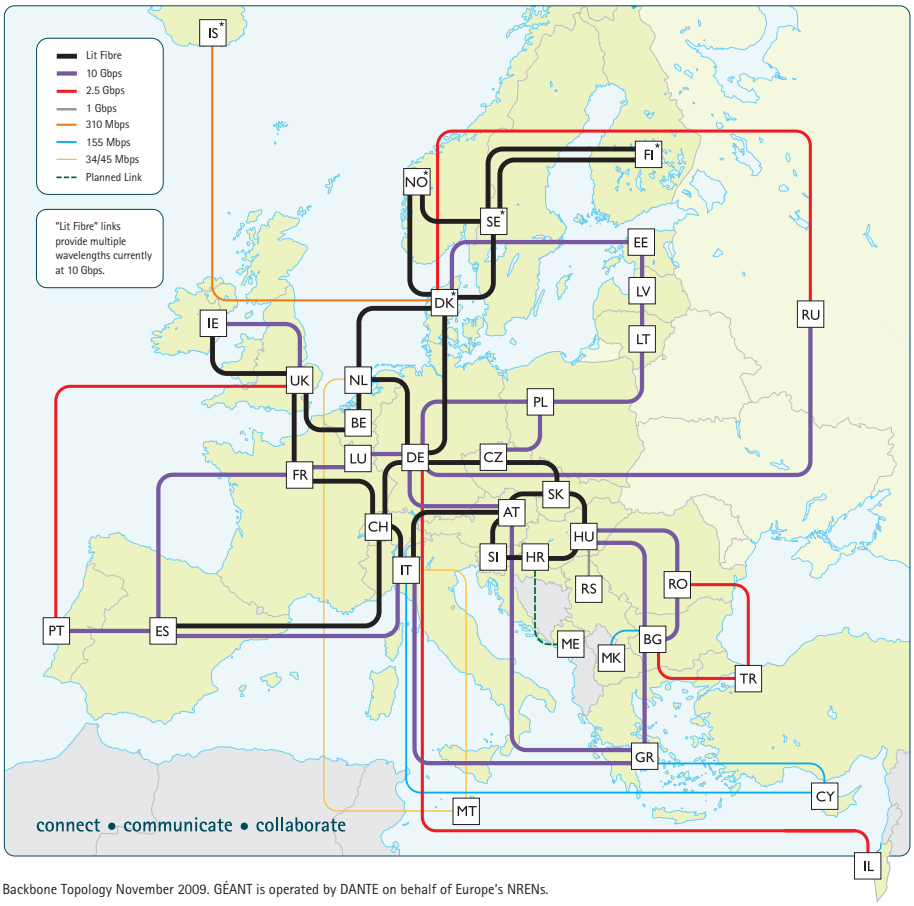


Abbildung 3.3.: Das Géant Backbone-Netz [Geant11]

Network (NREN) – und ist jetzt bereits in der siebten Generation. Dieses transeuropäische Netz bietet wissenschaftlichen und universitären Einrichtungen sowie internationalen Forschungsprojekten Netzdienste an. Das Géant-Netz basiert auf Dark-Fiber-Glasfaserstrecken; der Betrieb erfolgt eigenständig durch den Géant-Verbund: Die jeweiligen NRENs versorgen die wissenschaftlichen Einrichtungen in ihrem Zuständigkeitsgebiet, die Organisation Delivering Advanced Networking Technology to Europe (DANTE) betreibt das Géant Backbone Netz (siehe Abbildung 3.3), über das die innereuropäische Vernetzung sowie der Anschluss an außereuropäische Wissenschaftsnetze erfolgt.

3.1.3.1. Allgemeinbeschreibung

Die Dienste, die in Géant angeboten werden, sind spezifisch für sogenannte hybride Netze (es koexistieren klassische geroutete IP-Dienste und höherwertige Point-to-Point-Verbindungen).

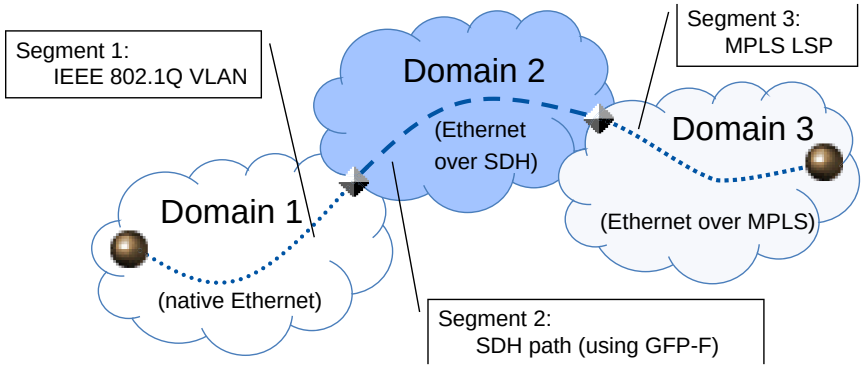


Abbildung 3.4.: Beispiel eines Géant E2E-Links [Hamm 09]

Diese Dienste sind im Gegensatz zum IP-Dienst verbindungsorientierte, dedizierte optische Links und müssen von Kunden explizit bestellt werden. In Géant werden zur Zeit zwei solche Dienste angeboten [Hamm 09]:

Géant Plus Das Géant Plus Dienst stellt Datenstrecken (zwischen 155Mb/s und 10Gb/s) innerhalb der optischen Géant-Infrastruktur bereit; ggf. auch darüber hinaus (z.B. für Transatlantikverbindungen). Géant Plus Links sind an einer Reihe von Aufpunkten bereits vorkonfiguriert und können kurzfristig bestellt werden.

E2E-Links End-to-End (E2E) Links sind Datenstrecken mit Bandbreiten bis zu 10 Gbps, die speziell auf Kundenanforderung eingerichtet werden. Für diesen Dienst gibt es keine Beschränkung der möglichen Zugangspunkte. Auch für diesen Dienst sind Verbindungen über das Géant2-Netz hinaus möglich.

heterarchische Form der Diensterbringung E2E-Links, wie in Abbildung 3.4 dargestellt, sind dedizierte optische Multi-Gigabit Verbindungen zwischen zwei Einrichtungen, die sich über mehrere Domänen erstrecken. Sie sind technisch realisiert als private Ethernet-Verbindungen, die eine Bandbreite bis zu 10 Gbps bereitstellen. Jede Domäne kann unterschiedliche Technologien einsetzen (z.B. Ethernet VLAN, SDH mit Generic Framing Protocol Encapsulation (GFP-F) oder MPLS mit Label Switched Path (LSP)). E2E-Links sind horizontale Dienstketten (wie in Abschnitt 2.1.3.1 definiert) basierend auf Teildiensten mehrerer Domänen. Die Tatsache, dass die Teilstrecken durch „Verkettung" zu einem E2E-Link führen, deutet auf eine heterarchische Form der interorganisationale Diensterbringung hin.

LHC Eins der Nutzungsszenarien von Géant E2E-Links ist das Large Hadron Collider (LHC) Projekt. Der Teilchenbeschleuniger LHC des Europäischen Labors für Teilchenphysik (früher: *Conseil Européen pour la Recherche Nucléaire*) CERN, der Anfang 2010 voll in Betrieb gegangen ist, produziert jährlich durch seine Experimente circa 8 Petabyte Messdaten und 4 Petabyte Simulationsdaten, die für Analyse und Forschung über 5000 Wissenschaftlern weltweit zur Verfügung stehen [Grid 09]. Für die Datenspeicherung, die Verteilung und die Analyse der

Daten wurde 2006 bis 2008 das LHC Compute Grid (LCG) aufgebaut. Nach Schauerhammer ist das LCG die aktuell größte gemeinschaftliche Anstrengung der Teilchenphysik im IT-Bereich [Scha 05].

Die unterliegende Netzarchitektur ist auf vier Ebenen verteilt (sogenannte Tiers). Tier 0 ist CERN. Hier werden die Rohdaten gesammelt und ein vollständiges Backup realisiert. Auf die 12 Tier-1-Zentren (Rechenzentren weltweit mit großen Rechen- und Speicherkapazitäten) werden die Daten anschließend verteilt. Die Aufgaben dieser Zentren sind: die verteilte Archivierung der Messdaten, die Durchführung daten-intensiver Analysevorgänge, das Management und die Speicherung von Daten sowie der Betrieb von Grid-Diensten. Mehrere Tier-2-Zentren sind mit einem oder mehreren Tier-1-Zentren verbunden. An Tier-2-Zentren werden Tier-3 (z.B. Forschungszentren) bzw. Tier-4 (z. B. Arbeitsplätze der Wissenschaftler) angeschlossen. Für das Szenario wird nur das Netz zwischen Tier-0 und Tier-1-Zentren betrachtet.

LHCOPN

Um die hohen Anforderungen an die Netzinfrastruktur zwischen Tier-0 und Tier-1-Zentren (Verfügbarkeit von 98%, enorme Datenströme bis zu 10Gbps sind zu erwarten) zu erfüllen, wurde auf Basis von E2E-Links ein Optisches Privates Netz (OPN) eingerichtet. Die Topologie des LHCOPN ist sternförmig mit Tier-0 in der Mitte. Jedes der Tier-1-Zentren ist an diesem mit einem E2E-Link angebunden [BMM 05, LHC 09].

E2ECU

Um eine hohe Verfügbarkeit der E2E-Links zu gewährleisten wurde eine E2E Coordination Unit (E2ECU) eingerichtet. Die Hauptaufgabe der E2ECU ist der Informationsaustausch zwischen den NRENs aber auch die Koordination des Fehlermanagements für E2E-Links zwischen den Kunden und den NREN sowie die Einrichtung von E2E-Links. Jedes NREN hat eine Betriebsgruppe für Schicht-2-Datenstrecken eingerichtet, Transmission Network Operation Centre (TNOC), die für das Management der Segmente von E2E-Links innerhalb des Organisationsbereich des jeweiligen NRENs verantwortlich ist. Diese Betriebsgruppe kommuniziert mit der E2ECU. Die E2ECU kommuniziert nicht direkt mit Benutzern von E2E-Links, sondern mit den jeweiligen Betriebsgruppen der Projekte.

3.1.3.2. Fehlermanagement für E2E-Links

Eine der Anforderungen des LHCOPN ist, dass Störungen und Ausfälle von E2E-Links nicht länger als zwei Tage dauern. Der Fault-Management-Prozess soll die Erkennung und die zügige Bearbeitung von Störungen ermöglichen. Der Prozess wird von der E2ECU koordiniert. Der Prozess definiert das planvolle und strukturierte Vorgehen im Störungsfall. Der globale Fault-Management-Prozess für E2E-Links ist in Abbildung 3.5 dargestellt.

Rollen

Der Prozess umfasst folgende Rollen: Die E2ECU besitzt die zentrale Rolle im Fehlermanagement-Prozess. Als koordinierende Rolle in der Störungsbehebung sind ihre Aufgaben ausschließlich administrativer Natur. Die Kommunikation bzw. der Informationsaustausch zwischen den Prozessteilnehmern verläuft ausschließlich über die E2ECU. Störungen werden

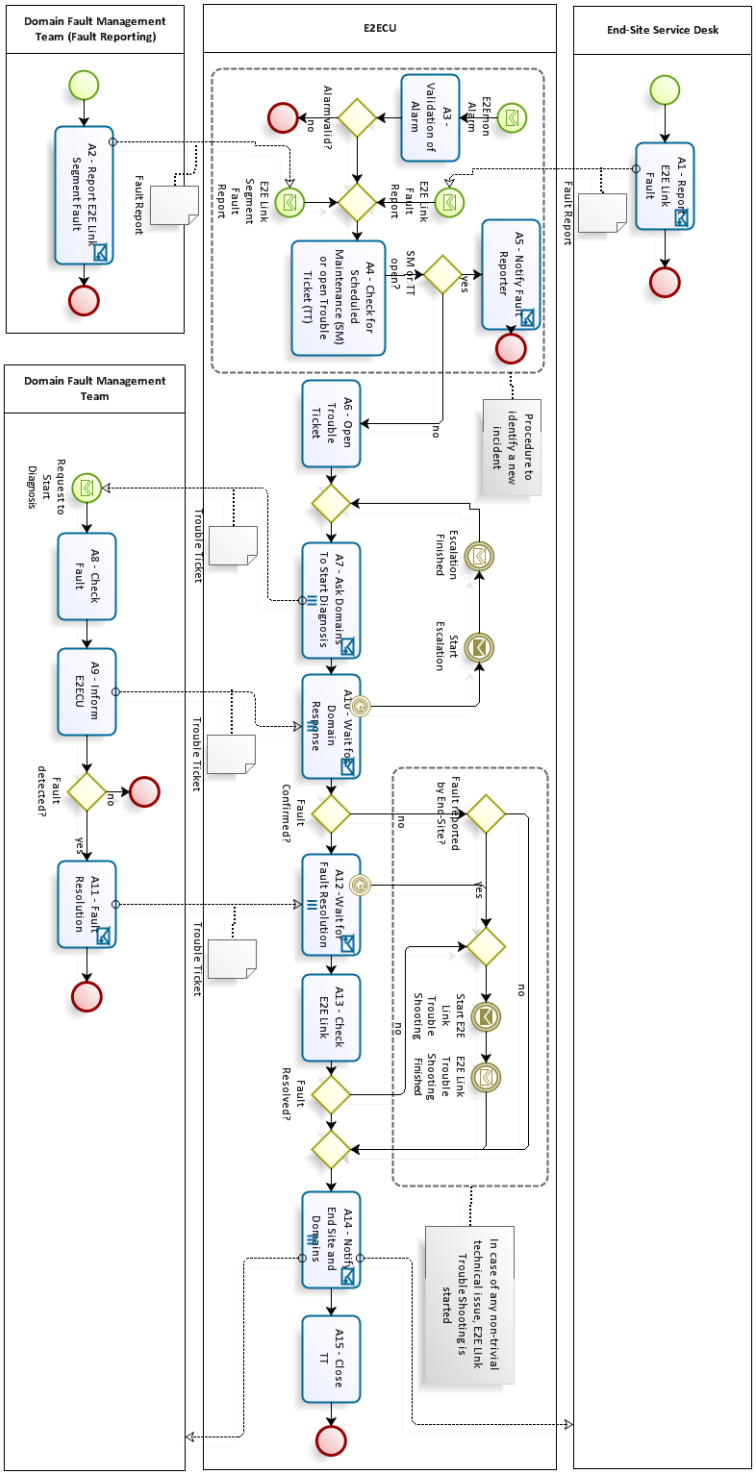


Abbildung 3.5.: Incident-Management-Prozess für den E2E-Link-Dienst [Hamm 09]

mit Hilfe des E2E-Link-Monitoring-Systems von der E2ECU erkannt, werden aber auch von den Nutzern via End-Site-Service-Desk entgegengenommen (Service Desk auf Kundenseite). Die Domänen können der E2ECU Störungen melden durch die Betriebsgruppen, die an der Erbringung der betroffenen E2E-Link Instanz beteiligt sind Domain-Fault-Management-Team. Diese Betriebsgruppen haben neben der Aufgabe der Fault-Reporting auch die Verantwortung für die Störungsbehebung auf Domänen-Seite.

Prozessbeschreibung

Der Fehlermanagement-Prozess startet mit der Erkennung bzw. Meldung einer Störung (A_1 ... A_5). Informationen aus jeder der beteiligten Domänen werden in regelmäßigen Zeitabständen von E2E Monitoring System (E2EMon) abgefragt und daraus ein E2E-Wert ermittelt. Wenn es sich um Störungen handelt, die durch das E2EMon gemeldet werden, wird erst die entsprechende Meldung überprüft, um Fehlalarme bzw. sehr kurze Ausfälle von Links auszuschließen. Ebenfalls wird überprüft, ob diese Störung auf geplante Wartungsmaßnahmen zurückzuführen ist oder die Störung schon gemeldet worden ist. Wenn die entsprechenden Voraussetzungen erfüllt sind, kann ein Trouble Ticket (Aktivität A_6) eröffnet werden.

Die E2ECU fordert die beteiligten Domänen dazu auf (A_7), die Störung zu bestätigen bzw., wenn möglich, deren Ursache zu ermitteln (A_8, A_9). Die beteiligten Domänen überprüfen die Funktionalität der Teildienste (und darunterliegende Ressourcen), die sie erbringen. Wenn es sich um eine Störung handelt, dann startet sofort die Fehlerbehebung (A_{11}), ohne dass die E2ECU beteiligt ist. Bleiben Rückmeldungen aus, wiederholt die E2ECU die Anfrage. Wird die Störung nicht bestätigt, wird unmittelbar der Hilfsprozess E2E Link Trouble Shooting gestartet. Ansonsten wartet die E2ECU, bis die Rückmeldungen der Domänen vorliegen, dass die Störung bestätigt ist (A_{12}), und überprüft deren erfolgreiche Behebung (A_{13}). Wenn die Störung nicht behoben wurde, wird der Hilfsprozess E2E Link Trouble Shooting begonnen. Abschließend werden die End Site bzw. die beteiligten Domänen über die Behebung der Störung informiert (A_{14}) und das Trouble Ticket wird geschlossen (A_{15}).

3.1.3.3. Einordnung des LCG-Szenarios

Bei der Einordnung des LCG-Szenarios wird Bezug auf den morphologischen Kasten (siehe Abschnitt 3.1.1) genommen. Abbildung 3.3 stellt die zusammengefasste Morphologie des LCG-Szenarios dar:

Die Dienstbeziehungen sind in dem vorliegenden Fall horizontal und das Koordinationsmuster ein heterarchisches.

Die meisten Betriebsprozesse sind geprägt von einer homogenen Aufgabenzuordnung, d.h. mehrere Provider müssen dieselben Teilvorgänge innerhalb ihrer Domänen durchführen. Die Aufgabenzuordnung ist homogen, da die Aufgaben aller NRENs prinzipiell gleich sind, sieht man von den providerübergreifenden Tätigkeiten etwa der E2ECU ab.

Merkmal	**Ausprägung**		
Dienstbeziehung	vertikal	horizontal	
Steuerung	zentralistisch	kollektiv	polyzentrisch
Aufgabenzuordnung	arbeitsteilig	homogen	
Kommunikation	Baum	Stern	Peer-to-Peer
Dynamik	statisch	dynamisch	
Rollenvergabe	statisch	dynamisch	
Werkzeugorientierung	keine (manuell)	teilweise	
Zuständige SP	1	n	

Tabelle 3.3.: Einordnung des LCG-Szenario im morphologischen Kasten

Bezüglich der Steuerung der Betriebsprozesse gibt es im Szenario keine verbindlichen Vorgaben. Auch wenn die E2ECU eine gewisse zentrale und koordinierende Rolle einnimmt, kann die Implementierung von Betriebsprozessen auf der Basis einer zentralistischen Steuerung durch eine einzelne Organisation über die Eigenständigkeit der NRENs hinweg nicht durchgesetzt werden. Daher wird für die Implementierung der Betriebsprozesse eine kollektive oder polyzentrische Steuerung notwendig.

Die Kommunikation zwischen den beteiligten Domänen erfolgt mit den Mustern Stern (mit der E2ECU als Information Hub) sowie durch direkte Kommunikation zwischen den Domänen (Peer-To-Peer).

Die Service Provider sind dynamisch organisiert, und ebenfalls werden die Rollen in diesem sogenannten Providernetzwerk dynamisch vergeben (jeder Provider kann zu einem gewissen Zeitpunkt und /oder Situation unterschiedliche Rollen besitzen).

Das providerübergreifende E2EMon-System unterstützt die Fehlererkennung; darüber hinaus gibt es in diesen Szenarien keine gemeinsam benutzten Werkzeuge. Daher ist die Werkzeugunterstützung als sehr gering einzustufen (manuell). Die Anzahl zuständiger Service Provider in diesem Fall ist n: Zwar gibt es für jeden E2E-Link einer Verantwortlichen gegenüber der Kunde, diese Rolle können jedoch alle Provider einnehmen.

Die Phasen des Lebenszyklus eines Fehlers werden folgendermaßen von den Aktivitäten abgedeckt: die Aktivitäten A_1 und A_2 entsprechen dem Fehler-Entdecken, A_3 bis A_7, A_{10} und A_{12} repräsentieren das Fehler-Eingrenzen und die restlichen Aktivitäten das Beheben. Eine Fehler-Vorhersage/Vorbeugung ist noch nicht Teil des Fehlermanagementprozesses, allerdings wird dies angestrebt.

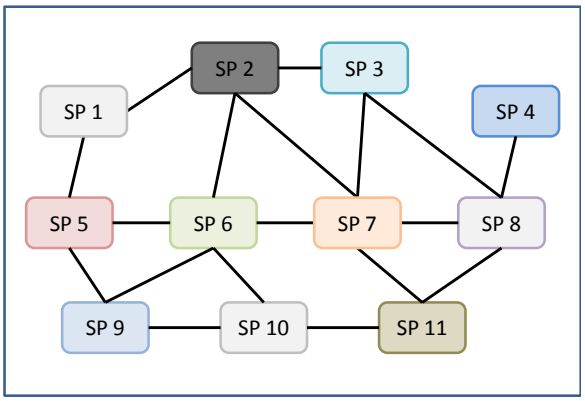


Abbildung 3.6.: Ein allgemeines Provider-Netzwerk

3.1.4. Verallgemeinertes Szenario

In den vorherigen zwei Abschnitten wurden reale Szenarien für die Diensterbringung in interorganisationalen Umgebungen beschrieben. Obwohl diese sehr wertvoll für das interorganisationale Fehlermanagement sind, verfügen sie trotzdem nicht über Allgemeingültigkeit. Dies wurde in den Abschnitten 3.1.2.3 und 3.1.3.3 bei der Einordnung der Szenarien in den morphologischen Kasten beschrieben. Eine Überlappung der beiden morphologischen Kästen (Abbildungen 3.2 und 3.3) hingegen führt zu einem ,,voll abgedeckten" Bereich . Das heißt, dass eine Kombination der zwei Szenarien bzw. der Eigenschaften der hierarchischen und heterarchischen Formen der Diensterbringung ein allgemeingültiges Szenario für diese Arbeit darstellen.

3.1.4.1. Allgemeine Beschreibung

Ausgehend von den zwei existierenden Szenarien IntegraTUM (in Abschnitt 3.1.2) und LHCOPN (Abschnitt 3.1.3) als Beispiele für eine hierarchische bzw. heterarchische Form der Diensterbringung wurde das verallgemeinerte Szenario zusammengefasst. Die Service Provider (SP1-SP11) sind in diesem Fall in einem Provider-Netzwerk organisiert wie in Abbildung 3.6 dargestellt. Die Provider im Provider-Netzwerk sind z. T. miteinander gekoppelt, einen Vollvermaschung zwischen diesen liegt aber nicht zwingend vor.

Wie auch in den realen Szenarien nehmen unterschiedliche Kunden (s. Abbildung 3.7) unterschiedliche Dienste, die von den Providern einzeln bzw. kooperativ erbracht werden, in Anspruch. Die Dienstbeziehungen zwischen Kunde und Service Providern bzw. zwischen zwei Service Providern werden in den Abbildungen durch gepunktete Linien gekennzeichnet. Je nachdem, welcher Kunde welchen Dienst benutzen will, werden die Dienstbeziehungen bzw. Kooperationsbeziehungen dynamisch innerhalb des Provider-Netzwerks instantiiert. *Diensterbringung:*

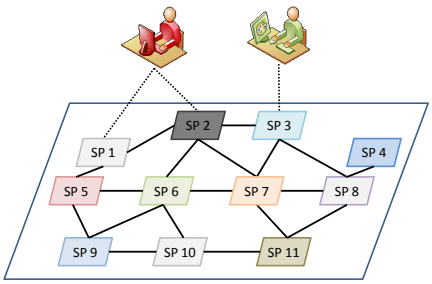


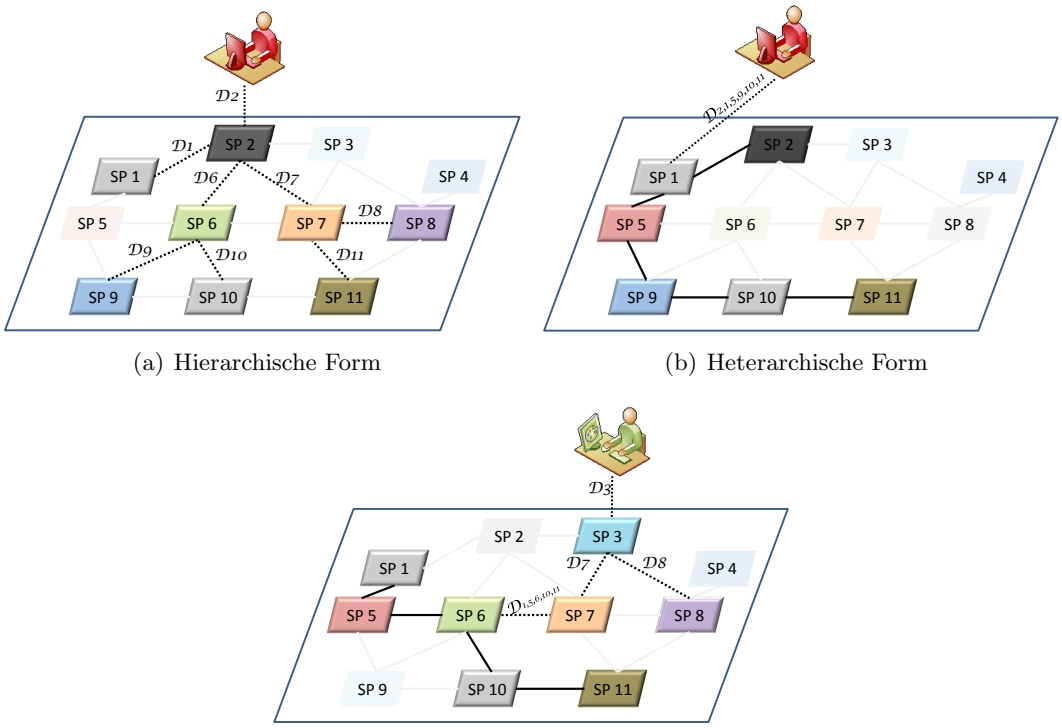
Abbildung 3.7.: Nutzung der Dienste in dem verallgemeinerten Szenario

Wie in Abbildung 3.8 dargestellt, kann es zu folgenden drei allgemeinen Situationen kommen:

...hierarchisch Ein Kunde benutzt einen Dienst $\mathcal{D}_2$, der vom Service Provider SP2 angeboten wird (Abbildung 3.8(a)). Für den Kunden ist dieser Dienst nicht transparent, d.h. der Kunde hat keinen Einblick in die Diensterbringung durch SP2. SP2 ist für den Kunden ein Single-Point-of-Contact bezüglich des Dienstes $\mathcal{D}_2$. SP2 benutzt wiederum die Dienste $\mathcal{D}_1$, $\mathcal{D}_6$, $\mathcal{D}_7$, die von den Provider SP1, SP6, SP7 erbracht werden. SP2 nimmt die Rolle des Nutzers für die drei obengenannten Service Provider ein. Dienst $\mathcal{D}_6$ benötigt zu seiner Realisierung die beiden Dienste $\mathcal{D}_9$ und $\mathcal{D}_{10}$, die von SP9 und SP10 bereitgestellt werden. Ebenfalls werden bei der Realisierung von $\mathcal{D}_7$ die Dienste $\mathcal{D}_8$ und $\mathcal{D}_{11}$ der Service Provider SP8 und SP11 benötigt. Also es ergibt sich hier eine logische Hierarchie innerhalb des Provider-Netzwerks, die für die Erbringung des Dienstes $\mathcal{D}_2$ „verantwortlich" ist.

...heterarchisch Ein anderer Kunde benötigt beispielsweise einen E2E-Dienst $\mathcal{D}_{2,1,5,9,10,11}$, der von den Service Providern SP2, SP1, SP5, SP9, SP10 und SP11 gemeinsam erbracht wird. Hier liegt ein Fall von heterarchischen Diensterbringung vor. Die genannten Service Provider sind verkettet (organisatorische Verkettung) und jeder davon beteiligt sich an dem E2E-Dienst mit seinem Teildienst. Der Kunde hat in diesem Fall keinen Single-Point-of-Contact, mit dem er kommunizieren kann, weil die Verantwortung für die Diensterbringung bei allen Service Providern gleichermaßen liegt.

...gemischt Sehr oft kommt es vor, dass es sich um eine gemischte Form der Diensterbringung handelt. Wie im Beispiel (Abbildung 3.8(c)) zu sehen ist, nutzt der Kunde einen Dienst $\mathcal{D}_3$, der von Service Provider SP3 erbracht wird. Dieser Service Provider ist für den Kunden der Single-Point-of-Contact, ist also allein verantwortlich für den Dienst $\mathcal{D}_3$, den er bereitstellt. Die Realisierung des Dienstes ist wie im Fall der Hierarchie nicht transparent für den Kunden. Bei der Realisierung des Dienstes $\mathcal{D}_3$ werden die Dienste $\mathcal{D}_8$ und $\mathcal{D}_{1,5,6,10,11}$ benötigt. Der Dienst $\mathcal{D}_8$ ist ein grundlegender Dienst, der in dieser Konstellation keinen anderen



(a) Hierarchische Form

(b) Heterarchische Form

(c) Gemischte Form

Abbildung 3.8.: Nutzung eines Dienstes in unterschiedlichen Formen der Diensterbringung

Dienst zur Realisierung braucht. Auf der anderen Seite ist der Dienst $\mathcal{D}_{1,5,6,10,11}$ ein E2E-Dienst, der gemeinsam von den fünf Service Providern zur Verfügung gestellt wird. Bei der Erbringung des Dienstes $\mathcal{D}_3$ greift dieser auf zwei andere Dienste zu. Bis zu diesem Punkt könnte man über eine hierarchische Form der interorganisationalen Diensterbringung reden. Weiter unten in der Hierarchie wird eines der ,,Baumblätter" durch einen heterarchischen Dienst repräsentiert.

Das verallgemeinerte Szenario erfasst die meisten Provider-Netzwerke bei der interorganisationalen Diensterbringung. Sie vereinigt Eigenschaften, die sowohl für hierarchische als auch heterarchische Formen der interorganisationalen Diensterbringung gültig sind.

3.1.4.2. Zusammenfassung der io-FM relevanten Eigenschaften

Um einen allgemeingültigen Fehlermanagementprozess für das verallgemeinerte Szenario beschreiben zu können, müssen folgende Aspekte aus den zwei realen Szenarien zusammen-

Rollen in Szenarien		Allgemeinrollen		
		GFMT	DFRMT	DFRT
IntegraTUM	TUM Service-Desk	✓	✓	
	LRZ Hotline		✓	
	LRZ Spezialisten			✓
LCG	E2ECU	✓		
	End-Site Service Desk		✓	
	DFMT(fault-Reporting)		✓	
	DFMT			✓

Tabelle 3.4.: Abbildung der Rollen in den realen Szenarien auf die Rollen im verallgemeinten Szenario

gefasst in Betracht gezogen werden:

Rollen. Die am io-Fehlermanagement beteiligten Rollen müssen allgemein gültig sowohl für beide Formen der interorganisationale Diensterbringung als auch für den gemischten Fall definiert sein. Sie werden aus den bestehenden Rollen in den beschriebenen Szenarien (Abschnitte 3.1.2.2 und 3.1.3.2) verallgemeinert. In beiden Szenarien gibt es koordinierende Rollen im io-Fehlermanagement. Im IntegraTUM Szenario ist dies TUM Service Desk (hier in seiner koordinierenden Rolle) und im LCG die E2ECU, die die Verantwortung über die Durchführung und Einhaltung des Fehlermanagement-Prozesses übernimmt.

Im hierarchischen Fall wird diese Rolle statisch vergeben an den TUM Service Desk. Im heterarchischen Fall wird eine mögliche koordinierende Rolle auch bevorzugt, weil es für den Kunden einfacher ist einen einzigen Ansprechspartner zu haben, der gleichermaßen die anderen Provider auch repräsentiert. Diese Rolle wird dabei dynamisch vergeben, abhängig von den an der Diensterbringung beteiligten Providern, sowie von der Art und dem Enstehungsort eines Fehlers. Zusammenfassend ergibt sich die Notwendigkeit einer allgemeinen koordinierenden Rolle Global-Fault-Management-Team (GFMT).

Für die Aufnahme der Störungsmeldung werden in den zwei Szenarien folgende Rollen einbezogen: TUM Service Desk und LRZ Hotline (für IntegraTUM) sowie End-Site-Service-Desk und Domain-Fault-Management-Team (Fault-Reporting) (für LCG). Daraus ergibt sich die Domain-Fault-Reporting-Management-Team (DFRMT) Rolle, die den tatsächlichen Service Desk der jeweiligen Domäne repräsentiert. Die LRZ Spezialisten (in IntegraTUM) und das Domain-Fault-Management-Team (in LCG) repräsentieren die Rolle, die die eigentliche Störungsbehebung übernimmt. Dies wird zusammengefasst

		ZU				
		user	GFMT	DFRMT	DFRT	SP
VON	user		✓	✓		✓
	GFMT	✓		✓	✓	✓
	DFRMT	✓	✓	✓	✓	✓
	DFRT		✓	✓		✓
	SP	✓	✓	✓	✓	✓

Tabelle 3.5.: Kommunikationsbeziehungen zwischen den Rollen im verallgemeinten Szenario

zu der allgemeinen Rolle Domain-Fault-Resolution-Team (DFRT). Ein Überblick über die definierten Rollen wird in Tabelle 3.4 gegeben.

Kommunikation. Die Kommunikation zwischen den unterschiedlichen Service Provider Organisationen bzw. zwischen den unterschiedlichen Rollen muss eindeutig definiert werden. Bei der Erkennung bzw. der Aufnahme einer Störung weist die Heterarchie am Beispiel von LCG mehrere Kommunikationsbeziehungen zwischen den unterschiedlichen Rollen (End-Site-Service-Desk, Domain-Fault-Management-Team (Fault-Reporting)) auf. Im Fall IntegraTUM werden die Störungserkennung bzw. -annahme zentral im TUM Service Desk realisiert anhand von Kunden- bzw. Monitoringsystemmeldungen. Die Weiterleitung der Störung an andere beteiligte Service Provider ist auch unterschiedlich realisiert: Im heterarchischen Fall leitet das GFMT die Störungsmeldung an alle beteiligten Domänen, und jede der Domänen sucht, ob diese ihnen zuzuordnen ist. Im hierarchischen Fall weiß das GFMT, welcher Service Provider ihm den Dienst bereitstellt, also leitet es die Störung auch nur diesem weiter. Also muss diese Kommunikationsbeziehung eine $1 : n$ Beziehung sein, wobei im Fall der Hierarchie $n = 1$ ist. Die unterschiedlichen Kommunikationswege zwischen den vorher beschriebenen Rollen sind in Tabelle 3.5 zusammengefasst.

Funktionalitäten. In Abschnitt 2.1.1.3 wurden die Teilmodelle der IT-Managementarchitekturen beschrieben. Dabei wurden bei dem Funktionsmodell (auf Seite 20) im Funktionsbereich Fehlermanagement Hauptaufgaben und Teilaufgaben definiert (Gewährleistung einer hohen Verfügbarkeit eines Systems und seiner bereitgestellten Dienste durch schnelle Entdeckung und Behebung von Fehler, Überwachung des Netz- bzw. Systemzustandes, Aufnahme und Verarbeiten von Alarmen, Diagnose der Fehlerursachen, Herausfinden von Fehlerpropagation, Einleiten und Überprüfen von Fehlerbehebungsmaßnahmen, Führen eines Trouble-Ticket-Systems, Hilfestellungen für den Benutzer). Diese Funktionalitäten werden übernommen und, aufgrund der bereits beschriebenen Szenarien, für das interorganisationale Fehlermanagement erweitert. Somit werden Funktionalitäten festgelegt, die allgemeingültig für die beiden Szenarien sind (und hiermit für das verallgemeinerte Szenario), insbesondere für die auf intra-

Rollen in Szenarien		Funktionalitäten				
		Fehler-lokali-sierung	Status-anzeige	Über-wachung	Reporting	False-Positives
Integra-TUM	TUM Service-Desk		✓	✓		✓
	LRZ Hotline		✓	✓		✓
	LRZ Spezialisten	✓		✓	✓	✓
LCG	E2ECU		✓	✓	✓	
	End-Site Service Desk		✓	✓		✓
	DFMT(fault-Reporting)			✓	✓	✓
	DFMT	✓		✓	✓	✓

Tabelle 3.6.: Beteiligung der Rollen aus den realen Szenarien an der Realisierung der Funktionalitäten

organisationale Ebene geforderten Aufgaben.

Fehlerlokalisierung wird beispielsweise in den beiden Szenarien in unterschiedlichen Ausprägungen benötigt: Lokalisierung in der eigenen Domäne oder in einer anderen Domäne. Eine andere wichtige Funktionalität, die in dem verallgemeinerten Szenario benötigt wird, ist die Statusanzeige der unterschiedlichen Tätigkeiten (Fehlerbearbeitungen oder Wartungsarbeiten). Diese müssen jederzeit dem Service Desk vorliegen. Die ständige Überwachung der Domänen oder die Gesamtüberwachung über den Zustand des Netzes oder der angebotenen Dienste ist eine grundlegende Tätigkeit, die miteinbezogen werden muss. Die Möglichkeit eines Berichtswesens (Reporting) sowohl über die aktuellen verfügbaren Accounting- oder Metrikdaten als auch über die verfügbaren QoS-Parameter muss gegeben sein; die Realisierung von Trendanalysen geht damit einher. Nicht unbedeutend ist die Erkennung von Fehlalarmen und deren Vermeidung.

Tabelle 3.6 stellt die Rollen in den realen Szenarien und deren Einbeziehung bei der Realisierung der Funktionalitäten dar.

Als qualitatives Merkmal muss das io-Fehlermanagement gleichermaßen Total- als auch Partialausfälle (siehe Laprie's Dienstausfälle und Fehlverhalten im Abschnitt 2.1.2.1) berücksichtigen. Dabei muss insbesondere der Fall berücksichtigt werden, dass die Dienstfunktionalität endet, und auch der Fall, dass ein Dienst bereitgestellt wird, der fehlerhaft läuft, d.h. in seinem Systemverhalten von der Spezifikation abweicht.

Informationsmodellierung. Für ein effizientes io-Fehlermanagement ist eine entsprechende Datenbasis (*repository*) im Sinne einer ioFM-MIB notwendig, die wichtige Informationen, wie beteiligte Domänen bei der Erbringung eines Dienstes und deren Kurzbeschreibung, die Form der io-Diensterbringung, die aktuelle Rollenbelegung usw. enthält. Für den Informationsaustausch zwischen den unterschiedlichen Domänen müssen Service Provider bzw. die DFRMT und GFMT sich auf ein gemeinsames Informationsformat, das sie bei der ioFM benutzen, einigen. Das lokale Fehlermanagement kann ggf. sein proprietäres Format auf ein gemeinsam vereinbartes Format abbilden.

ioKoordination. Jeder der an der Diensterbringung beteiligten Service Provider besitzt seine eigenen lokalen Fehlermanagementsysteme. Unter diesen Umständen steigt die Notwendigkeit einer geeigneten Koordination zwischen lokalen Fehlermanagementsystemen bzw. zwischen diesem und dem Globalen Fehlermanagementsystem.

3.1.4.3. Fehlermanagement für das verallgemeinerte Szenario

Störung aufnehmen...

Abbildung 3.9 stellt den Fehlermanagement-Prozess im Fall des verallgemeinerten Szenarios dar. Der hier beschriebene Fehlermanagement-Prozess ist übergreifend zu den beiden spezifischen Fehlermanagement-Prozessen, die in den Abschnitten 3.1.2.2 und 3.1.3.2 beschrieben wurden. Umgekehrt sind die beiden Prozesse, die in den Szenarien beschrieben worden sind, Ausprägungen dieser allgemeinen Fehlermanagement-Prozesse. Eine Störung wird vom Domain-Fault-Reporting-Management-Team (DFRMT) gemeldet (Aktivität A_1). Dieses bekommt die Störungsmeldung von einem Nutzer oder von Netzüberwachungssystemen. Die Störung wird ans Global-Fault-Management-Team (GFMT) weitergeleitet. Der GFMT überprüft die Meldung über Zusammenhänge mit Wartungsarbeiten oder schon bekannten Störungsmeldungen (A_2). Wenn es sich um einen solchen Fall handelt, wird der DFRMT darüber informiert (A_3). Ansonsten wird die Störung aufgenommen (A_5), klassifiziert (A_6) und anschließend allen weiteren Domänen zur Störungsüberprüfung geschickt (A_7).

an Domänen weitergeleiten...

Die Störung wird in der jeweiligen Domäne vom Domain-Fault-Resolution-Team (DFRT) entgegengenommen und überprüft (A_8). In den beiden Fällen, wenn es sich um eine Störungsmeldung handelt, die durch einen Ausfall in der entsprechenden Domänen verursacht worden ist (A_{10}) oder nicht (A_9), wird der GFMT benachrichtigt. Dieser wartet auf alle Antworten aus allen beteiligten Domänen (A_{11}) bzw. auf Behebung der Störung, wenn diese bestätigt wurde (A_{14}). Wenn aber die Störung in keiner der Domänen bestätigt wird, startet das GFMT eine Eskalationsprozedur, die zur Lösung des Problems führen soll.

und lösen

Wird die Störung bestätigt, sucht das betreffende DFRT die Fehlerursache und löst das Problem (A_{12}) für die entsprechende Domäne. Das DFRT meldet die Störungsaufhebung an das GFMT und dieses kann alle Domänen darüber informieren (A_{15}) und die Störung abschließen (A_{16}).

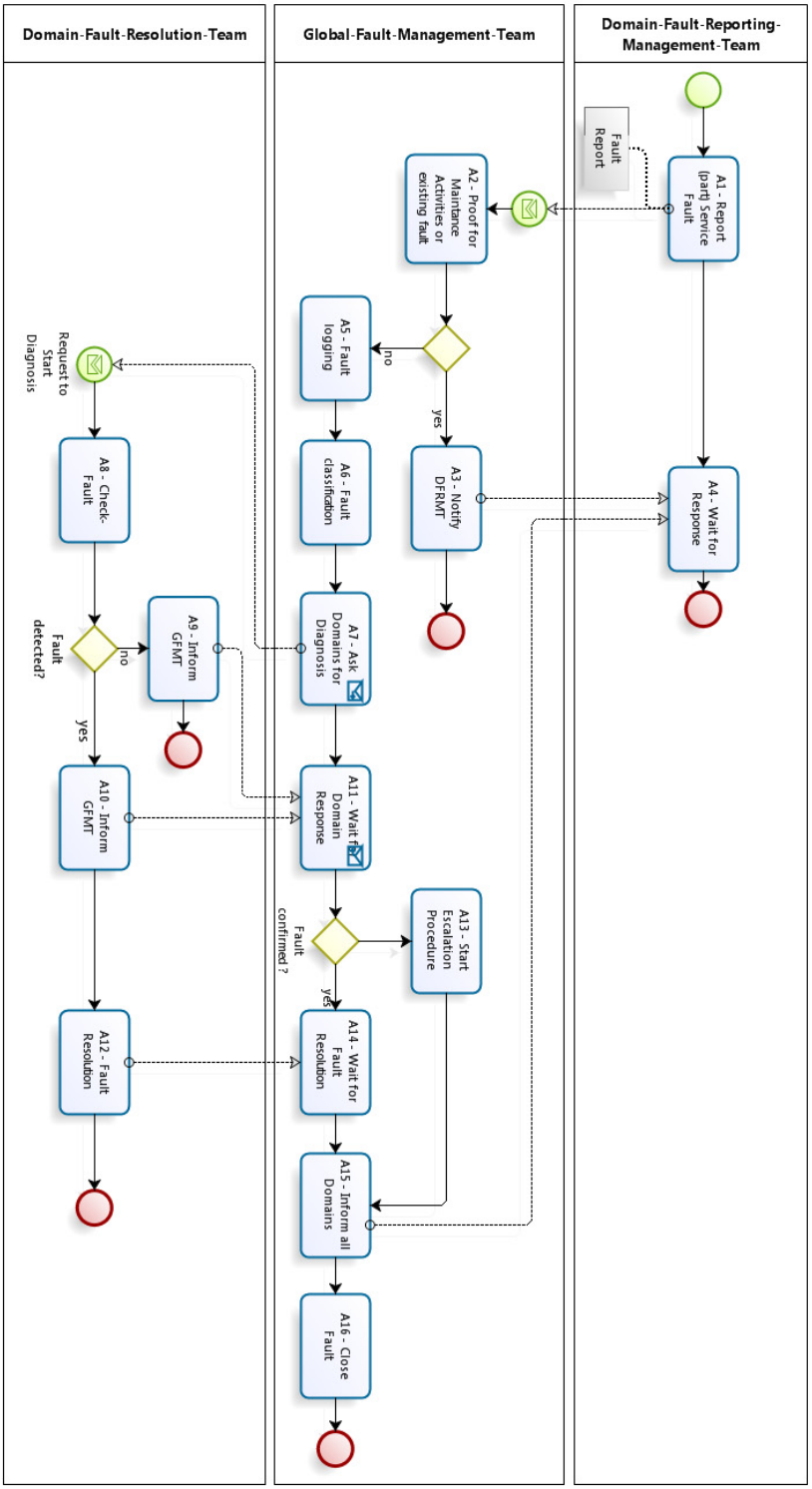


Abbildung 3.9.: Fehlermanagement-Prozess für das verallgemeinerte Szenario

Anwendungsfall		Beschrieben auf
Fehlerlokalisierung	L01: in der eigenen Domäne	Seite 65
	L02: interorganisational	Seite 67
	L03: in einer anderen Domäne	Seite 68
Statusanzeige	P01: der Fehlerbearbeitung	Seite 71
	P02: der Wartungsarbeit	Seite 72
Überwachung	M01: Domänenüberwachung	Seite 75
	M02: Gesamtstatusüberwachung	Seite 75
	M03: eines interorg. Dienstes	Seite 76
Reporting	R01: Statistiken, Accountingdaten	Seite 79
	R02: QoS-Parameter	Seite 80
	R03: Trendanalyse	Seite 81
False-Positives	F01: Lokalisierung	Seite 84
	F02: Markierung	Seite 84

Tabelle 3.7.: Übersicht der beschriebenen Anwendungsfälle

Zu bemerken ist hier, dass die Rollen im Allgemeinszenario dynamisch vergeben werden. Eine Rolle wird meistens von einem Team übernommen. Beispielsweise wenn eine Störungsmeldung in einer Domäne ,,gemeldet" wird, dann hat diese die Rolle des DFRMT; sie kann aber auch gleichzeitig auch den Hut des GFMT übernehmen und die anderen beteiligten Domänen benachrichtigen. Wenn das GFMT eine Meldung bekommt und es ,,gerade" in einer bestimmten Domäne angesiedelt ist, überprüft man auch in dieser Domäne den Ausfall, aber in der Rolle des DFRT.

3.2. Anforderungen an die ioFMA

Im Abschnitt 3.1.4 wurden allgemeine Voraussetzungen an das interorganisationale Fehlermanagement geschildert und dabei der interorganisationale Fehlermanagement-Ablauf beschrieben. In diesem Abschnitt werden daraus folgende Anforderungen an die interorganisationale Fehlermanagementarchitektur (ioFMA) abgeleitet. Die Anforderungen werden in zwei Kategorien geteilt, in funktionale Anforderungen, die auf die erforderlichen Funktionalitäten der Architektur zielen, und nicht-funktionale Anforderungen, die sich auf die Verwendbarkeit, Verfügbarkeit, Leistungsfähigkeit und Wartbarkeit beziehen.

Diese Anforderungen werden anhand der Beschreibung von Anwendungsfällen (use cases) ermittelt. Im Abschnitt 3.2.1 werden zuerst die Akteure, die an den Anwendungsfällen beteiligt sind, beschrieben. Im Abschnitt 3.2.2 werden die fünf Kategorien von Anwendungsfällen

mit deren Unterkategorien beschrieben (siehe Tabelle 3.7). Nach der Beschreibung der jeweilige Kategorie von Anwendungsfällen werden Anforderungen an diese zusammengefasst. Abschnitt 3.2.3 fasst diese Anforderungen zu einem Anforderungskatalog an eine ioFMA zusammen.

3.2.1. Beschreibung der Akteure

Aus den oben beschriebenen realen Szenarien wie auch aus dem Allgemeinszenario werden folgende Akteure identifiziert. Wie schon in Abschnitt 3.1.4.2 kurz angeschnitten gibt es einige Rollen, die von Belang im interorganisationalen Fehlermanagement sind [MaHo 10]. Allerdings beteiligen sich die Akteure, die Informationen mit dem System von außen austauschen, in diversen Rollen am interorganisationalen Fehlermanagementprozess. Zur Spezifikation der Anforderungen ist zunächst die Beschreibung der Akteure, die mit der ioFMA interagieren, notwendig.

Aktor *Nutzer des Dienstes*	
Bezeichner	user
Beschreibung	Endnutzer eines Dienstes
Beispiele	Die LHCOPN-Mitarbeiter möchten den Zustand ihres Netzes betrachten.
Assoziierte Anwendungsfälle	L01

Tabelle 3.8.: Zusammenfassung des Aktors *Nutzer des Dienstes*

Wenn man über Fehlermanagement im Allgemeinen spricht, ist ein sehr wichtiger Aktor der Nutzer des Dienstes. Auch in interorganisationalen Umgebungen spielt der Nutzer des Dienstes eine wichtige Rolle (s. Abbildung 3.7), da er derjenige ist, der den Fehlermanagementprozess anstößt. Fehlermeldungen von Nutzern werden meist über Trouble Ticket System (TTS) oder Incident Ticketing Systeme (ITS) angenommen und zur vollständigen Fehlerdiagnose bzw. -bearbeitung weitergeleitet. In interorganisationalen Umgebungen kann allerdings der Nutzer eines Dienstes auch ein anderer Service Provider sein, der ähnlich wie ein Nutzer die Dienste der anderen Provider nutzt.

Service Provider (SP) sind diejenigen Akteure, die den Dienst für den Kunden bereitstellen und für die Einhaltung der versprochenen SLAs gegenüber diesem verantwortlich sind. Im interorganisationalen Fehlermanagement spielen diese eine wichtige Rolle, da sie diejenigen sind, die *gemeinsam* das Provider-Netzwerk ausmachen und die Dienste auch *gemeinsam* erbringen (siehe Abbildung 3.6). Dieser Aktor kann gegebenenfalls auch als Nutzer agieren; allerdings wird allgemein ein Service Provider für die folgenden Anwendungsfälle als Diensterbringer betrachtet.

Aktor *Service Provider*	
Bezeichner	SP
Beschreibung	Jeder IT-Dienstleister innerhalb einer interorganisationalen Umgebung
Beispiele	keine spezifischen Rollen in den Szenarien
Assoziierte Anwendungsfälle	alle

Tabelle 3.9.: Zusammenfassung des Aktors *Service Provider*

Aktor *Domain-Fault-Manager*	
Bezeichner	DFM
Beschreibung	Überwachung des lokalen Fehlermanagementprozesses
Beispiele	keine spezifischen Rollen in den Szenarien
Assoziierte Anwendungsfälle	alle

Tabelle 3.10.: Zusammenfassung des Aktors *Domain-Fault-Manager*

In jedem Service Provider-Bereich existieren zuständige Personen für das lokale Fehlermanagement. Der Aktor Domain-Fault-Manager (DFM) ist derjenige, der innerhalb einer Domäne die Erkennung, Aufnahme, Weiterleitung, Bearbeitung und die Verfolgung einer Störung überwacht. Er behält zu jeder Zeit den gesamten Fehlermanagementprozess im Auge und sorgt für seine Einhaltung. Sehr oft sind die zwei Akteure Service Provider und Domain-Fault-Manager identisch, allerdings sind sie von der genauen Verantwortlichkeit unterschiedlich; der DFM ist einem SP untergeordnet.

Der DFM kommuniziert mit DFMs anderer Domänen und daher ist dieser Akteur für das interorganisationale Fehlermanagement so wichtig. Innerhalb der Domänen existieren, wie auch bei der Rollenbeschreibung für das Allgemeinszenario in Abschnitt 3.1.4.2, lokale Teams, die für die Behebung von Fehlern zuständig sind. Der Aktor Domain-Fault-Operator (DFO) übernimmt, wie der Name auch andeutet, die tatsächliche Fehlersuche und danach die Fehlerbehebung innerhalb einer Domäne. Er liefert wichtige Informationen an den DFM, der diese wiederum anderen Domänen mitteilt.

Der Global-Fault-Coordinator-Manager (GFCM) hat im interorganisationalen Fehlermanagement hauptsächlich eine koordinierende Rolle. Er überwacht den Fehlermanagementprozess und gegebenenfalls involviert er sich in der Weiterleitung von Informationen bzw. in der Interdomänen-Kommunikation. Im Fall einer hierarchischen Form der Diensterbrin-

Aktor *Domain-Fault-Operator*	
Bezeichner	DFO
Beschreibung	Fehlerbearbeitung und -eskalation
Beispiele	Die Spezialisten des LRZ beheben Fehler, die die Kompetenz der LRZ Hotline überschreiten.
Assoziierte Anwendungsfälle	L01, L02, L03, P01, P02, F01, F02

Tabelle 3.11.: Zusammenfassung des Aktors *Domain-Fault-Operator*

Aktor *Global-Fault-Coordination-Manager*	
Bezeichner	GFCM
Beschreibung	Koordination, Überwachung des io-Fehlermanagement-Prozesses
Beispiele	E2ECU und/oder TUM Service Desk koordinieren und überwachen Ticketweiterleitung, -bearbeitung und -status
Assoziierte Anwendungsfälle	L02, L03, P01, P02, M02, M03, R01, R02, R03; F01

Tabelle 3.12.: Zusammenfassung des Aktors *Global-Fault-Coordination-Manager*

gung ist der GFCM gleich dem DFM des ,,verantwortlichen Provider". In Fall der heterarchischen Form der Diensterbringung wird die Rolle zeitweise (z.B. pro Störung) jeder Domäne zugeschrieben; die Rolle kann durch mehrere Domänen eingenommen werden, seine Zuständigkeit ist aber immer dieselbe, egal in welcher Domäne er gerade angesiedelt ist.

Das Domain-Monitoring-System (DMS) ist zuständig innerhalb einer Domäne für die Überwachung der Systeme. Es benachrichtigt durch Meldung oder Alarme bei einen Fehlverhalten eines Dienstes der DFM. In Tabelle 3.14 werden alle hier beschriebenen Akteure zusammengefasst.

3.2.2. Beschreibung der Anwendungsfälle

Eine interorganisationale Fehlermanagementarchitektur (ioFMA) muss unterschiedliche Unterstützungsmechanismen für die obengenannten Akteure betrachten, da deren Aufgaben

Aktor *Domain-Monitoring-System*	
Bezeichner	DMS
Beschreibung	Meldung von Systemfehlern, Alarmen
Beispiele	Das E2EMon Tool im LCG Szenario liefert den Zustand der Teilsegmente.
Assoziierte Anwendungsfälle	L01, M01, M02, M03, R01, R02, R03

Tabelle 3.13.: Zusammenfassung des Aktors *Domain-Monitoring-System*

Aktor		
Bezeichner	Name	Beschreibung
user	Nutzer des Dienstes	Endnutzer eines Dienstes
SP	Service Provider	Jeder einzelne IT-Dienstleister innerhalb einer interorganisationalen Umgebung
DFM	Domain-Fault-Manager	Überwachung des lokalen Fehlermanagementprozesses
DFO	Domain-Fault-Operator	Fehlerbearbeitung und -eskalation
GFCM	Global-Fault-Coordination-Manager	Koordination, Überwachung des io-Fehlermanagement-Prozesses
DMS	Domain-Monitoring-System	Meldung von Systemfehlern, Alarmen

Tabelle 3.14.: Übersicht der beschriebenen Aktoren

hinsichtlich des Fehlermanagements unterschiedlich sind. Um diese Mechanismen zu beschreiben, werden die Anwendungsfälle definiert. In Abschnitt 3.1.4.2 wurden die relevanten Eigenschaften für das io-FM zusammengefasst, die möglichen Funktionalitäten anhand der Hauptaufgaben und Teilaufgaben des Funktionsbereichs Fehlermanagement (intraorganisational) und der Beschreibung der drei interorganisationalen Szenarien (zwei reale und ein übergreifendes abstraktes Szenario) beschrieben.

Diese dienen als Basis zur Beschreibung von Anwendungsfällen. Bezug nehmend auf diese Funktionalitäten muss die Managementarchitektur insbesondere Fehlerlokalisierung, Fehlerbearbeitung, Monitoring, Reporting und Beseitigung der Fehlalarmen in interorganisationalen Umgebungen unterstützen. Folglich werden die Anwendungsfälle dementsprechend gruppiert. Die Anwendungsfälle unterstützen alle Phasen des Lebenszyklus eines Fehlers. Da das

Anwendungsfall	Phasen des Lebenszyklus eines Fehlers			
	Entdecken	Eingrenzen	Beheben	Vorhersage/-Vorbeugung
Fehlerlokalisierung	✓	✓		
Statusanzeige		✓	✓	
Überwachung	✓	✓	✓	✓
Reporting	✓	✓		✓
False-Positives		✓	✓	✓

Tabelle 3.15.: Abdeckung der Phasen des Lebenszyklus eines Fehlers durch die Anwendungsfälle

interorganisationale Fehlermanagement z. T. die intraorganisationale Realisierung beinhaltet, wurden manche Anwendungsfälle in Unterkategorien verfeinert.

Die hier dargestellten Anwendungsfälle sind nicht vollständig, aber sie repräsentieren eine Mindestmenge (hinreichende Menge) an Anwendungsfällen, die den Problemraum dieser Arbeit umspannen.

Abbildung 3.15 stellt die unterschiedlichen Klassen an Anwendungsfällen und der Einordnung in die Phasen des Lebenszyklus dar.

Zur Ableitung der Anwendungsfälle wird die Vorgehensweise, die von Hennicker in [Henn 06] bzw. Jacobson et al. in [BRJ 06] beschrieben ist, angewendet. Diese Vorgehensweise besteht aus folgenden Schritten:

- Bestimmung der Akteure, die mit dem System interagieren. (Wer benutzt das System? Wer holt/liefert Informationen von dem/für das System?)
- Bestimmung der Anwendungsfälle aufgrund der Aufgaben, die das System für jeden einzelnen Akteur erledigen soll.
- Erstellung eines Anwendungsfall-Diagramms, ggf. mit kurzer Beschreibung der Akteure und Use Cases.
- Beschreibung der Anwendungsfälle (iterativ)

3.2.2.1. Anwendungsfälle bezüglich Fehlerlokalisierung

Die wichtigste Funktionalität einer Managementarchitektur für interorganisationales Fehlermanagement ist die Unterstützung der Fehlerlokalisierung. Diese Funktionalität wird

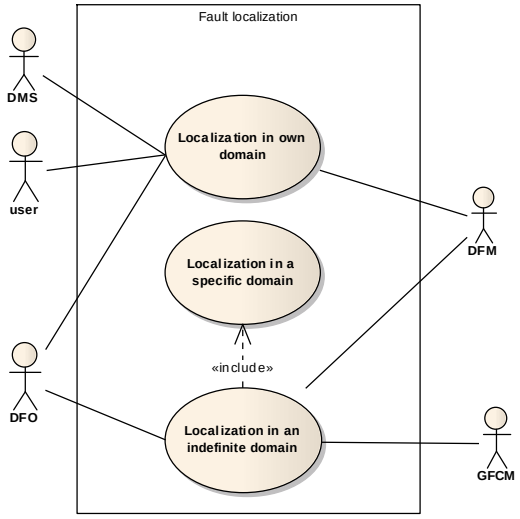


Abbildung 3.10.: Fehlerlokalisierung: zusammenfassende Anwendungsfälle

in drei Anwendungsfälle eingeteilt, die sich gegenseitig ergänzen: Fehlerlokalisierung in der eigenen Domäne, in irgendeiner anderen Domäne und in einer bestimmten anderen Domäne.

Fehlerlokalisierung in der eigenen Domäne (L01)

Anwendungsfall L01

Kurzbeschreibung: In diesem Anwendungsfall wird nach einer Störungsmeldung seitens des Nutzers oder des DMS der darunterliegende Fehler in der eigenen Domäne gesucht, lokalisiert und gelöst.

Initiierender Aktor: Nutzer des Dienstes (user) oder Domain-Monitoring-System (DMS)

Vorbedingungen: Es existiert bereits innerhalb der Domäne ein Fehlermanagement (Ticketing) System.

L01-Primärszenario: Erfolgreiches Lokalisieren eines Fehlers direkt vom DFM.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. Nutzer meldet eine Störung.
2. Domain-Fault-Manager (DFM) nimmt die Störung auf.
3. DFM schaut in der ioFMA nach bereits existierender Lösung.
4. DFM benachrichtigt den user.
5. DFM schließt die Störungsmeldung ab.

Nachbedingungen: Der Fehler wurde lokalisiert, der User darüber benachrichtigt.

L01-Sekundärszenario I: Kein bekannter Treffer in der ioFMA.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. Vorgehensweise wie im Primärszenario.
2. Die Suche in der ioFMA nach bereits existierender Lösung hat keine Lösung gebracht.
3. DFM leitet den Fehler zum Domain-Fault-Operator (DFO) zur Lösung weiter. Wartet auf Antwort.
4. DFO findet den Fehler mit Hilfe des DMS und löst ihn.
5. DFO meldet die Fehlerlösung an DFM.
6. Weiter mit Schritt 4. in Primärszenario.
7. DFM dokumentiert den Fehler und dessen Lösung in der ioFMA.

Nachbedingungen: wie Primärszenario. Ein neuer Fehler und dessen Lösung wurde zu der ioFMA hinzugefügt.

L01-Sekundärszenario II: Fehler kann in dieser Domäne nicht lokalisiert werden.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. Vorgehensweise wie im Primärszenario bzw. Sekundärszenario I.
2. Nachdem der DFO die Fehlermeldung vom DFM entgegennimmt, findet er den Fehler nicht oder kann ihn innerhalb der Domäne nicht lösen.
3. DFO meldet dies dem DFM.
4. DFM dokumentiert den Fehler in der ioFMA.
5. DFM leitet den Fehler an Global-Fault-Coordination-Manager (GFCM) weiter.

Nachbedingungen: Fehler noch nicht lokalisiert und nicht gelöst.

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

- ALLE Fehlermeldungen müssen aufgenommen und gespeichert werden.
- Zwischen den unterschiedlichen Domänen sollten definierte Schnittstellen existieren.

Interorganisationale Fehlerlokalisierung (L02)

Anwendungsfall L02

Kurzbeschreibung: Die Störungsmeldung hat eine unbekannte Ursache in der eigenen Domäne und wird zur Lokalisierung in eine andere Domäne weitergeleitet.

Initiierender Aktor: Domain-Fault-Manager (DFM)

Vorbedingungen: Der Fehler wurde bereits in einer der Domänen aufgenommen.

L02-Primärszenario: Erfolgreiches Lokalisieren eines Fehlers.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. DFM leitet den Fehler an den Global-Fault-Coordination-Manager (GFCM) weiter.
2. GFCM benachrichtigt alle beteiligte DFMs.
3. Jeder DFM leitet die Störungsmeldung zu den eigenen DFOs weiter.
4. Die DFOs versuchen, den Fehler zu lokalisieren und lösen.
5. Die DFOs antworten den entsprechenden DFMs.
6. Die DFMs leiten die Antwort an GFCM weiter.
7. Der GFCM meldet an alle Domänen die Lösung des Fehlers.

Nachbedingungen: Der Fehler wurde lokalisiert, alle Domänen wurden darüber benachrichtigt.

L02-Sekundärszenario I: Der Fehler ist in keiner der beteiligten Domäne gefunden und/ oder gelöst.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. Vorgehensweise wie im Primärszenario.
2. Nachdem der GFCM die Antworten der Domänen überprüft und feststellt, dass keine Lokalisierung bzw. Lösung aus den Domänen kam, leitet er Eskalationsprozedur(en) ein.
3. Abschließend wie in Primärszenario.

Nachbedingungen: Alle Domänen wurden darüber benachrichtigt und warten auf die Fehlerlösung.

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

- Im Fall einer hierarchischen Form der Diensterbringung ist der DFM gleich der GFCM

- Im Fall einer hierarchischen Form der Diensterbringung kommuniziert der GFCM nicht allen Domänen das Ergebnis seines Befundes.

Anwendungsfall L03

Fehlerlokalisierung in einer anderen Domäne (L03)

Kurzbeschreibung: das ist ein Spezialfall von L02, in dem die Störungsmeldung eine unbekannte Ursache in der eigenen Domänen hat und zur Lokalisierung einer bestimmten anderen Domäne zugewiesen und weitergeleitet wird.

Initiierender Aktor: Domain-Fault-Manager (DFM)

Vorbedingungen: Der Fehler wurde bereits in einer der Domänen aufgenommen und einer bestimmten (anderen) Domäne zugewiesen.

L03-Primärszenario: Erfolgreiches Lokalisieren eines Fehlers.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. DFM leitet den Fehler an dem Global-Fault-Coordination-Manager (GFCM) oder an den bekannten DFM weiter.
2. Wenn noch nicht geschehen, benachrichtigt der GFCM den DFM der betroffenen Domäne.
3. DFM leitet die Störungsmeldung zum DFO weiter.
4. Der DFO lokalisiert und löst den Fehler.
5. Der DFO antwortet dem DFM.
6. Der DFM leitet die Antwort an den GFCM weiter.
7. Gegebenenfalls meldet der GFCM an alle Domänen die Lösung des Fehlers.

Nachbedingungen: Der Fehler wurde lokalisiert, alle Domänen wurden darüber benachrichtigt.

L03-Sekundärszenario I: Der Fehler ist in der bestimmten Domäne nicht gefunden und/oder gelöst.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. Vorgehensweise wie im Primärszenario.
2. Nachdem der GFCM eine negative Antwort von den DFM der betroffenen Domäne erhalten hat, folgt er den Schritten in L02 weiter.

Nachbedingungen: Alle Domänen wurden über die Fehlerbearbeitungsstatus benachrichtigt und warten eventuell auf die Lösung.

Eine Zusammenfassung der Anwendungsfälle bezüglich der Fehlerlokalisierung ist graphisch in Abbildung 3.10 dargestellt. Auf der Basis der hier beschriebenen Anwendungsfälle werden Anforderungen an das Informations-, Organisations-, Funktions- und Kommunikationsmodell des ioFMA zusammengefasst. Zu bemerken ist, dass jeder Anwendungsfall in sich eine funktionale Anforderung an das Funktionsmodell ist. Trotzdem ergeben sich noch zusätzliche Anforderungen an das Funktionsmodell, die hier zusätzlich noch erwähnt werden.

Anwendungsfälle L01-L03: Anforderungen an das Informationsmodell

Informationsmodell

Schnittstellendefinition Für die Fehlerlokalisierung in interorganisationalen Umgebungen ist äußerst wichtig, eine definierte Schnittstelle zwischen den Domänen zu haben. Diese unterstützt die Interdomänen-Kommunikation, indem sie festlegt, welche Informationen, von wem und in welchem Format übergeben werden. In Abwesenheit einer Schnittstelle können keine Fehlermeldungen zwischen den Domänen kommuniziert werden, also ist auch keine Fehlerlokalisierung möglich.

Gemeinsames Fehlermeldungsformat Um die Daten (Fehlermeldungen) aus anderen Domänen entsprechend einzuordnen, wird ein gemeinsames Datenaustauschformat benötigt. Es wird als Fehlermeldungsformat benannt, da es sich bei der Fehlerlokalisierung hauptsächlich um den Austausch von Fehlermeldungen handelt. In der Tat müssen diese in allen beteiligten Domänen gleichmäßig definiert werden, oder zumindest gemeinsame Attribute haben, um die Fehlerweitermeldung oder Fehlerverfolgung zu unterstützen.

Konvertierungsmethoden Wenn kein gemeinsames Datenformat gegeben ist, sollten zumindest Konvertierungsmethoden von einem proprietären Fehlermeldungsformat (innerhalb einer Domäne) zu dem allgemeingültigen in der io-Umgebung verfügbar sein.

Fehlerlebenszyklus Das Informationsmodell muss das Entdecken, Eingrenzen und Beheben von Fehlern unterstützen. Die Fehlerlokalisierung bezieht sich auf das Entdecken und Eingrenzen und liefert notwendige Informationen für das Beheben von Fehlern. Es muss jederzeit nachvollziehbar sein, in welchem Zustand sich ein Fehler befindet.

Anwendungsfälle L01-L03: Anforderungen an das Organisationsmodell

Organisationsmodell

io-Formen der Diensterbringung Die interorganisationale Form der Diensterbringung erfordert eine intensive Interdomänen-Kommunikation. Bei der Fehlerlokalisierung ist anzunehmen, dass unterschiedliche Aktionen unterschiedlich ablaufen können – abhängig von der io-Form der Diensterbringung.

Rollendefinition Dabei handelt sich um die Definition von Rollen, die zuständig für die Lokalisierung von Fehlern innerhalb einer Organisation und organisationsübergreifend sind.

Anwendungsfälle L01-L03: Anforderungen an das Funktionsmodell

Funktionsmodell

Visualisierungsmöglichkeit Für die Fehlerlokalisierung sollte die Möglichkeit der Visualisierung gegeben werden, Fehler nachvollziehen zu können. Dies dient auch zur Verbesserung der Übersichtlichkeit der Fehlermeldungen und der Beschleunigung des Lokalisierungsprozesses.

Kommunikationsmodell

Anwendungsfälle L01-L03: Anforderungen an das Kommunikationsmodell

Kommunikationsmechanismen Die Fehlerlokalisierung ist auf die von der ioFMA unterstützten Kommunikationsmechanismen angewiesen. Idealerweise sollte die Managementarchitektur sowohl Push- als auch Pull-Mechanismen unterstützen. Der erste Fall (Pull-Modus) bezieht sich auf die Situation, in der bei Fehlerlokalisierung explizit bestimmte Informationen aus anderen Domänen angefordert werden. Im Fall des Push-Modus stehen Informationen über mögliche Fehlermeldungen der ioFMA bereits zur Verfügung, bevor die Fehlerlokalisierung begonnen hat. Diese Informationen werden vom Managementsystem regelmäßig aktualisiert.

Anwendungsfälle L01-L03: Nicht-funktionale Anforderungen

NF Anforderungen

Zugriffskontrolle Nur autorisierte und authentifizierte Rollen bzw. Personen können die Fehlerlokalisierung in ,,Fremddomänen" durchführen oder weiter delegieren. Die ioFMA darf bei der Fehlerlokalisierung entsprechend nur für Berechtigte zur Verfügung stehen.

Datenintegrität Hinreichende Sicherheitsmechanismen sollen die Integrität der zu kommunizierenden Informationen bei Fehlerlokalisierung gewährleisten.

Skalierbarkeit Die Fehlerlokalisierung sollte sowohl für eine oder zwei Domänen als auch für größere Hierarchien oder Heterarchien von Service Providern durchführbar sein.

Performanz Die Lösungszeit ist ein entscheidender Faktor im io-FM. Eine schnelle gezielte Lokalisierung auch bei mehreren beteiligten Domänen trägt zu einer schnelleren Fehlerlösung bei.

Gemeinsame Informationsdatenbank Dies kann z. B. der Datenbestand aus einem Ticketing System sein, auf den alle beteiligten Domänen Zugriff haben. Allerdings ist eher unwahrscheinlich, dass mehrere Domänen freien Zugriff auf ihrer Informationen zulassen und deswegen empfiehlt sich z. B. eine ioCMDB wie in [HoKn 08].

3.2.2.2. Anwendungsfälle bezüglich des Status der Fehlerbearbeitung

Die Statusanzeige der Fehlerbearbeitung ist eine wichtige Funktionalität, da die DFMs, GFCM und letztlich auch der Kunde darauf angewiesen sind zu wissen, wann ein Dienst wieder funktionsfähig ist. Da es sich bei einem Ausfall nicht nur um Fehler handelt, sondern

auch um vorangekündigte Wartungsarbeiten, werden die beiden Anwendungsfälle hierunter zusammengefasst.

Statusanzeige der Fehlerbearbeitung (P01)

Anwendungsfall P01

Kurzbeschreibung: Dieser Anwendungsfall behandelt die Statusüberwachung der Fehlerbearbeitung.

Initiierender Aktor: DFM, GFCM

Vorbedingungen: Der Fehler wurde bereits lokalisiert.

P01-Primärszenario: Ermitteln des Fehlerbearbeitungsstatus

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. Der DFM oder GFCM fragt den DFM des Bereiches, wo der Fehler lokalisiert wurde, nach dem Bearbeitungsstand.
2. Der DFM fragt beim DFO nach.
3. Der DFO aktualisiert den Bearbeitungsstatus.
4. Der DFM leiten die Antwort an den GFCM oder den DFM (der angefragt hatte) weiter.
5. Gegebenenfalls veröffentlicht der GFCM den Bearbeitungszustand des Fehlers.

Nachbedingungen: Bearbeitungsstatus der Fehler wurde ermittelt.

P01-Sekundärszenario I: Nutzer fragt nach dem Fehlerbearbeitungsstatus

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. Nutzer fragt den Fehlerbearbeitungsstatus beim GFCM nach.
2. Vorgehensweise wie im Primärszenario.
3. Der GFCM informiert Nutzer über den Bearbeitungszustand des Fehlers.

Nachbedingungen: Der Bearbeitungsstatus der Fehler wurde ermittelt.

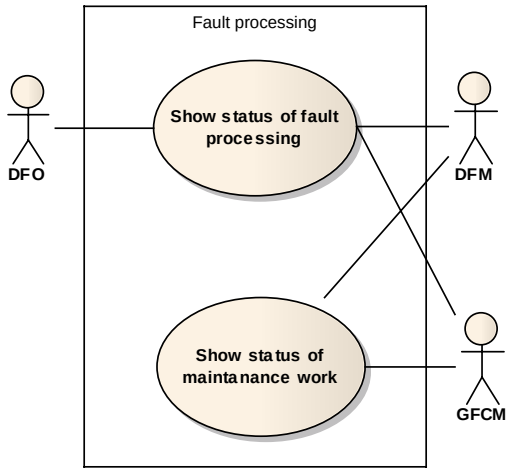


Abbildung 3.11.: Anwendungsfälle bezüglich des Status der Fehlerbearbeitung

Statusanzeige von Wartungsarbeiten (P02)

Kurzbeschreibung: Dieser Anwendungsfall behandelt die Statusüberwachung der Wartungsarbeit. Es handelt sich um einen Spezialfall von P01.

Initiierender Aktor: DFM, GFCM

Vorbedingungen: Die Wartungsarbeit wurde bereits angekündigt. Der Dienst ist deswegen ausgefallen. Der Ausfall wird nicht als Fehler gewertet.

P02-Primärszenario: Ermitteln des Status der Wartungsarbeit

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. DFM oder GFCM fragt den DFM des Bereiches, wo die Wartungsarbeit angekündigt wurde, nach dem Bearbeitungsstand.
2. Der DFM fragt beim DFO nach.
3. Der DFO aktualisiert den Bearbeitungsstatus.
4. Der DFM leitet die Antwort an den GFCM oder den DFM (der angefragt hatte) weiter.
5. Gegebenenfalls veröffentlicht der GFCM den Wartungsstatus.

Nachbedingungen: Ein Wartungsstatus wurde ermittelt.

P02-Sekundärszenario I: Nutzer fragt nach dem Wartungsstatus

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. Ein Nutzer fragt den Wartungsstatus beim GFCM nach.

2. Vorgehensweise wie im Primärszenario.
3. Der GFCM informiert den Nutzer über den Wartungsstatus.

Nachbedingungen: Der Bearbeitungsstatus der Fehler wurde ermittelt.

Abbildung 3.11 stellt zusammenfassend die Anwendungsfälle bezüglich des Status der Fehlerbearbeitung dar. Folgende Anforderungen an die Modelle der ioFMA folgen daraus:

Anwendungsfälle P01-P02: Anforderungen an das Informationsmodell *Informationsmodell*

Einheitliches Informationsformat Bei der Statusüberwachung der Fehlerbearbeitung und /oder Wartungsarbeiten ist ein einheitliches Datenformat unabdingbar. Wenn diese Arbeiten in unterschiedlichen Domänen stattfinden und der Status in anderen Domänen angezeigt wird, müssen diese dieselbe Symbolik und dieselbe Interpretation verwenden.

Fehlerlebenszyklus Der globale (interorganisationale) Fehlerlebenszyklus muss von der ioFMA unterstützt werden, um den Status bezüglich Entdecken, Eingrenzen und Beheben von Fehlern als auch über den Fortschritt der Wartungsarbeiten bereitstellen zu können.

Anwendungsfälle P01-P02: Anforderungen an das Organisationsmodell *Organisationsmodell*

io-Formen der Diensterbringung Abhängig von der interorganisationalen Form der Diensterbringung muss der Status der Fehler- bzw. Wartungsarbeiten über alle beteiligten Domänen ermittelt werden können.

Rollendefinition Wer innerhalb einer Organisation bzw. interorganisational den Bearbeitungsstatus anschauen und/oder anfordern kann, wird durch die konkrete Rollendefinition für diesen Zweck geregelt.

Anwendungsfälle P01-P02: Anforderungen an das Funktionsmodell

Visualisierungsmöglichkeit Die Statusanzeige kann durch Visualisierung unterstützt werden, damit eine Übersichtlichkeit über die überwachten Domänen erzielt wird.

Statusänderung Insbesondere für den Bearbeiter eines Fehlers ist es notwendig, den Status der Fehlerbearbeitung ändern zu können. *Funktionsmodell*

Anwendungsfälle P01-P02: Anforderungen an das Kommunikationsmodell *Kommunikationsmodell*

Kommunikationsmechanismen Für die Statusanzeige ist ein hybrider Kommunikationsmechanismus notwendig. Normalerweise wird der Status der Fehlerbearbeitung bzw. die Wartungsarbeit durch ein Pull-Mechanismus angefordert, um den momentanen aktuellen Stand zu ermitteln. Wünschenswert ist aber auch ein Push-Mechanismus, der in regelmäßigen Zeitabständen den Status aller Fehlerbearbeitungen und Wartungsarbeiten bereitstellt.

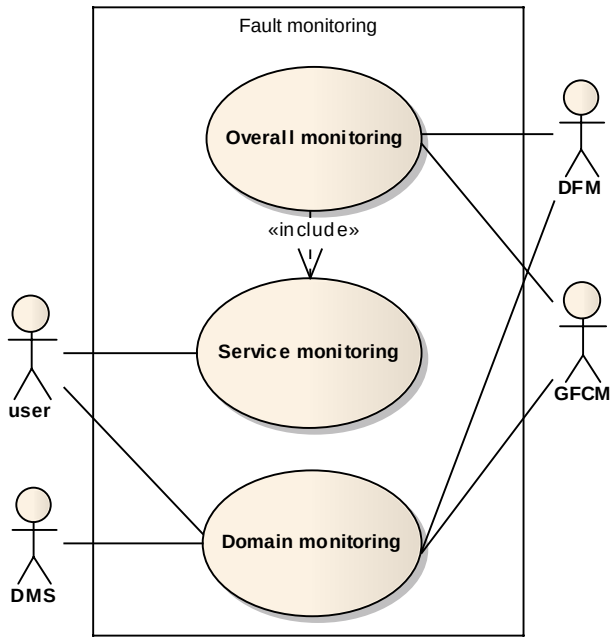


Abbildung 3.12.: Anwendungsfälle bezüglich Monitoring

NF Anforderungen

Anwendungsfälle P01-P02: Nicht-funktionale Anforderungen

Automatisierung Um eine schnelle und effiziente Anzeige des Status zu gewährleisten, ist Automatisierung notwendig.

Zugriffskontrolle Nur autorisierte und authentifizierte Rollen bzw. Personen können den Status der Fehlerbearbeitung bzw. Wartungsarbeit in einer fremden Domäne einsehen.

Datenaktualität Die zu kommunizierenden Informationen müssen regelmäßig aktualisiert werden. Diese Anforderung ergibt sich aus dem Pull-Kommunikationsmuster, bei dem immer in regelmäßigen Zeitabständen der Status aktualisiert wird.

3.2.2.3. Anwendungsfälle bezüglich Monitoring

Die Überwachung ist ein wichtiger Punkt sowohl auf intraorganisationaler als auch auf interorganisationaler Ebene. Eine permanente Überwachung des Netzes trägt zu einer schnelleren Fehlerlokalisierung bei. Dies ist eine vorbeugende Funktionalität der ioFMA. Sie meldet nicht nur, ob ein Teildienst funktioniert, sondern auch, ob die Funktionalität vollständig ist (z. B. Partialausfälle, die den Gesamtdienst nicht ganz unterbrechen). Die Anwendungsfälle, die dabei betrachtet werden müssen, sind: die Überwachung einer Domäne, eines Dienstes (insgesamt) und die Gesamtüberwachung des Providernetzwerkes.

Domänenüberwachung (M01)

Anwendungsfall M01

Kurzbeschreibung: Bei diesem Anwendungsfall handelt es sich um die Überwachung einer an der Diensterbringung beteiligten Domäne.

Initiierender Aktor: DFM, GFCM, user

Vorbedingungen: Die Domäne besitzt ein Monitoringsystem.

M01-Primärszenario: Realisierung der Domänenüberwachung.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. Der DFM, der GFCM oder der user fragt den DFM eines bestimmten Bereiches nach dem Status des Teildienstes.
2. Der DFM fordert die Informationen von Domain-Monitoring-System (DMS) an.
3. Der DMS stellt die Überwachungsdaten bereit.
4. Der DFM überreicht den Domänenstatus an GFCM.
5. Domänenstatus wird bekannt gegeben.

Nachbedingungen: Die Domäne wird überwacht.

M01-Sekundärszenario I: Die Überwachung einer Fremddomäne kann nicht stattfinden mangels Rechten.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. Vorgehensweise wie im Primärszenario.
2. DFM einer bestimmten Domäne erlaubt die Überwachung nicht.

Nachbedingungen: Eine Überwachung der domäne nicht statt.

Gesamtstatusüberwachung (M02)

Anwendungsfall M02

Kurzbeschreibung: Bei diesem Anwendungsfall handelt es sich um die Überwachung des Gesamtprovidernetzwerks.

Initiierender Aktor: DFM, GFCM

Vorbedingungen: Alle Domänen besitzen ein Monitoringsystem.

M02-Primärszenario: Realisierung der Domänenüberwachung.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. Ein bestimmtes (oder mehrere) DFM fragt beim GFCM den Gesamtstatus nach.

2. GFCM fordert von allen DFM den Status deren Dienste (bzw. Teildiensten).
3. Jeweiliger DFM fordert die Informationen von DMS an.
4. DMS stellt die Überwachungsdaten bereit.
5. Alle DFMs überreichen den jeweiligen Domänenstatus an der GFCM.
6. Der GFCM gibt den Gesamtstatus bekannt.

Nachbedingungen: Das Providernetzwerk wird überwacht.

M02-Sekundärszenario I: Nicht alle Domänen liefern den Status.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. Vorgehensweise wie im Primärszenario.
2. DFM einer bestimmten Domäne erlaubt die Überwachung durch Fremddomänen nicht.
3. GFCM gibt den Teilstatus bekannt.

Nachbedingungen: Eine Teilüberwachung des Gesamtprovidernetzwerkes findet statt.

Anwendungsfall M03

Überwachung eines interorganisationalen Dienstes (M03)

Kurzbeschreibung: Dieser Anwendungsfall ist ein Spezialfall von M02, für die Überwachung eines Dienstes, der interorganisational erbracht wird.

Initiierender Aktor: DFM, user

Vorbedingungen: Alle Domänen besitzen ein Monitoringsystem.

M03-Primärszenario: Realisierung der Dienstüberwachung

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. Ein bestimmter (oder mehrere) DFM oder ein user fragt beim GFCM den Gesamtstatus eines bestimmten Dienstes nach.
2. Der GFCM fordert bei allen DFM der Domänen, die an diesem Dienst beteiligt sind, den Status ihrer Teildienste an.
3. Die jeweiligen DFM fordern die Informationen vom DMS an.
4. Der DMS stellt die Überwachungsdaten bereit.
5. Alle DFMs überreichen den jeweiligen Domänenstatus an den GFCM.
6. Der GFCM gibt den Gesamtstatus des Dienstes bekannt.

Nachbedingungen: Ein interorganisationaler Dienst wird überwacht.

M03-Sekundärszenario I: Nicht alle Domänen liefern den Status.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. Vorgehensweise wie im Primärszenario.
2. DFM einer bestimmten Domäne erlaubt die Überwachung durch Fremddomänen nicht.
3. GFCM gibt den Teilstatus bekannt.

Nachbedingungen: Eine Teilüberwachung des interorganisationalen Dienstes findet statt.

M03-Sekundärszenario II: Der Nutzer besitzt keine Rechte, um den Gesamtstatus des Dienstes anzusehen.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. Vorgehensweise wie im Primärszenario.
2. Der GFCM stellt fest, dass der user nicht die entsprechende Zugriffsrechte besitzt.

Nachbedingungen: Es findet keine Überwachung des Dienstes statt.

In Abbildung 3.12 sind die Anwendungsfälle in Zusammenhang mit der Überwachung zusammengefasst.

Anwendungsfälle M01-M03: Anforderungen an das Informationsmodell *Informationsmodell*

Gemeinsames Informationsformat Das Monitoring liefert die kontinuierliche Gesamtübersicht über alle beteiligten Domänen (Eigene Domänen, Fremddomäne, alle Domänen) bzw. über einen gemeinsam erbrachten Dienst. Hierfür wird ein einheitliches Informationsformat benötigt, um Daten vergleichen und anzeigen zu können.

Konvertierungsmethoden Für den Fall, dass kein gemeinsames Informationsformat vereinbart wurde, müssen zumindest Abbildungsvorschriften zwischen der lokalen Monitoringdaten unterstützt sein.

Schnittstellendefinition Monitoring wird im Kontext der ioFM realisiert, also müssen die Schnittstellen zwischen den beteiligten Domänen klar definiert werden. Kommunikation zwischen den unterschiedlichen Domänen bzgl. Monitoring kann ohne eine geeignete Schnittstellendefinition nicht stattfinden.

Standard-Metriken Die Feststellung von domänenübergreifenden Standard-Metriken ist notwendig, damit die Monitoringdaten in den unterschiedlichen Domänen auch vergleichbar sind.

Fehlerlebenszyklus Der interorganisationale Fehlerlebenszyklus besteht darin, die im Rahmen des intraorganisationalen Fehlerlebenszyklus ermittelten Informationen zu korrelieren, abzubilden und regelmäßig zu aktualisieren.

Organisationsmodell

Anwendungsfälle M01-M03: Anforderungen an das Organisationsmodell

io-Formen der Diensterbringung Beim Monitoring insbesondere eines gemeinsam erbrachten Dienstes sind unterschiedliche Ausprägungen zu berücksichtigen, abhängig von der interorganisationalen Form der Diensterbringung.

Rollendefinition Die initiierenden und auch die partizipierenden Rollen bei Monitoring müssen sorgfältig definiert und mit den entsprechenden Autorisierungsmechanismen abgesichert werden.

Funktionsmodell

Anwendungsfälle M01-M03: Anforderungen an das Funktionsmodell

Visualisierungsmöglichkeit Eine graphische Darstellung kann die Übersichtlichkeit der Monitoringdaten verbessern und anwenderfreundlicher gestalten.

Anwendungsfälle M01-M03: Anforderungen an das Kommunikationsmodell

Kommunikationsmechanismen Für das Monitoring wird grundsätzlich ein Push-Mechanismus benötigt, der in regelmäßigen Zeitabständen Monitoringdaten zur Verfügung stellt.

NF Anforderungen

Anwendungsfälle M01-M03: Nicht-funktionale Anforderungen

Zugriffskontrolle Nur authentifizierten und autorisierten Rollen darf es erlaubt sein, die Monitoringdaten anzuschauen.

Skalierbarkeit Da das Monitoring potentiell für einen Vielzahl von Domänen oder für einen Gesamtdienst über mehrere Domänen hinweg realisiert werden soll, ist es notwendig, dass die Lösung gut skaliert.

Performanz Um kritische Bereiche zu überwachen braucht man die Sicherheit, dass die Monitoringdaten schnell gesammelt und dargestellt werden. Aufgrund dieser Anforderung ist einen gute Skalierbarkeit des Systems erforderlich.

Datenaktualität Die zu auszutauschenden Informationen müssen regelmäßig aktualisiert werden.

Automatisierung Um ein schnelles und effizientes Monitoring zu gewährleisten, ist Automatisierung notwendig.

3.2.2.4. Anwendungsfälle bezüglich Reporting im Bereich Fehlermanagement

Statistiken, Accountingdaten realisieren (R01)

Anwendungsfall R01

Kurzbeschreibung: In diesem Anwendungsfall werden für jeden Dienst eine Reihe von Statistiken bezüglich Fehlerlokalisierung und -bearbeitung (mittlere Lösungszeit, maximale Lösungszeit, Zeit bis zur Fehlerlokalisierung usw.) oder Accountigdaten zur Dienstnutzung bereitgestellt.

Initiierender Aktor: GFCM

Vorbedingungen: Alle Domänen besitzen ein Monitoringsystem und können Statistiken bzgl. der Fehlerbearbeitung für die eigene Domäne bereitstellen.

R01-Primärszenario: Statistiken bzw. Accountingdaten pro zusammengesetztem Dienst werden bereitgestellt.

Ablauf:

1. Der GFCM fordert bei allen DFM der am Dienst beteiligten Domänen die Daten der Teildienste an.
2. Die jeweiligen DFM fordern die Informationen aus dem DMS an.
3. Das DMS stellt die Daten bereit.
4. Alle DFMs senden die jeweiligen Daten an den GFCM.
5. Der GFCM fasst die Daten zusammen.
6. Der GFCM stellt die Berichte bereit.

Nachbedingungen: Statistiken bzw. Accountingdaten bereitgestellt.

R01-Sekundärszenario I: Nicht alle Domänen können Statistiken und/oder Accountingdaten bereitstellen.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. Vorgehensweise wie im Primärszenario.
2. Der DFM meldet, dass für einen Bereich keine Statistiken vorhanden sind.

Nachbedingungen: Die Statistiken oder Accountingdaten können nicht bereitgestellt werden für den Gesamtdienst.

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

- Das Informationsmodell muss für die Statistik bzw. die Accountingdaten ein einheitliches Datenformat beinhalten.
- Der Zugriff der DFMs auf bestimmte Domänen muss geregelt sein.

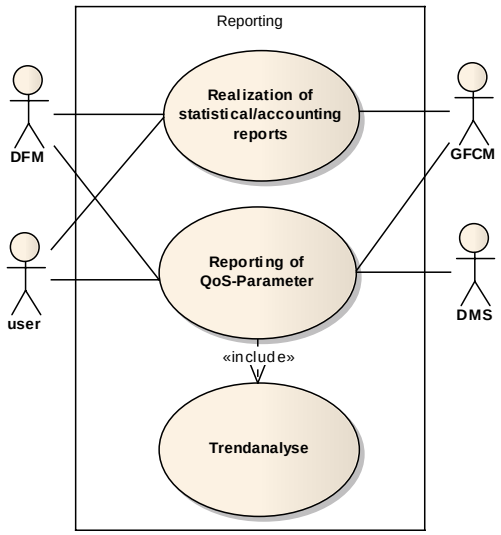


Abbildung 3.13.: Anwendungsfälle bezüglich Reporting

Anwendungsfall R02

QoS-Parameter (R02)

Kurzbeschreibung: In diesem Anwendungsfall werden für jeden Dienst eine Reihe von QoS-Parametern bereitgestellt. Dadurch kann man Abweichung in der Qualität des Dienstes oder der Teildienste feststellen (z.B. der Dienst ist verfügbar, aber der Jitter überschreitet einen bestimmten Grenzwert.).

Initiierender Aktor: DFM, GFCM

Vorbedingungen: Alle Domänen besitzen ein Monitoringsystem und können QoS-Parameter für die Teildienste, die sie erbringen, bereitstellen.

R02-Primärszenario: QoS-Parameter werden bereitgestellt.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. Der GFCM fordert von allen DFM der am Dienst beteiligten Domänen die QoS-Parameter für die Teildienste an.
2. Die DFM fassen die QoS-Parameter für die Teildienste (ermittelt aus dem DMS) zusammen.
3. Die DFM überreichen die jeweiligen QoS-Parameter an den GFCM.
4. Der GFCM fasst die Daten zusammen.
5. Der GFCM stellt die QoS-Parameter für den Gesamtdienst bereit.

Nachbedingungen: QoS-Parameter sind bereitgestellt.

R02-Sekundärszenario I: Nicht alle Domänen können QoS-Parameter bereitstellen.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. Vorgehensweise wie im Primärszenario.
2. Der DFM meldet, dass für einen Bereich keine QoS-Parameter vorhanden sind.

Nachbedingungen: Die QoS-Parameter können für diesen Gesamtdienst nicht ermittelt werden.

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

- Im Fall der Hierarchie ist es der „Wurzel"-Service-Provider, der die einzige Verantwortung für die QoS der Dienste für seinen Kunden hat.
- Im Fall der Heterarchie sind alle an dem interorganisationaler Diensterbringung beteiligten Service-Provider für die Einhaltung der QoS verantwortlich.

Trendanalyse (R03)

Anwendungsfall R03

Kurzbeschreibung: Dieser Anwendungsfall (als Spezialfall von R02) betrachtet, auf Grund der bereits existierenden Daten (hauptsächlich QoS-Parameter), die Entwicklung bzgl. der Fehler in bestimmten Bereichen, z. B. bei bestimmten Dienste deren Wiederherstellungszeit.

Initiierender Aktor: DFM, GFCM

Vorbedingungen: Alle Domänen verfügen über ein Monitoring- und Speicherungssystem, um die Daten on demand zur Verfügung zu stellen.

R02-Primärszenario: Trendanalyse wird realisiert.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. Der GFCM fordert von allen DFMs der an Dienst beteiligten Domänen die QoS-Parameter für eine bestimmte Zeitspanne in der Vergangenheit an.
2. Jeder beteiligte DFM fasst die QoS-Parameter für den Teildienst (ermittelt aus dem DMS) für die bestimmte Zeitspanne zusammen.
3. Die DFM überreichen die pro Bereich zusammengefassten QoS-Parameter an den GFCM.
4. Der GFCM fasst die QoS-Parameter für eine bestimmte Zeitspanne zusammen.
5. Der GFCM stellt die Fehler-Trendanalyse für den Gesamtdienst bereit.

Nachbedingungen: Trendanalyse steht zur Verfügung.

R02-Sekundärszenario I: Nicht alle Domänen können QoS-Parameter bereitstellen.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. Vorgehensweise wie im Primärszenario.
2. Der DFM meldet, dass für einen Bereich keine QoS-Parameter für die bestimmte Zeitspanne vorhanden sind.

Nachbedingungen: Die Trendanalyse kann nicht oder nur teilweise realisiert werden.

Abbildung 3.13 stellt die Zusammenfassung der Anwendungsfälle bezüglich Reporting in Rahmen des Fehlermanagement dar.

Die Anforderungen an die ioFMA, die sich aus den Anwendungsfällen bezüglich Reporting ergeben, können folgendermaßen zusammengefasst werden:

Anwendungsfälle R01-R03: Anforderungen an das Informationsmodell

Informationsmodell

Gemeinsames Informationsformat Um Reporting organisationsübergreifend zu realisieren ist ein einheitliches Format für Accounting- und Metrikdaten gefordert. Wenn es keine vergleichbaren Größen für Accounting- und Metrikdaten in den unterschiedlichen beteiligten Domänen gibt, können diese nicht zusammengefasst werden , eine providerübergreifende Auswertung ist dann nicht möglich.

Konvertierungsmethoden Für die Domänen, die sich an das gemeinsame Datenformat nicht oder nur mit größerem organisatorischen Aufwand halten können, sollten Konvertierungsmethoden der lokalen Accounting- und Metrikdaten in den gemeinsamen Informationsformat definiert werden, um eine einheitliche interorganisationale Übersicht zu bekommen.

Unterstützung von Standardmetriken Das ioFMA muss dienstspezifische Standardmetriken unterstützen. Ein solcher Standard wäre z. B. bei Netzdiensten der Durchsatz und die Auslastung oder IP Leistungsmetriken (engl. IP Performance Metrics (IPPM)) wie z.B. Delay, One-way Delay (OWD), (Jitter), Paketverlust und andere.

Aggregationsfunktionen In interorganisationalen Umgebungen müssen auf Metriken unterschiedlichen Aggregationsmechanismen angewendet werden um domänenübergreifende Kennzahldaten bereitzustellen. Das Informationsmodell muss solche Aggregationsfunktionen unterstützen.

Funktionsmodell

Anwendungsfälle R01-R03: Anforderungen an das Funktionsmodell

Visualisierungsmöglichkeit Visualisierung durch Metrikgraphen muss unterstützt werden. Die Übersichtlichkeit und Benutzerfreundlichkeit wird dabei steigen.

Anwendungsfälle R01-R03: Anforderungen an das Kommunikationsmodell

Kommunikationsmechanismen Für das Reporting ist ein hybrider Kommunikationsmechanismus notwendig. Durch Push-Mechanismen werden die Accounting- und Metrikdaten regelmäßig gesammelt, und per Pull-Mechanismus explizit für einen bestimmten Zeitraum, Domäne, Dienst usw. abgefragt.

Kommunikationsmodell

Anwendungsfälle R01-R03: Nicht-funktionale Anforderungen

Zugriffskontrolle Nur autorisierte und authentifizierte Rollen können auf die Accounting- bzw. Metrikdaten zugreifen und diese explizit anfordern.

NF Anforderungen

Datenintegrität Die Accounting- und Metrikdaten sollten vor Verlust und gegen vorsätzliche Veränderung geschützt sein.

Performanz Der Zugriff auf den Accounting- und Metrikdaten kann von vielen Domänen aus gleichzeitig erfolgen, dabei sind definierte Performanz-Ziele einzuhalten (z. B. Dauer für Datenbereitstellung).

Dokumentation Die Reports müssen entsprechen dokumentiert sein, um die wichtigsten Informationen über das Dargestellte zu geben.

3.2.2.5. Anwendungsfälle bezüglich Fehlalarmen

In vielen Fällen handelt es sich bei den Fehlermeldungen um Fehlalarme (False-Positives), also Fehlermeldungen, denen gar keine Fehler zugrunde liegen. Um die Suche nach Fehlern zu vermeiden, die gar nicht bestehen, sollte eine Managementarchitektur die Eigenschaft haben, False-Positive-Fehlermeldungen zu erkennen und auszufiltern.

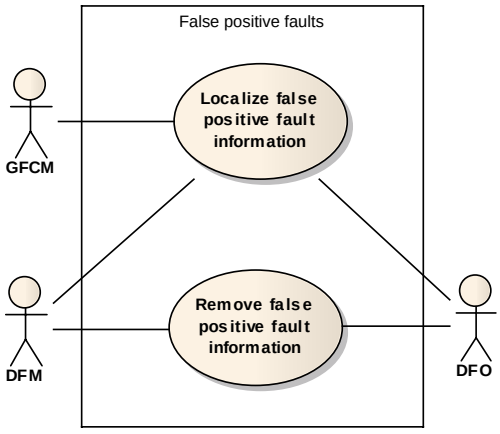


Abbildung 3.14.: Anwendungsfälle bezüglich False-Positive-Fehlermeldungen

Anwendungsfall F01

Lokalisierung von False Positives (F01)

Kurzbeschreibung: Dieser Anwendungsfall adressiert die Lokalisierung von False-Positive-Fehlermeldungen.

Initiierender Aktor: DFM, GFCM

Vorbedingungen: Ein Fehler wurde vom DMS oder Nutzer gemeldet.

F01-Primärszenario: Erkennen einer der False-Positive-Fehlermeldung

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. Der GFCM fordert den zuständigen DFM auf, den Fehler zu überprüfen.
2. Der DFM fordert Informationen vom DFO.
3. Der DFO meldet einen Fehlalarm.
4. DFM benachrichtigt den GFCM.

Nachbedingungen: False-Positive-Fehlermeldung gefunden.

Anwendungsfall F02

Markierung von False Positives (F02)

Kurzbeschreibung: Dieser Anwendungsfall behandelt die Markierung von Fehlalarmen.

Initiierender Aktor: DFO

Vorbedingungen: Ein Fehler ist bereits als False-Positive erkannt worden.

F02-Primärszenario: False-Positive-Fehlermeldung wird markiert.

Ablauf: Die folgenden Schritte werden in diesem Szenario durchlaufen:

1. Der DFO erkennt ein Fehlalarm.
2. Der DFO kennzeichnet den Fehler als False Positive.
3. Der DFO benachrichtigt den DFM.

Nachbedingungen: False Positive markiert.

Die Anwendungsfälle bezüglich False-Positive-Fehlermeldungen sind in Abbildung 3.14 zusammengefasst.

Nach einer ausführlichen Betrachtung dieses Anwendungsfalles werden Anforderungen an die ioFMA, die spezifisch für die Anwendungsfälle zu False Positive von Belang sind, zusammengefasst.

Anwendungsfälle F01-F02: Anforderungen an das Informationsmodell

Gemeinsames Informationsformat Auch im Fall der Anwendungsfälle zu False Positive wird ein einheitliches Datenformat benötigt; die False-Positives sollten auch möglichst schnell zu erkennen sein aufgrund von ähnlichen Daten, die schon im Ticketsystem sind. *Informationsmodell*

Konvertierungsmethoden Für die Domänen, die sich an das gemeinsame Datenformat gleich oder nur mit größerem organisatorischen Aufwand halten können, sollten Konvertierungsmethoden von den lokalen Accounting- und Metrikdaten in das gemeinsame Datenformat definiert werden, um providerübergreifende Übersicht zu bekommen.

Schnittstellendefinition Da bei der Lokalisierung von Fehlalarmen eine intensive Interdomänen-Kommunikation gefordert ist, wird eine angemessene Schnittstellendefinition benötigt.

Anwendungsfälle F01-F02: Anforderungen an das Funktionsmodell *Funktionsmodell*

Visualisierungsmöglichkeit Eine Visualisierung durch Metrikgraphen muss unterstützt werden. Die Übersichtlichkeit und Benutzerfreundlichkeit wird dabei steigen.

Datenänderung Insbesondere für den Anwendungsfall F02 ist wichtig, Fehlermeldungen ggf. als False Positive zu markieren. Dafür wird die Möglichkeit der Änderung von Fehlerdaten benötigt.

Anwendungsfälle F01-F02: Nicht-funktionale Anforderungen *NF Anforderungen*

Zugriffskontrolle Eine notwendige Voraussetzung ist eine klare Definition der Zugriffsrechten der unterschiedlichen Rollen; insbesondere die Festlegung, wer in welcher Domäne den Status der Fehler zu False Positive ändern darf.

Skalierbarkeit Die Lokalisierung, aber auch die Beseitigung von False Positives sollte sowohl für eine oder zwei Domänen als auch für größere Hierarchien und Heterarchien von Service Providern durchführbar sein.

Datenintegrität Die Daten sollten vor Verlust oder vorsätzlicher Veränderung gesichert sein.

Datenaktualität Zur Lokalisierung von False Positives ist es äußerst wichtig, immer auf aktuelle Daten Zugriff zu haben.

Gemeinsame Informationsdatenbank Wie schon im Fall der Fehlerlokalisierung wird für die Lokalisierung von False Positives ein gemeinsamer Datenbestand benötigt, auf den allen oder einigen Domänen Zugriff gewährt wird. In dieser Datenbank werden auch Fehlermeldungen ggf. als False Positives gekennzeichnet, so dass diese zu einem späteren Zeitpunkt als solche erkannt werden können.

Anwendungsfall		Szenarien		
		MWN	LCG	allg
Fehlerlokalisierung	L01: in der eigenen Domäne	✓	✓	✓
	L02: interorganisational		✓	✓
	L03: in einer andern Domäne		✓	✓
Statusanzeige	P01: der Fehlerbearbeitung	✓	✓	✓
	P02: der Wartungsarbeit	✓	✓	✓
Überwachung	M01: Domänenüberwachung	✓	✓	✓
	M02: Gesamtstatusüberwachung		✓	✓
	M03: eines interorg. Dienstes	✓	✓	✓
Reporting	R01: Statistiken, Accountingdaten	✓	✓	✓
	R02: QoS-Parameter	✓	✓	✓
	R03: Trendanalyse	✓	✓	✓
False-Positives	F01: Lokalisierung		✓	✓
	F02: Markierung		✓	✓

Tabelle 3.16.: Referenz der Anwendungsfälle in den Szenarien

Nachdem die Anwendungsfälle und die daran beteiligten Akteure beschrieben wurden, wird die Abdeckung der Anwendungsfälle in den Szenarien (vgl. Tabelle 3.16) und welche Akteure daran beteiligt sind (vgl. Tabelle 3.17) in den entsprechende Tabellen zusammengefasst.

3.2.3. Kategorisierte Zusammenfassung der Anforderungen

In den vorherigen Abschnitten dieses Kapitels wurden die Akteure und die Anwendungsfälle beschrieben. Daraus wurden für jeden Anwendungsfall spezifische Anforderungen abgeleitet. Diese werden in Tabelle 3.18 zusammengefasst. Aufgrund der hier gelisteten Anforderungen werden in diesem Abschnitt die allgemeinen Anforderungen an die interorganisationale Fehlermanagementarchitektur (ioFMA) zusammengefasst. Aufgrund der Anzahl an Anwendungsfällen, für die die jeweiligen Anforderungen wichtig sind, könnte man eine Gewichtung der Anforderungen vornehmen. Im Folgenden wird dennoch keine Gewichtung der Anforderungen vorgenommen, weil letztendlich alle Anwendungsfälle von der ioFMA unterstützt werden müssen. Daher müssen auch alle Anforderungen für jeden einzelnen Anwendungsfall erfüllt sein; selbst mehrfach genannte Anforderungen sind deshalb nicht als wichtiger einzustufen als die anderen.

Anwendungsfall		Akteure				
		user	DMS	DFO	DFM	GFCM
Fehlerlokalisierung	L01: in der eigenen Domäne	✓	✓	✓	✓	
	L02: interorganisational			✓	✓	✓
	L03: in einer andern Domäne			✓	✓	✓
Statusanzeige	P01: der Fehlerbearbeitung			✓	✓	✓
	P02: der Wartungsarbeit			✓	✓	✓
Überwachung	M01: Domänenüberwachung		✓		✓	✓
	M02: Gesamtstatusüberwachung		✓		✓	
	M03: eines interorg. Dienstes		✓		✓	✓
Reporting	R01: Statistiken, Accountingdaten		✓		✓	✓
	R02: QoS-Parameter		✓		✓	✓
	R03: Trendanalyse		✓		✓	✓
False-Positives	F01: Lokalisierung			✓	✓	✓
	F02: Markierung			✓	✓	

Tabelle 3.17.: Beteiligung der Akteure an den Anwendungsfällen

3.2.3.1. Anforderungen an das Funktionsmodell

Durch die Beschreibung der Szenarien und die darauf folgende Ableitung von Anwendungsfällen wurden die Grundlagen des Funktionsmodells geschaffen. Das Funktionsmodell muss die Anwendungsfälle unterstützen, d.h. diese legen die Funktionalitäten der ioFMA fest, und damit wird der Großteil der funktionalen Anforderungen an eine interorganisationale Fehlermanagementarchitektur (ioFMA) adressiert. In der objektorientierten Softwareentwicklung werden die beschriebenen Anwendungsfälle [BrDu10] als funktionale Anforderungen betrachtet. Die Anforderungen sind:

- **FM-L01** Die Fehlerlokalisierung in der eigenen Domäne
- **FM-L02** Fehlerlokalisierung in irgendeiner Domäne
- **FM-L03** Fehlerlokalisierung in einer bestimmten anderen Domäne
- **FM-P01** Status der Fehlerbearbeitung
- **FM-P02** Status der Wartungsarbeit
- **FM-M01** Intraorganisationales Monitoring
- **FM-M02** Interorganisationales Monitoring (bzw. Gesamtüberwachung)

Anforderungen		Anwendungsfälle				
		L01-L03	P01-P02	M01-M03	R01-R03	F01-F02
IM	Gemeinsames Datenformat	✓	✓	✓	✓	✓
	Konvertierungsmethoden	✓		✓	✓	✓
	Schnittstellendefinition	✓		✓		✓
	Fehlerlebenszyklus	✓	✓	✓		
	Standard-Metriken			✓	✓	
	Aggregationsfunktionen				✓	
OM	io-Form der Diensterbringung	✓	✓	✓		
	Rollendefinition	✓	✓	✓		
FM	Visualisierungsmöglichkeit	✓	✓	✓	✓	✓
	Statusänderung		✓			
	Datenänderung					✓
KM	Kommunikationsmechanismen	✓	✓	✓	✓	
NFA	Zugriffskontrolle	✓	✓	✓	✓	✓
	Datenintegrität	✓			✓	✓
	Skalierbarkeit	✓		✓		✓
	Performanz	✓		✓	✓	
	Gemeinsame Informationsdatenbank	✓			✓	
	Automatisierung		✓	✓		
	Datenaktualität		✓	✓		✓
	Dokumentation				✓	

Tabelle 3.18.: Spezielle Anforderungen bei den Anwendungsfällen

- **FM-M03** Überwachung eines gesamterbrachten Dienstes
- **FM-R01** Bereitstellung von Statistiken und Accountingdaten
- **FM-R02** Bereitstellung von QoS-Parametern
- **FM-R03** Bereitstellung von Trendanalysen
- **FM-F01** Lokalisierung von False-Positives Fehlermeldungen
- **FM-F02** Beseitigung von False-Positives Fehlermeldungen

Zusätzlich müssen noch folgende Anforderungen erfüllt werden:

FM-01 Visualisierungsmöglichkeit Die Visualisierungsmöglichkeit muss gegeben werden. Wie schon bei den speziellen Anforderungen, die bei den jeweiligen Anwendungsfall zusammengefasst wurden, kann man feststellen, dass die graphische Darstellung in

allen Fällen benötigt wird, um die Übersichtlichkeit der ermittelten Daten und die Benutzerfreundlichkeit zu steigern.

FM-02 Datenänderung Änderung von Daten unter bestimmten Umständen. Wie oben beschrieben muss in bestimmten Fällen die Änderung von Daten zugelassen und realisierbar sein. Nachdem ein Fehler gemeldet wurde, je nach dem, in welchem Bearbeitungsstadium dieser sich befindet, muss z.B. der Status veränderbar sein. Ebenfalls muss beim Erkennen eines False Positive der Fehler als solcher gekennzeichnet werden.

3.2.3.2. Anforderungen an das Informationsmodell

IM-01 Gemeinsames Datenformat Zusammengefasst zeigen die speziellen Anforderungen, dass für die Erfüllung jedes der Anwendungsfälle ein gemeinsames Datenformat benötigt wird. Es kann auch nur eine bestimmte Untermenge der Eigenschaften (Attribute) gemeinsam sein, so dass der Datenaustausch zwischen den Domänen und interorganisational funktionieren kann.

IM-02 Konvertierungsmethoden Als Übergangslösung oder auch parallel zu der ersten Anforderung wird auf einen bestimmten Konvertierungsmechanismus zurückgegriffen, um die lokalen Datenbestände einzelner Domänen auf interorganisationale Datenformate abbilden zu können. Diese wird in fast allen Anwendungsfällen als notwendige Anforderung betrachtet.

IM-03 Schnittstellendefinition Beim Übergang zwischen den Domänen bringt eine klar definierte Schnittstelle große Vorteile in der Interdomänen-Kommunikation, also beim Datentransfer von einer Domäne in die andere.

IM-04 Fehlerlebenszyklus Der Fehlerlebenszyklus mit Entdecken, Eingrenzen und Beheben von Fehlern muss durch die ioFMA unterstützt werden. Mit anderen Worten, die ioFMA muss für alle drei Phasen des Lebenszyklus einsetzbar sein.

IM-05 Standard-Metriken Die Notwendigkeit der Nutzung von Standard-Metriken ist insbesondere beim Monitoring und Reporting gegeben, wo das Ziel eine gewisse vereinheitlichte Datendarstellung ist. Wie bereits erwähnt gibt es Standard-Metriken, die unterstützt werden können, z. B. der Durchsatz und die Auslastung oder die IP Leistungsmetriken.

IM-06 Aggregationsfunktionen Eine ioFMA kann unmöglich auf eine interorganisationale Datendarstellung verzichten, und dabei ist die Aggregationsfunktion von lokalen Domänendaten auf eine providerübergreifende Sicht für die meisten Metriken ein wichtiger Faktor. Die Aggregationsfunktionen können z. B. Topologie-, oder Zeitaggregationen sein.

3.2.3.3. Anforderungen an das Organisationsmodell

OM-01 Formen der interorganisationale Diensterbringung Wie schon im Abschnitt 2.1.3 beschrieben, betrachtet man bei der Modellierung der ioFMA zwei organisatorische Grundformen der Diensterbringung (hierarchische und heterarchische). Da diese einen starken Einfluss auf die interorganisationalen Workflows haben, muss eine ioFMA mit Anspruch auf Allgemeinheit beide von ihnen sowie Mischformen unterstützen.

OM-02 Rollendefinition Ein Rollenmodell in der ioFMA ist notwendig sowohl für die Durchführung unterschiedlicher Aktionen als auch für die Einteilung der Verantwortlichkeiten.

3.2.3.4. Anforderungen an das Kommunikationsmodell

KM-01 Kommunikationsmechanismen In fast allen Anwendungsfällen wird nach Bedarf die Unterstützung von Push- als auch der Pull-Mechanismen benötigt.

Es werden noch folgende zwei Anforderungen abgeleitet:

KM-02 Interdomänen-Kommunikation Die Freiheit jedes Service Providers und die teilweise „Konkurrenz" führen dazu, dass nur die absolut notwendigen Informationen zwischen den Netzen ausgetauscht werden und das Verhalten im Voraus aufgrund des mangelnden Wissens nur sehr bedingt vorhergesagt werden kann.

KM-03 Kommunikationsprotokolle Der Austausch von Nachrichten in interorganisationalen Umgebungen erfordert ein Zusammenspiel verschiedener Protokolle (aus den unterschiedlichen Domänen), die unterschiedliche Aufgaben übernehmen. Damit ergibt sich eine Komplexität, die durch die Organisation der Kommunikationsprotokolle in Schichten geregelt wird. Im Rahmen einer solchen Architektur gehört jedes Protokoll einer bestimmten Schicht an und ist für die Erledigung von speziellen Aufgaben zuständig (z. B. innerhalb der Schicht 2 des ISO-OSI-Referenzmodells [OSI 94] werden die Daten auf Vollständigkeit geprüft). Protokolle höherer Schichten verwenden Dienste von Protokollen tieferer Schichten (Schicht 7 verlässt sich z. B. darauf, dass die Daten vollständig angekommen sind). Zusammen bilden die so strukturierten Protokolle einen Protokollstapel.

3.2.3.5. Nicht-funktionale Anforderungen

Nachdem die vorherigen Kategorien von Anforderungen sich auf den Funktionalitäten der ioFMA bezogen, werden hier diejenigen Anforderungen, die sich auf die Verwendbarkeit, Verfügbarkeit, Leistungsfähigkeit und Wartbarkeit der Architektur beziehen, zusammengefasst.

NF-01 Zugriffskontrolle Eine geregelte Zugriffskontrolle auf die ioFMA ist eine der Voraussetzungen für die Realisierung jeder der Funktionalitäten.

NF-02 Datenintegrität Gegen den Datenverlust und vorsätzliche Änderungen müssen Mechanismen vorgesehen werden. Insbesondere wichtig ist diese Anforderung für die Fehlerlokalisierung, das Reporting und False Positives Funktionalitäten.

NF-03 Datenaktualität Regelmäßige (in möglichst kurzen Zeitabständen) sollen Updates die Datenaktualität gewährleisten. Der Zugriff auf veraltete Daten kann im Fall des Monitoring, der Statusüberwachung und False-Positives negative Auswirkungen auf die Erbringung des Dienstes haben.

NF-04 Skalierbarkeit Insbesondere für Fehlerlokalisierung, Monitoring und False Positives muss eine ioFMA-Lösung skalierbar sein. Sie sollte für 2-3 Domänen als auch für eine sehr große Anzahl an Domänen einsetzbar sein. Ebenfalls muss diese skalieren für die hierarchische als auch für die heterarchische Form der interorganisationalen Diensterbringung.

NF-05 Performanz Unabhängig von der Anzahl an Nutzern oder Domänen, die gleichzeitiger Zugriff auf dieioFMA haben, sollte die Performanz stabil sein. Bei Lokalisierung, Monitoring und Reporting ist die Performanz ein äußerst wichtiger Faktor.

NF-06 Automatisierung Ein automatisiertes Vorgehen sollte insbesondere beim Monitoring und der Statusüberwachung, wie schon bei den entsprechenden Anwendungsfällen beschrieben, unterstützt werden.

NF-07 Gemeinsame Informationsdatenbank Ein gemeinsamer Datenbestand wird insbesondere für die Fehler- bzw. False-Positives-Lokalisierung benötigt. Dies ist unter normalen Umständen eine Datenbank, in der alle Fehlermeldungen zusammengefasst sind. Interorganisational ist es als Herausforderung anzusehen, da die unterschiedlichen Domänen nicht unbedingt freien Zugriff auf deren Datenbestand gewähren (vgl. Anforderung KM-02).

NF-08 Dokumentation Insbesondere beim Reporting ist die Notwendigkeit einer ausführliche Dokumentation gegeben. Allerdings sollten alle anderen Vorgänge und Sachverhalte bezüglich der ioFMA ebenfalls dokumentiert werden.

3.3. Zusammenfassung

In diesem Kapitel wurden anhand der Szenarien Anwendungsfälle und beteiligte Akteure definiert und daraufhin funktionale und nicht-funktionale Anforderungen abgeleitet. Dadurch konnte das umfangreiche Gebiet der in dieser Arbeit beschriebenen Aufgabenstellung abgedeckt werden. Die Tabellen 3.16 und 3.17 stellen die Abdeckung der Anwendungsfälle

Anforderungen	Laut der Anforderung
an IM	IM-01: Definition eines gemeinsamen Datenformats IM-02: Konvertierungsmethoden IM-03: Schnittstellendefinition IM-04: Unterstützung des Fehlerlebenszyklus IM-05: Unterstützung von Standardmetriken IM-06: Aggregationsfunktionen
an OM	OM-01: Form der interorganisationale Diensterbringung OM-02: Rollendefinition
an FM	FM-L01: Fehlerlokalisierung in der eigenen Domäne FM-L02: Interorganisationale Fehlerlokalisierung FM-L03: Fehlerlokalisierung in einer anderen Domäne FM-P01: Status der Fehlerbearbeitung FM-P02: Status von Wartungsarbeiten FM-M01: Intraorganisationales Monitoring FM-M02: Interorganisationales Monitoring (bzw. Gesamtüberwachung) FM-M03: Überwachung eines interorganisationalen Dienstes FM-R01: Bereitstellung von Statistiken und Accountingdaten FM-R02: Bereitstellung von QoS-Parametern FM-R03: Bereitstellung von Trendanalysen FM-F01: Lokalisierung von False Positives FM-F02: Markierung von False Positives FM-01: Visualisierungsmöglichkeit FM-02: Datenänderung
an KM	KM-01: Kommunikationsmechanismen KM-02: Interdomänen-Kommunikation KM-03: Kommunikationsprotokolle
nicht funktionale	NF-01: Zugriffskontrolle NF-02: Datenintegrität NF-03: Datenaktualität NF-04: Skalierbarkeit NF-05: Performanz NF-06: Automatisierung NF-07: Gemeinsame Informationsdatenbank NF-08: Dokumentation

Tabelle 3.19.: Liste der allgemeinen Anforderungen

Anforderungen		Phasen des Lebenszyklus eines Fehlers			
		Entdecken	Eingrenzen	Beheben	Vorhersage
an IM	IM-01		✓		✓
	IM-02	✓	✓		✓
	IM-03	✓	✓	✓	
	IM-04	✓	✓	✓	✓
	IM-05	✓			✓
	IM-06	✓	✓		✓
an OM	OM-01	✓	✓	✓	✓
	OM-02	✓	✓	✓	✓
an FM	FM-L01	✓	✓		
	FM-L02	✓			✓
	FM-L03	✓	✓		
	FM-P01		✓	✓	
	FM-P02			✓	
	FM-M01	✓	✓	✓	✓
	FM-M02	✓	✓	✓	✓
	FM-M03	✓	✓	✓	✓
	FM-R01	✓			✓
	FM-R02	✓			✓
	FM-R03				✓
	FM-F01		✓		✓
	FM-F02			✓	✓
	FM-01	✓	✓		✓
	FM-02		✓	✓	✓
an KM	KM-01	✓	✓	✓	✓
	KM-02	✓	✓	✓	✓
	KM-03	✓	✓	✓	✓
NFA	NF-01		✓	✓	
	NF-02		✓	✓	
	NF-03	✓			✓
	NF-04	✓	✓	✓	✓
	NF-05		✓	✓	
	NF-06	✓	✓	✓	
	NF-07		✓	✓	✓
	NF-08		✓	✓	

Tabelle 3.20.: Abdeckung der Phasen des Lebenszyklus eines Fehlers durch die Anforderungen

in den Szenarien dar und welche Akteure daran beteiligt sind. Die hier resultierenden Anforderungen (siehe Tabelle 3.19) fließen als Input für den Entwurf der Managementarchitektur in Kapitel 5 ein.

Abbildung 3.20 fasst die für jede Phase des Fehlerlebenszyklus relevanten Anforderungen zusammen (diese sind durch die grau hinterlegten Felder gekennzeichnet). Zu bemerken ist daher, dass die Anforderungen an das Organisations- bzw. Kommunikationsmodell für alle Phasen von Bedeutung sind. Hiermit sind diese viel stärker repräsentiert als die Anforderungen an das Informations- bzw. Funktionsmodell.

4 Status Quo

Inhalt des Kapitels

In diesem Kapitel werden bestehende Ansätze, die für das interorganisationale Fehlermanagement relevant sind, untersucht. Nach einer Beschreibung erfolgt eine Evaluierung der Ansätze bezüglich den in Abschnitt 3.2 aufgestellten Anforderungen.

Dieses Kapitel ist folgendermaßen strukturiert: Als erstes werden in Abschnitt 4.1, Ansätze aus dem interorganisationalen Umfeld analysiert. Es folgt dann in Abschnitt 4.2 eine Einordnung des Fehlermanagements in das IT-Service-Management (ITSM). Abschnitt 4.3 stellt Ansätze aus dem Bereich Fehlermanagement dar, die auch für das interorganisationale Fehlermanagement relevant sind. Ansätze im Bereich Dienstkomposition, ein wichtiger Bestandteil des interorganisationalen Managements, werden in Abschnitt 4.4 ausgeführt. Das Kapitel wird mit einer zusammenfassenden Bewertung der Ansätze abgeschlossen.

4.1. Arbeiten zum interorganisationalen IT-Service-Management

In diesem Abschnitt werden Ansätze aus dem interorganisationalen IT-Service-Management betrachtet, die stark in Verbindung mit dem Fehlermanagement als Fokus dieser Arbeit stehen. In Abschnitt 4.1.1 wird die interorganisationale Dienstmonitoring-Plattform perfSONAR beschrieben, gefolgt in Abschitt 4.1.2 vom E2E Monitoring System (E2EMon), einem Werkzeug zum Monitoring von Ende-zu-Ende-Links. Der abschließende Abschnitt 4.1.3 stellt einige Forschungsansätze im Bereich des interorganisationale Fehlermanagements vor.

4.1.1. perfSONAR

Zur QoS- und SLA-Überwachung sowie zur Vereinfachung der domänenübergreifenden Fehlersuche für Dienste im europäischen Netzverbund Géant (siehe Abschnitt 3.1.3) wurde eine Komponentenarchitektur für ein föderiertes Multi-Domain Monitoring namens Performance focused Service Oriented Network monitoring ARchitecture (perfSONAR) entwickelt [HKM+ 08]. perfSONAR besteht aus drei Schichten: Measurement Point Layer, Service Layer und User Interface Layer. Als gemeinsame Basis definiert perfSONAR [per 08] übergreifende Metriken und Messverfahren zum Performance Monitoring.

4.1.1.1. Messungen in perfSONAR

Für Messungen in Rahmen von perfSONAR kommt hauptsächlich das von der Internet Engineering Task Force (IETF) standardisierte Rahmenwerk IP Performance Metrics (IPPM) zum Einsatz. Dieses beinhaltet Definitionen von Metriken zur Messung der Internet-Performance [PAMM 98]. Einige der in perfSONAR verwendeten und von der IPPM-Arbeitsgruppe definierten Metriken werden im folgenden kurz zusammengefasst.

Unter dem One-way Delay (OWD) [RFC 2679] versteht man diejenige Zeitverzögerung, die einem Paket auf dem Weg von einem Quell- an einen Zielrechner widerfährt. Dabei gilt als (optimaler) Startzeitpunkt der Messung derjenige Zeitpunkt, zu dem der Quellrechner das erste Bit des Pakets überträgt, und als Endzeitpunkt derjenige, zu dem der Zielrechner das

letzte Bit des Pakets empfängt. Der OWD in Bezug auf ein einzelnes Paket ergibt sich als Differenz von Start- und Endzeitpunkt.

One-way Packet Loss (OWPL) [RFC 2680] gibt an, ob ein bestimmtes, von einem Quell- an einen Zielrechner gesendetes Paket dort tatsächlich angekommen oder ob es auf dem Weg zu diesem verloren gegangen ist. Wendet man diese Metrik auf eine Menge nacheinander versendeter Pakete an, lässt sich schließlich der Anteil der auf dem Weg von der Quelle zum Ziel verlorengegangenen Pakete bestimmen.

IP Packet Delay Variation (IPDV) oder One-way Delay Variation (OWDV) [RFC 3393] bezieht sich jeweils auf zwei Pakete aus einer Menge von sequentiell versendeten Paketen. IPDV, bekannt auch als Jitter, ist dabei definiert als die Differenz der Verzögerungen (OWD), die den beiden Paketen auf dem Weg vom Quell- zum Zielrechner widerfahren.

Utilization (Auslastung) bezeichnet das tatsächlich auf einem Pfad auftretende Datenvolumen. Hierbei wird die übertragene Datenmenge pro Zeit zwischen zwei Endpunkten ermittelt und über einen Zeitraum dargestellt.

Neben diesen Metriken werden in perfSONAR auch weitere Netz-Kennzahlen ermittelt (beispielsweise passive Performance Daten, Interface Errors, Interface Drops) [HLMS 06a].

In Abhängigkeit von der Metrik erfolgt die Erfassung der Messungen über drei unterschiedliche Systeme: HADES, BWCTL und SNMP-Polling. Die Metriken OWD, OWPL und IPDV werden mit dem im DFN-Labor entwickelten Messsystem Hades Active Delay Evaluation System (HADES) [HADES] ermittelt. Zur Ermittlung der verfügbaren Bandbreite (Available Bandwidth, AB) wird Band Width Controller (BWCTL) [BWCTL] benutzt, ein Wrapper für iperf, entwickelt von Internet2.

4.1.1.2. Die perfSONAR Architektur

In diesem Abschnitt wird eine Übersicht über die dreischichtige perfSONAR-Architektur gegeben. Abbildung 4.1 stellt diese graphisch dar.

Measurement Point Layer

Die Measurement Point Layer ist die unterste Schicht der Architektur. In diesen werden Messungen der im vorherigen Abschnitt vorgestellten Metriken durchgeführt, die entweder direkt über Messpunkte Measurement Point (MP), ein perfSONAR-Interface, zugreifbar sind oder in nicht direkt zugreifbaren Datensammlungen für Messarchive Measurement Archive (MA) gespeichert werden, die über Dienste der Service Layer (s. u.) nutzbar sind. Vom Rahmenwerk werden keine festen Vorgaben gemacht, welche Arten von Messpunkten installiert sein müssen, sondern die Auswahl bleibt den einzelnen Domänen überlassen.

Service Layer

Die Service Layer dient dem Management von Messungen innerhalb und zwischen Netzwerkdomänen. Der Name dieser Schicht bezieht sich auf deren Implementierung als Web Services, wodurch eine logische Aufteilung von unterschiedlichen Funktionen erreicht wird.

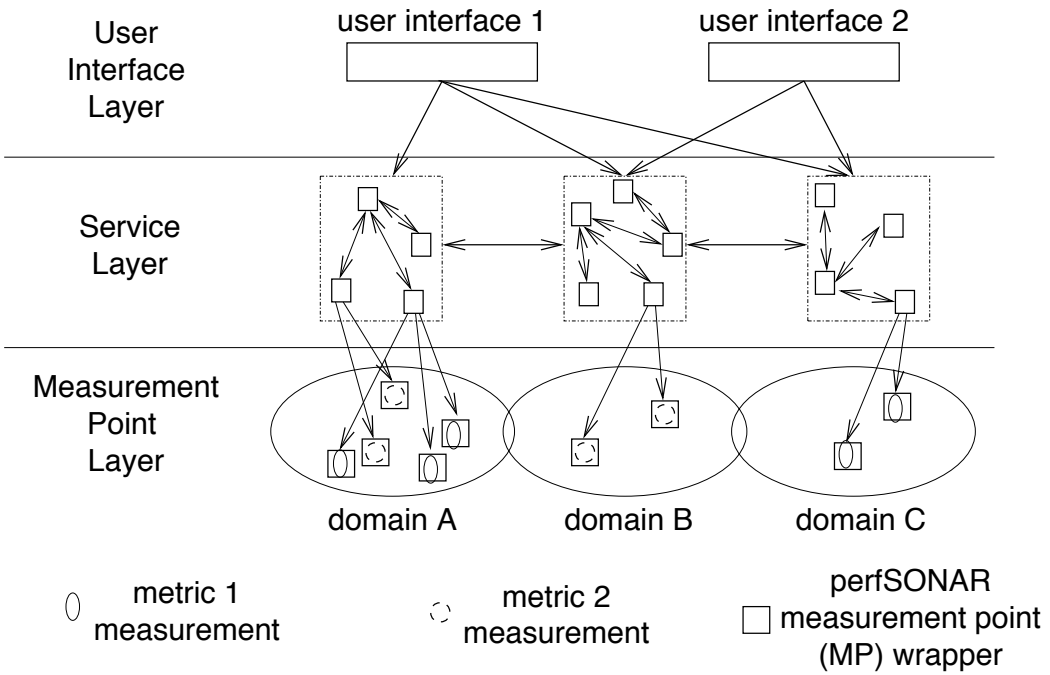


Abbildung 4.1.: Die Multi-Domain-Monitoring-Architektur perfSONAR [HBB+ 05]

Die Aufgabe eines Dienstes kann beispielsweise die Authentifizierung und Autorisierung von Nutzern, die Verwaltung von anderen Diensten in einem Dienstverzeichnis, der Schutz von Ressourcen zur Überlastvermeidung oder die Archivierung von Messdaten sein. Die Kommunikation der Dienste erfolgt über SOAP und verwendet dabei ein gemeinsames Protokoll der Network Measurement Working Group (NMWG) [NMWG] des Open Grid Forum (OGF), in dessen Standardisierung die Anforderungen von perfSONAR berücksichtigt werden.

Die wichtigsten Dienste der perfSONAR Service Layer sind:

Telnet/SSH MP – ein Werkzeug, um mittels der Protokolle Telnet oder SSH auf Router zuzugreifen. Dieser Messpunkt ermöglicht derzeit den Zugriff auf Cisco und Juniper-Router und übersetzt herstellerunabhängige Befehle in deren proprietäre Kommandos. Der MP ermöglicht damit (zusammen mit dem dazu entwickelten Visualisierungstool) eine vereinfachte Fehlersuche über Domänengrenzen hinweg.

HADES MA – eine perfSONAR-Schnittstelle zu einer Datenbank, in der HADES-Messungen (OWD, IPDV und OWPL) gesammelt und verwaltet werden.

BWCTL MP – ein Dienst zum Testen der verfügbaren Bandbreite. Dabei wird zwischen zwei MPs so viel Verkehr erzeugt, wie man erfolgreich übertragen kann.

Round Robin Database Measurement Archive (RRD MA) – eine perfSONAR-Schnittstelle für den Zugriff auf archivierte Messdaten. Hierbei geht es vor allem um Auslastungsdaten, aber auch um Paketverwerfungen an Routerinterfaces. Alternativ zum RRD MA gibt es das Structured Query Language Measurement Archive (SQL MA), das einen Zugriff auf SQL-Datenbanken wie z. B MySQL oder PostgreSQL ermöglicht.

Lookup Service – ein Verzeichnisdienst für perfSONAR-Services, bei dem sich Services registrieren und darüber gefunden werden können.

Authentication and Authorization Service (AA) – über diesen Dienst wird von den einzelnen Domänen festgelegt, wie Installationen von Diensten von anderen Domänen verwendet werden können. Der Dienst erlaubt dabei die Festlegung einer Zuordnung zwischen Attributen eines Nutzers und dessen Rechten.

User Interface Layer

Auf der obersten Schicht (User Interface Layer) befinden sich zielgruppenspezifische Visualisierungswerkzeuge [HJKM 06], die die gemessenen Kennzahlen in graphischer Form darstellen und deren Analyse ermöglichen. Als Zielgruppen werden dabei Mitarbeiter von Network Operation Centers (NOC), Projektgruppen, einzelne Wissenschaftler und administrative Mitarbeiter unterschieden.

Das Tool perfSONAR User Interface (pS UI) bietet einen direkten Zugriff auf perfSONAR-Dienste und stellt diese gruppiert nach Metriken dar. Das Tool setzt Grundkenntnisse über perfSONAR voraus und ist für die Fehleruntersuchung konzipiert.

Das VisualperfSONAR-Tool läuft innerhalb einer Weboberfläche und ist auf die Analyse von Pfaden spezialisiert.

Das CNM-Tool stellt die Topologie von Netzen mittels hierarchischer Karten dar. Die Karten beinhalten sowohl Netzknoten (Router) und Verbindungen (Links) zwischen unterschiedlichen Knoten als auch Statusinformationen und Kennzahlen.

Parallel zu pS UI wurden zur Darstellung der gemessenen Metriken OWD, OWPL und IPDV am DFN-Labor geeignete Visualisierungen entwickelt [HADES]. Auch BWCTL Messungen können in die Datendarstellung integriert werden.

Der perfSONAR-Ansatz stellt eine Komponentenarchitektur für interorganisationales Monitoring dar. Deren dreischichtiger Aufbau ist für die vorliegende Arbeit sehr wichtig bei der plattformspezifischen Transformation auf die Systemsicht (siehe Abschnitt 6). Einige Dienste (z. B. RRD MA, HADES MA) können als Bestandteil der plattformspezifischen Entwicklung eingesetzt werden. Allerdings ist die Architektur aufgrund ihrer plattformspezifischen Ausrichtung auf Web-Services nicht allgemeingültig. Nichtsdestotrotz kann man Elemente dieser Architektur in der Entwicklung einer allgemeinen ioFMA wiederverwenden. Beispielsweise können die benutzten Metriken und deren Messungen in der vorliegenden Arbeit sowohl als Unterstützung für die Anforderung Standard-Metriken (IM-05) benutzt werden als auch für Fehlerlokalisierung (FM-L01, FM-L02) und interorganisationales Monitoring (FM-M01, FM-M02). Auf Grund dieser Messungen werden Statistiken (FM-R01),

QoS-Parameter (FM-R02) sowie Trendanalysen (FM-R03) bereitgestellt. Ein Kritikpunkt an dieser Architektur bezieht sich auf die Visualisierung: es gibt vier verschiedene, nicht integrierte Visualisierungstools um den Kunden/Spezialisten über den Zustand des Netzes zu berichten. Obwohl jedes dieser Visualisierungstools auf spezielle Zielgruppen ausgelegt ist, um jedem Nutzer entsprechend seiner Zielgruppe sein Tool anzubieten, ist der Nutzeraufwand immer noch zu hoch.

4.1.2. E2E-Monitoring Tool

Ebenfalls in Rahmen des Géant-Projektes wurde ein Monitoring-System für Netzverbindungen auf ISO/OSI Schicht 2 entwickelt, das die Autonomie der beteiligten Domänen berücksichtigt [HaYa 07, HaYa 08]. Jeder der an einem End-to-End (E2E)-Link beteiligten Domänen (NREN) werden Informationen bezüglich des Zustands dieser Verbindung über eine standardisierte Schnittstelle zur Verfügung gestellt. Informationen aus jeder dieser Domänen werden in regelmäßigen Zeitabständen über das Monitoring-System E2EMon abgefragt und daraus ein E2E-Wert bzgl. des Link-Zustandes aggregiert. E2EMon wird bei einer zentralen Stelle E2E Coordination Unit (E2ECU) betrieben. Deren Rolle ist die Überwachung und Koordination des Incident- und Problemmanagements. Bei E2EMon wird kein direkter Zugriff auf die Domäneninfrastruktur benötigt, so dass die Sicherheitsrichtlinien und die Autonomie aller Domänen bewahrt werden können. Vorteile des Tools sind: die automatische Synchronisation von Monitoring-Informationen aller Domänen durch gleichzeitige Abfrage des Ist-Zustands, ebenso wie die automatische Erkennung von Störungen sowie die Identifizierung derjenigen Domäne, in der das Problem aufgetreten ist.

Für diese Arbeit ist dieser Ansatz vielversprechend, da es ein Konzept für das interorganisationales Monitoring (FM-M02) bietet. Der Aufbau von E2EMon, das in regelmäßigen Abständen den Zustand eines Links ermittelt, passt in die plattformspezifische Entwicklung der ioFMA, auf Systemsicht. Allerdings handelt es sich hier um eine Fall-spezifische Lösung für den E2E-Dienst der Géant.

Fehlerlokalisierung in einer bestimmten ,,fremden" Domäne (FM-L03) wird von E2EMon unterstützt. Es bietet ebenfalls Schnittstellendefinitionen (IM-03), Aggregationsfunktionen (IM-06), Interdomänen-Kommunikation (KM-02) und Kommunikationsmechanismen (KM-01). Auch wenn sie nur für heterarchische Formen der Diensterbringung vorgesehen sind, können diese gut eingesetzt werden, wenn man sie geeignet verallgemeinert. Von den nichtfunktionalen Anforderungen aus Abschnitt 3.2.3.5 sind auch folgende erfüllt: Zugriffskontrolle (NF-01), Datenintegrität (NF-02), Datenaktualität (NF-03) und Automatisierung (NF-06).

4.1.3. Sonstige Forschungsansätze

Steider und Sethi untersuchen in [StSe 04b] bzw. in [StSe 07] die Problematik der Multidomänen-Diagnose. Dies wird anhand von Ende-zu-Ende-Dienstausfällen in hierarchisch gerouteten Netzen realisiert. Die Methode ist eine verbesserte Variante der unten beschriebenen Methode zur probabilistischen Fehlerentdeckung [StSe 04d, StSe 04c]. In der beschriebenen Methode wird der Rechenaufwand und das Systemwissen unter mehreren hierarchisch verteilten Managern verteilt. Jeder Manager realisiert eigenständig in seiner Domäne die Fehlerlokalisierung basierend auf dem Wissen der eigenen Domäne. Wenn sich Fehler außerhalb einer Domäne propagieren, kommunizieren die Domänen miteinander, um eine Konsenserklärung für den beobachteten Fehler zu finden. Dieser Ansatz kann auf Netze mit mehreren Routing-Levels angewandt werden, aber die Beschreibung und Implementierung zieht nur zwei Levels in Betracht. Als Wurzel dieser Hierarchie fungiert der Network Manager (NM), der die Aktivitäten der ihm untergeordneten Domain Manager (DMs) verteilt und sie koordiniert. Es wird ein sogenanntes Distributed-Fault-Propagation-Model für den NM und die DMs eingeführt und definiert.

Da die Aufgabenstellung in der vorliegenden Arbeit sehr ähnlich ist, kann für das Organisationsmodell (OM-01, OM-02) ein ähnlicher Rollenaufbau wie hier beschrieben vorgenommen werden. Allerdings ist die Einteilung dieses Ansatzes ziemlich grob, da aus dem Allgemeinszenario die Notwendigkeit mehrerer Rollen innerhalb einer Organisation resultiert (vgl. Abschnitt 3.2.1) Für das Funktionsmodell sind folgende Punkte relevant: Fehlerlokalisierung in eigener und einer bestimmten Domäne (FM-L01, FM-L03) durch die Konsenserklärung der beteiligten Partner und die Überwachung insgesamt erbrachter Dienste (FM-M03). Allerdings ist dieser Ansatz nur für hierarchische Netze ausgelegt und nicht für heterarchische. Deswegen ist ein Ausbau dieses Konzepts für den Allgemeinfall nötig.

Eine Architektur für Interdomänen-Diensterbringung mit Fokus auf Einhaltung von E2E-QoS ist in [PDPT 10] beschrieben. Das Netz ist nach diesem Ansatz in mehrere Plan-Levels (Netzschichten) eingeteilt. Die Architektur lehnt sich stark an die IPSphere- Empfehlungen [ipsh10] an. Da es sich hier nur um Service-Delivery im Allgemeinen handelt, sind nur die grundlegenden Konzepte des Architekturaufbaus bzgl. der Netzsichten, die betrachtet werden, relevant. Diese werden in der vorstehenden Arbeit auf Systemsicht beim Aufbau der Managementplatform eingesetzt.

Tang et al. untersuchen in [TASZ 07] das Ereignismonitoring in großen, großräumig verteilten Overlay-Netzen. Die von ihnen vorgeschlagene Monitoringinfrastruktur *MOON* dient zur Senkung der Monitoringverzögerung und der Kosten für die Ereigniskorrelation. *MOON* clustert und organisiert Overlay-Knoten effizient, so dass die Applikationen, die auf dem entsprechenden Overlay-Netz laufen, mit einer minimalen Verzögerung und minimalen Kosten überwacht werden. Zusätzlich führt im Bereich des interorganisationaler Überwachung Gonzales Pieto in [PrSt 09] ein Konzept zum Echtzeit-Monitoring für große (großräumig verteilte) Netze ein. Der Kern dieser Arbeit ist das Design und die Implementierung eines Protokolls für verteilte Überwachung großer Netze. Für die Anforderungen in der vorliegende

Anforderungen		Phasen des Lebenszyklus eines Fehlers			
		Entdecken	Eingrenzen	Beheben	Vorhersage
an IM	IM-01				
	IM-02				
	IM-03	[E2EMon]	[E2EMon]		
	IM-04				
	IM-05	[HADES]			[HADES]
	IM-06	[E2EMon]	[E2EMon]		
an OM	OM-01	[per 08, StSe 04b]	[per 08, StSe 04b]	[per 08]	[per 08]
	OM-02	[StSe 04b]	[StSe 04b]		
an FM	FM-L01	[HADES, StSe 04b]	[HADES, StSe 04b]		
	FM-L02	[HADES]			[HADES]
	FM-L03	[E2EMon, StSe 04b]	[E2EMon, StSe 04b]		
	FM-P01				
	FM-P02				
	FM-M01	[HADES]	[HADES]	[HADES]	
	FM-M02	[per 08, E2EMon, TASZ 07, PrSt 09]	[per 08, E2EMon, TASZ 07, PrSt 09]	[per 08, E2EMon]	[per 08]
	FM-M03	[E2EMon, StSe 04b, TASZ 07, PrSt 09]	[E2EMon, StSe 04b, TASZ 07, PrSt 09]	[E2EMon]	
	FM-R01	[HADES, BWCTL]			[HADES, BWCTL]
	FM-R02	[HADES, BWCTL]			[HADES, BWCTL]
	FM-R03				[HADES, BWCTL]
	FM-F01				
	FM-F02				
	FM-01	[per 08]	[per 08]		
	FM-02				
an KM	KM-01	[E2EMon]	[E2EMon]	[E2EMon]	
	KM-02	[E2EMon]	[E2EMon]	[E2EMon]	
	KM-03				
NFA	NF-01		[E2EMon]		
	NF-02		[E2EMon]		
	NF-03	[E2EMon]			
	NF-04				
	NF-05				
	NF-06	[E2EMon]	[E2EMon]		
	NF-07		[per 08]	[per 08]	[per 08]
	NF-08				

Tabelle 4.1.: Bewertung der Ansätze aus dem interorganisationalen ITSM gegenüber den Anforderungen

Arbeit bezüglich Gesamtüberwachung (FM-M02), Überwachung eines interorganisationalen Dienstes (FM-M03) sowie Kommunikationsprotokolle (KM-03) sind diese Ansätze sehr wichtig.

Die Ansätze aus dem interorganisationalen Umfeld werden in Tabelle 4.1 bzgl. Abdeckung der Anforderungen zusammengefasst. Die grau hinterlegten Boxen repräsentieren die relevanten Felder aus Tabelle 3.20 des vorherigen Kapitels (Seite 93).

4.2. Arbeiten zum ITSM bezogen auf Fehlermanagement

In diesem Abschnitt werden Arbeiten im bereich ITSM betrachtet. Auch wenn es sich in dieser Arbeit um eine allgemeine Managementarchitektur handelt, wird beim methodischen Vorgehen zuerst ein berechnungsunabhängiges Modell (CIM des MDA-Ansatzes) benötigt. In dieser Arbeit korrespondiert das CIM-Modell mit der Prozesssicht, die in diesem Fall von allgemein anerkannten Prozessen bzgl. Fehlermanagement repräsentiert werden sollen. Allgemein anerkannt sind diesbezüglich die ITSM-Ansätze, die Managementprozesse für unterschiedliche Funktionsbereiche des Service Managements zur Verfügung stellen. Die ersten zwei Themengebiete dieses Abschnitts fallen in die Kategorie Best-Practice (Abschnitt 4.2.1) bzw. Standards (Abschnitt 4.2.2); die restlichen sind relevante Arbeiten aus der Wissenschaft und Forschung (Abschnitte 4.2.3 und 4.2.4).

4.2.1. Incident-Management-Prozess nach ITIL

Die IT Infrastructure Library (ITIL) ist ein umfassendes Rahmenwerk für ein kundenorientiertes und kosteneffizientes Management von IT-Diensten. ITIL ist kein theoretisches Rahmenwerk, sondern beschreibt Verfahrensweisen des IT-Service-Managements unter einem praktischen Gesichtspunkt [Bren 07]. ITIL ist heute das ITSM-Framework mit der größten Verbreitung insbesondere im europäischen Raum; so gaben im Jahr 2008 bereits 74% der im Rahmen einer Online-Umfrage befragten Unternehmen im deutschsprachigen Raum an, ITIL einzusetzen [Uebe 09]. ITIL hat auch Eingang gefunden in andere ITSM Frameworks, wie das Microsoft Operations Framework (MOF), das IT Process Model (ITPM) von IBM oder ITSM von HP [Musi 09, LeHe 05]. ITIL fungiert zudem als Grundlage des ISO/IEC 20000 Standards [ISO 20000:1].

Allgemeines zur ITIL

Im Mittelpunkt von ITIL befindet sich der Begriff Prozess. Dieser ist definiert als eine Menge koordinierter Aktivitäten, die Ressourcen zur Erzielung von Ergebnissen nutzen bzw. transformieren. Die Strukturiertheit der ITIL-Prozesse weist eine große Bandbreite auf; nicht für alle Prozesse wird ein explizites Prozessmodell angegeben. Es werden Referenzprozesse definiert, die die Maßnahmen und Tätigkeiten des IT-Service-Management in den unterschiedlichen Bereichen beschreiben. Für jeden Prozess werden kritische Erfolgsfaktoren sowie

Key Performance Indicators (KPI) angegeben, die die Messbarkeit des Prozesserfolgs ermöglichen. Das Prozessmanagement nach ITIL basiert auf Prinzipien der zyklischen Optimierung der Prozessqualität nach Deming.

Nach der ITIL Version 2 mit seinen zwei Teilen Service Support [OGC 00] und Service Delivery [OGC 01] folgte im Jahr 2007 ITIL Version 3. Diese besteht aus fünf Büchern, die entlang des Lebenszyklus von IT-Diensten organisiert sind: Service Startegy [OGC 07a], Service Design [OGC 07b], Service Transition [OGC 07c], Service Operation [OGC 07d] und Continual Service Improvement [OGC 07e].

Für diese Arbeit sind der Incident-Management-Prozess (Teil von Service Operation) und die Funktion Service Desk die wichtigsten Ansätze der ITIL. Das Incident-Management kümmert sich um die schnellstmögliche Wiederherstellung der normalen Dienstfunktionalität nach dem Auftreten einer Störung. Ein Service Desk ist die Schnittstelle einer IT-Organisation zu ihren Nutzern. Diese Funktion nimmt Störungsmeldungen und Service Requests entgegen und koordiniert die nachfolgenden Betriebseinheiten (vgl. mit den Szenarienbeschreibungen in den Abschnitten 3.1.2.2 und 3.1.3.2). Das Service Desk kann die administrativen Tätigkeiten des Incident Managements übernehmen.

Nach dieser kurzen Übersicht über ITIL wird stellvertretend der Incident-Management-Prozess genauer betrachtet (Darstellung nach [Hamm 09])

Ziel des Incident-Management-Prozesses

Das Ziel des Incident-Management-Prozesses nach ITIL ist die schnellstmögliche Wiederherstellung des normalen Dienstbetriebs und die Minimierung der Auswirkungen einer Störung auf die Geschäftstätigkeit. Eine Störung ist dabei definiert als eine nicht geplante Unterbrechung oder eine Einschränkung der mit dem Kunden vereinbarten Qualität eines IT-Dienstes. Der Prozess trägt dazu bei, die mit dem Kunden vereinbarte Dienstqualität und -verfügbarkeit zu erreichen und damit die SLAs einzuhalten. Incident Management beschäftigt sich mit allen Ereignissen, die den normalen Dienstbetrieb unterbrechen und die als Resultat eine Abweichung vom vereinbarten IT-Dienst haben.

Prozessschritte

Abbildung 4.2 zeigt die Aktivitäten des Incident-Management-Prozesses nach ITIL. Störungen können von Nutzern, dem Service Desk oder dem Event Management an das Incident Management berichtet werden. Die erste Phase des Prozesses ist die Identifzierung der Störung (Incident Identification), danach wird diese erfasst (Incident Logging) z. B. in einem Trouble Ticket System (TTS). Die Störung wird dann klassifiziert, d. h. eine Kategorisierung anhand des Dienstkatalogs wird vorgenommen (Incident Priorization) und die Priorität der Störung festgestellt (Initial Diagnosis).

Ergibt die Kategorisierung, dass es sich um eine erhebliche Störung (Major Incident) handelt, wird eine spezielle Prozedur (Major Incident Procedure) initiiert. Die Störungsbearbeitung beginnt mit einer ersten Diagnose der Störung (Initial Diagnosis). Falls die Störung vom Service Desk aufgenommen wurde, übernimmt dieses die Aufgabe des Incident-Managements; Ziel ist, dass das Service Desk eigenständig die Störung lösen kann. Abhängig davon, ob es die Störung lösen kann oder nicht, führt das Service Desk die weiteren Untersuchungen (Investigation & Diagnosis) durch.

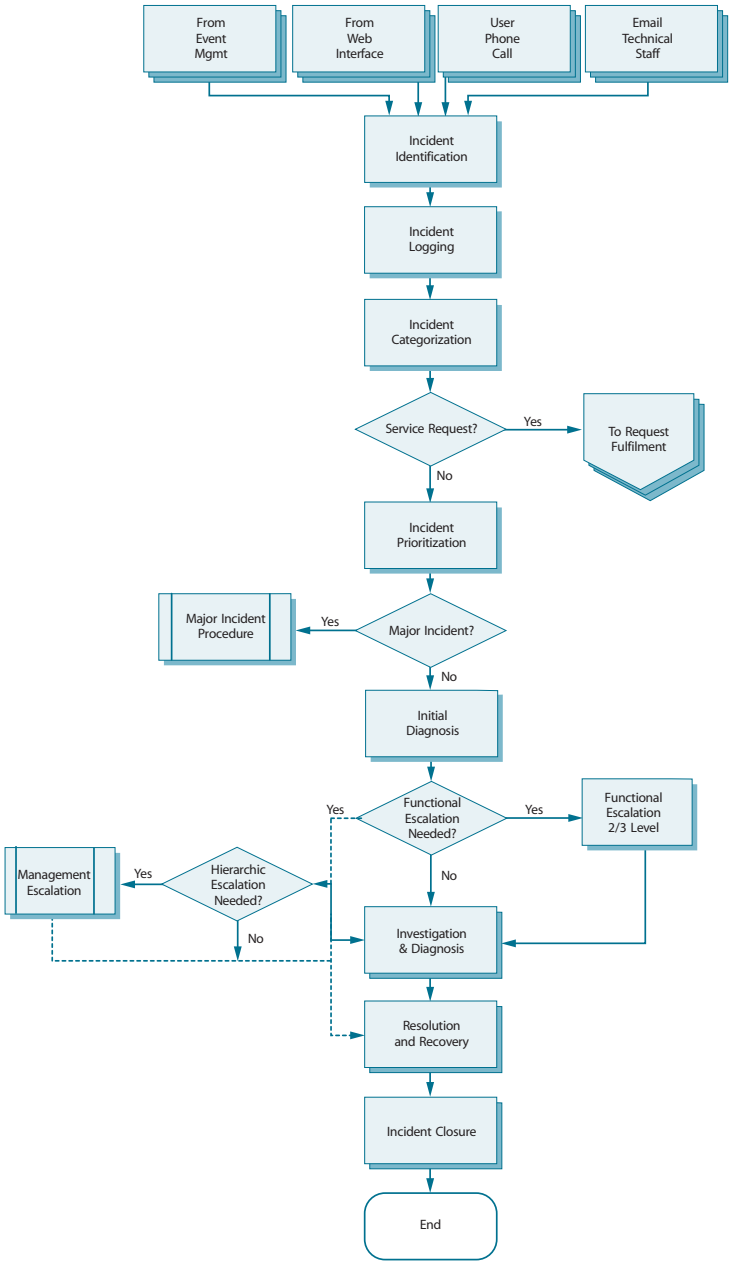


Abbildung 4.2.: ITIL Incident Management Prozess [OGC 07d]

Kann das Service Desk die Störung nicht selbständig lösen, erfolgt eine funktionale Eskalation (Functional Escalation), d. h. weitere Spezialistengruppen werden in die Störungsbearbeitung mit einbezogen. Unterstützend dafür sind Datenbanken mit bekannten Fehlern (Known Errors Data Bases) und Workarounds, alte Störungsmeldungen sowie weitere Informationsquellen. Wird eine Lösung nicht bzw. nicht in der mit dem Kunden vereinbarten Zeit gefunden, erfolgt eine hierarchische Eskalation (Hierarchical Escalation), d. h. eine Information an den verantwortlichen IT Service Manager. Je nach Bedarf kann die hierarchische Eskalation auch weiter in der Hierarchie fortschreiten, um geeignete Maßnahmen zur Lösung der Störung herbeizuführen (z. B. Hinzuziehen externer Lieferanten).

Falls eine Lösung gefunden wurde, erfolgt im Schritt Resolution and Recovery die Wiederherstellung der Dienstfunktionalität entsprechend der gefundenen Lösung. Abschließend im Schritt Incident Closure wird die Störung abgeschlossen, jedoch erst nach Rückfrage beim Kunden, ob die Dienstfunktionalität wieder zur Zufriedenheit besteht.

4.2.2. Incident Management Prozess nach ISO/IEC 20000

ISO/IEC 20000 ist ein internationaler Standard in Bereich ITSM, der auf Best Practices im ITSM (z. B ITIL, MOF, CoBIT) und auf Grundlagen des Qualitätsmanagements (nach ISO 9000) basiert. Dieser wurde verabschiedet mit dem Ziel, die Einführung eines integrierten prozessorientierten Ansatzes für die Bereitstellung von IT-Diensten zu fördern. ISO/IEC 20000 definiert Mindestanforderungen an einen IT-Service Provider, die erfüllt werden müssen, um Dienste gemäß der mit den Kunden vereinbarten Qualität bereitstellen zu können. Der Standard besteht aus zwei Dokumenten: Teil 1 Specification [ISO 20000:1] und Teil 2 Code of Practice [ISO 20000:2]. Im Teil 1 werden Anforderungen an Service-Provider aufgestellt, die ihren Kunden gemanagte Services mit vereinbarter Qualität anbieten wollen. Teil 2 hingegen beinhaltet zusätzliche Empfehlungen und Leitlinien für Auditoren und Service-Provider.

Der Incident-Management-Prozess der ISO/IEC 20000 ist in Anlehnung an ITIL Version 2 und Version 3 beschrieben, d. h. er beinhaltet dieselben Prozessschritte. Einer der großen Unterschiede zwischen den Standard- und den Best-Practice-Ansätzen ist, dass keine Angaben gemacht werden, ob ein Service Desk existieren muss oder nicht; die einzige Vorgabe ist, dass *„alle Störungen erfasst werden müssen"*.

4.2.3. ITSM-Prozesse Verketteter Dienste

Die oben genannten ITSM-Ansätze eignen sich sehr gut für intraorganisationale IT-Dienste oder für IT-Dienste, die eine hierarchische Form der Diensterbringung aufweisen. Für heterarchische Formen der Diensterbringung (wie in Abschnitt 2.1.3.4 beschrieben) können diese Prozesse nicht unmittelbar eingesetzt werden.

Hamm beschreibt in [Hamm 09] die Methode ITSM*CooP* zur Spezifikation der ITSM-Prozesse für Verkettete Dienste, das sind kooperativ von mehreren Providern erbrachte Dienste. Zunächst werden Konventionen für die Modellierung interorganisationaler ITSM-Prozesse bereitgestellt. Ebenfalls werden Empfehlungen für die Definition von Betriebsprozessen Verketteter Dienste gegeben. Die Vorgehensweise basiert auf der Prozessmodellierungssprache BPML sowie dem Informationsmodell SID und erweitert diese um die eingeführten Modellierungskonventionen.

Die Vorgehensweise zur Prozessdefinition in ITSM*CooP* setzt die Formierung eines Providernetzwerks, das Vorhandensein eines Dienstmodells sowie geeigneter Referenzprozesse voraus. Ausgehend von diesen Voraussetzungen erfolgt die eigentliche Prozessdefinition in fünf Schritten.

So werden auf Basis der bereitgestellten Modellierungskonventionen und Empfehlungen Referenzprozesse für die heterarchischen Formen der Diensterbringung definiert und implementiert. Für die vorliegende Arbeit ist der Incident-Management-Referenzprozess äußerst wichtig.

Ziele

Der Incident Management Prozess für Verkettete Dienste übernimmt die Ziele des in ITIL definierten Prozesses und überträgt die damit verbundenen Aufgaben auf die spezifische Dienst- und Organisationsstruktur Verketteter Dienste.

Prozessschritte

Da der ITSM*CooP* -Referenzprozesse auf ITIL aufbaut, werden für den Incident-Management-Prozess verketteter Dienste teilweise dieselben Aktivitäten benutzt. Wie in Abbildung 4.3 dargestellt, kommen hier noch drei weiteren Aktivitäten hinzu: Aktivität A_5 – Ask for Diagnosis ist zuständig für die Ermittlung und Weiterleitung der Störung an die betroffenen Teildienst-Provider. Diese ersetzt die funktionale Eskalation vom 2. zum 3. Level des ITIL-Prozesses.

Mit den Aktivitäten A_7 – Wait for Diagnosis und A_8 – Resolution/Recovery prüft das Service Desk die Diagnose und beauftragt die Wiederherstellung der Dienstfunktionalität. Die Aktivität A_9 – Resolution & Recovery bezieht sich auf das Wiederherstellen eines Teildienstes, die Aktivität A_{11} – Ask for Resolution auf den gesamten Verketteten Dienst. Diese zwei Aktivitäten sind durch Aufteilung der Aktivität Resolution & Recovery des ITIL Prozesses entstanden.

Im Incident-Management-Prozess der ITSM*CooP* sind folgenden Rollen beteiligt: Der Service Desk, der für die Kommunikation mit den Anwendern, die Erfassung von Störungsmeldungen und die Koordination nachgelagerter Support-Gruppen zuständig ist. Der Partial Service Support, der die Diagnose von Störungsursachen für einen Teildienst auf Basis von Informationen über die Dienstinstanz (Dienstmodell) durchführt. Und schließlich der Partial Service Support (Causative), der die Behebung von Störungen für einen Teildienst innerhalb seiner Domäne vornimmt.

Der Referenzprozess ist in Abbildung 4.4 dargestellt.

Aktivität	Bezeichnung	Beschreibung	
A_1	INCIDENT IDENTIFIKATION	*Input*	Event oder Störungsmeldung
		Tätigkeiten	Erkennen einer Störung
		Output	–
A_2	INCIDENT LOGGING	*Input*	Details der Störung
		Tätigkeiten	Anlegen eines Incident Record
		Output	Incident Record
A_3	INCIDENT CLASSIFICATION	*Input*	Incident Record
		Tätigkeiten	Zuordnung der Störung zu einer Dienstinstanz (Klassifikation), Ermittlung der Auswirkungen und der Dringlichkeit der Störung (Priorisierung)
		Output	Incident Record (ergänzt)
A_4	INITIAL DIAGNOSIS	*Input*	Incident Record Known Errors
		Tätigkeiten	Erste Diagnose einer Störungsursache auf Ebene des Verketteten Dienstes (auf Basis von Known Errors oder Daten aus Monitoring-Systemen, wenn vorhanden)
		Output	Incident Record (ergänzt)
A_5	ASK FOR DIAGNOSIS	*Input*	Incident Record
		Tätigkeiten	Ermittlung und Weiterleiten der Störung an die betroffenen Teildienst-Provider
		Output	–
A_6	INVESTIGATION & DIAGNOSIS	*Input*	Incident Record
		Tätigkeiten	Diagnose der Störungsursache auf Ebene eines Teildienstes
		Output	Incident Record (ergänzt)
A_7	CHECK DIAGNOSIS	*Input*	Incident Record
		Tätigkeiten	Rückmeldungen der Teildienst-Provider prüfen
		Output	–
A_8	ASK FOR RESOLUTION	*Input*	Input
		Tätigkeiten	Beauftragung der die Störung verursachenden Teildienst-Provider zur Wiederherstellung der Dienstfunktionalität
		Output	–
A_9	RESOLUTION & RECOVERY	*Input*	Incident Record
		Tätigkeiten	Wiederherstellung der Dienstfunktionalität eines Teildienstes
		Output	Incident Record (ergänzt)
A_{10}	CHECK RESOLUTION	*Input*	Incident Record
		Tätigkeiten	Prüfen der durchgeführten Tätigkeiten zur Wiederherstellung der Dienstfunktionalität
		Output	Incident Record (ergänzt)
A_{11}	RESOLUTION & RECOVERY	*Input*	Incident Record
		Tätigkeiten	Wiederherstellung der Dienstfunktionalität
		Output	Incident Record (ergänzt)
A_{12}	INCIDENT CLOSURE	*Input*	Incident Record
		Tätigkeiten	Abschluss des Incident Record, nach Rücksprache mit Anwender (nur wenn Störungsmeldung vorliegt)
		Output	Incident Record (ergänzt)

Abbildung 4.3.: Aktivitäten des ITSM*CooP* -Referenzprozesses Incident-Management [Hamm 09]

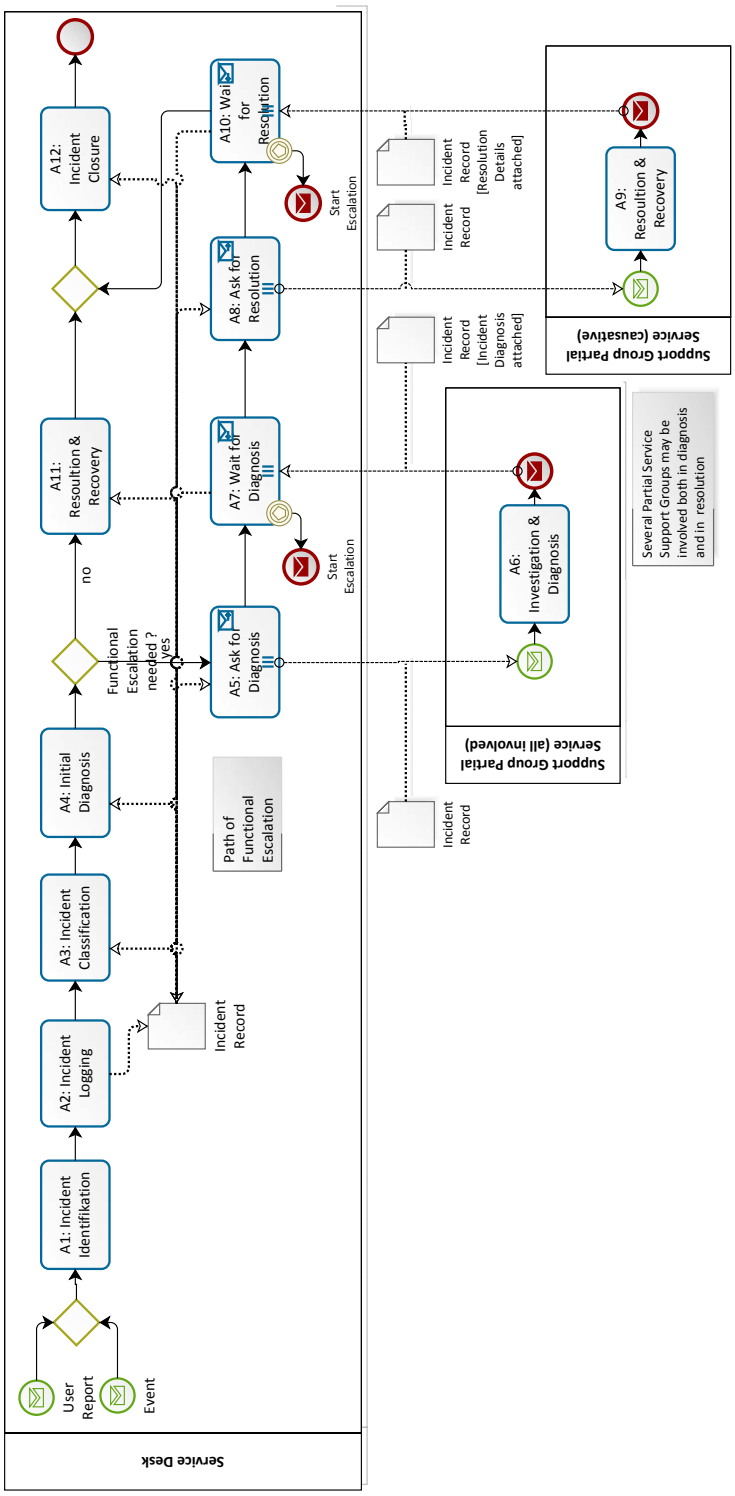


Abbildung 4.4.: Das globale Prozessmodell des ITSM*CooP* Incident-Management-Prozesses [Hamm 09]

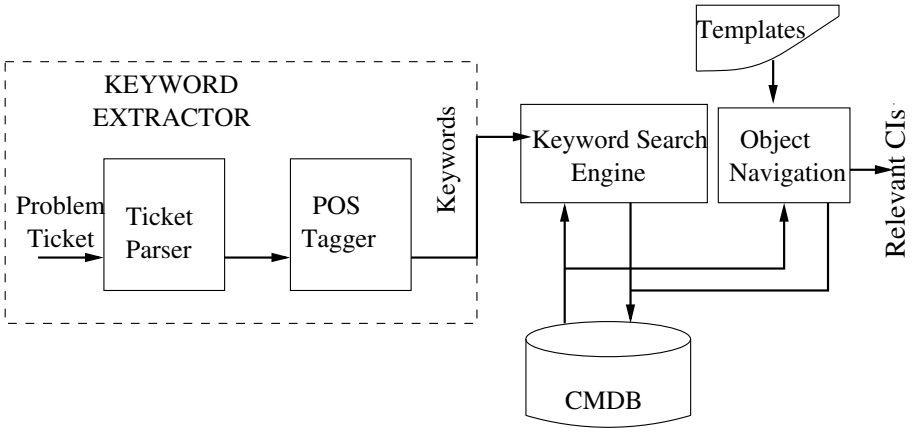


Abbildung 4.5.: Systemarchitektur von Incident-Management-Tools nach [GPM 08a]

4.2.4. Automatisierung des ITSM Incident-Management-Prozesses

Gupta et al. schlagen in [GPM 08a] eine mögliche Automatisierung des Incident-Management-Prozesses vor. Dabei werden nur Tickets von Nutzern betrachtet. Eine ausführliche Beschreibung der entsprechenden Architektur und Implementierungsaspekte werden zusätzlich in [GPM 08b] gegeben. Dieser Ansatz basiert auf der Suche nach Schlüsselworten (engl. keyword based search). Hierfür wird eine Systemarchitektur definiert und implementiert (Abbildung 4.5). Das Beschreibungsfeld in einem Incident-Ticket wird benutzt, um die Schlüsselwortsuche in einer Configuration Management Database (CMDB) nach Configuration Item (CI) durchzuführen. Dies erfolgt anhand von drei Komponenten (siehe Abbildung 4.5): keyword extractor (für das Extrahieren des Beschreibungsfeldes), keyword search engine (für die Durchsuchung der CMDB nach zutreffenden CIs) und object navigation (für das Herausfinden der fehlerhaften CIs, wenn diese im Ticket nicht explizit aufgeschrieben werden). Zudem werden in [GPL$^+$ 09] Informationen von Monitoring-Systemen in Betracht gezogen, um die automatische Klassifikation von Incidents zu vervollständigen.

Bartolini et al. schlagen in [BST 09] ein Tool für kritisches Incident-Management vor, das die Auswirkungen, die Störungen auf den Geschäftsprozesse haben, berücksichtigt. Dieses Tool (HANNIBAL) unterstützt Manager auf der Geschäftsebene bei der Entscheidungsfindung bzgl. kritischer Management-Prozesse auf unterschiedlichen organisatorischen Ebenen. HANNIBAL besteht aus vier Phasen: Empfehlung möglicher Strategien, die geführt werden können (engl. Strategy Selection), Einschätzung der Leistung des Incident-Management-Prozesses für jede dieser möglichen Strategien (engl. Performance Evaluation), Kostenberechnung für jeder dieser Strategien (engl. Cost Estimation) und anschließend die Auswahl der Strategie mit den kleinstmöglichen Auswirkungen auf den Geschäftsprozess (engl. Business Impact Analysis). Für die Leistungsberechnung der entsprechenden Strategien wird ein

weiteres Tool benutzt, SYMulation for Incident ANalysis (SYMIAN) [BST 08], das die IT-Supportorganisation als ein Warteschlangensystem betrachtet. Es wird dabei zwischen zwei Teilen der Bearbeitungszeit eines Incidents unterschieden: Die Arbeitszeit (Working-Time), die die tatsächliche Zeit wiedergibt, die Operateure oder Spezialisten für die Behebung einer Störung benötigen, und die Wartezeit (Waiting-Time), die Verweildauer der Tickets innerhalb des Warteschlangensystems. Das hier benutzte Konzept hat als Grundlage die Impact-Analyse beschrieben in [Schm 08], in der ein Datenmodell für die möglichen Degradierungen auf Ressourcen-, IT-Dienst- und Geschäftsebene vorgeschlagen ist.

4.2.5. Bewertung

Die betrachteten ITSM-Ansätze haben gemeinsame Eigenschaften: Die Prozessorientierung und einen hohen Abstraktionsgrad. Sowohl im ITIL bzw. ISO/IEC 20000 als auch bei ITSM*CooP* kann man eine klare Definition der Prozesse mit klar definierten Prozessschritten und Rollen beobachten. Die ersten zwei können hauptsächlich auf hierarchische Formen der Diensterbringung angewandt werden, und der auf heterarchische. Die in dieser Arbeit angestrebte ioFMA wird anhand des MDA-Ansatzes entwickelt (siehe Abschnitt 2.3). Auf der obersten Ebene steht das CIM als berechnungsunabhängiges Modell. Für CIM (bzw. die Prozesssicht) ist es notwendig, ein unabhängiges Modell zu besitzen, und dies ist von den Prozessen der ITIL bzw. der ITSM*CooP* erfüllt. Darauf können die weiteren Schritte der Entwicklung beruhen, insbesondere die Konzeption eines PIM (bzw. der Architekturschicht). ITIL und ISO/IEC 20000 auf der einen und Hamms ITSM*CooP* auf der anderen Seite unterstützen den Lebenszyklus eines Fehlers (Anforderung IM-04). Wertvolle Informationen für die Realisierung des Organisationsmodells können aus den ITSM-Ansätzen herangezogen werden, sowohl für die Anforderung OM-01 bzgl. der Unterstützung beider Formen der Diensterbringung als auch für OM-02, was die Rollendefinition betrifft. Auch die Prozessbeschreibungen werden in der Realisierung des Funktionsmodells mit einbezogen.

Bei anderen Ansätzen, die hier erläutert wurden [BST 08, BST 09, GPM 08a, GPL$^+$ 09], steht hauptsächlich der nicht-funktionale Aspekt Automatisierung (NF-06) in Vordergrund. Weitere nicht-funktionale Anforderungen, die der Ansatz von Bartolini et al. in [BST 08, BST 09] erfüllt, sind Skalierbarkeit NF-04, Performanz NF-05 und eine gemeinsame Informationsdatenbank NF-07 in Form einer CMDB. Bei Gupta et al. wird noch zusätzlich zur Automatisierung (NF-06) und CMDB (NF-07) die Möglichkeit der Definition eines gemeinsamen Datenformats erwähnt (IM-01).

Tabelle 4.2 stellt die Abdeckung der Anforderungen aus Abschnitt 3.2 durch die beschriebenen ITSM-Ansätze dar. Die grau hinterlegten Felder repräsentieren die relevante Felder der Tabelle 3.20 im vorherigen Kapitel (Seite 93).

Anforderungen		Phasen des Lebenszyklus eines Fehlers			
		Entdecken	Eingrenzen	Beheben	Vorhersage
an IM	IM-01		[GPM 08a]		
	IM-02	[GPL$^+$ 09]	[GPL$^+$ 09]		
	IM-03				
	IM-04	[Hamm 09]	[Hamm 09]	[Hamm 09]	
	IM-05				
	IM-06	[GPL$^+$ 09]	[GPL$^+$ 09]		
an OM	OM-01	[OGC 07d, Hamm 09]	[OGC 07d, Hamm 09]	[OGC 07d, Hamm 09]	[OGC 07d, Hamm 09]
	OM-02	[OGC 07d, Hamm 09]	[OGC 07d, Hamm 09]	[OGC 07d, Hamm 09]	
an FM	FM-L01	[OGC 07d, Hamm 09]	[OGC 07d, Hamm 09]		
	FM-L02	[OGC 07d, Hamm 09]			
	FM-L03	[Hamm 09]	[Hamm 09]		
	FM-P01		[OGC 07d, Hamm 09]	[OGC 07d, Hamm 09]	
	FM-P02			[OGC 07d, Hamm 09]	
	FM-M01				
	FM-M02	[Hamm 09]	[Hamm 09]	[Hamm 09]	
	FM-M03	[Hamm 09]	[Hamm 09]	[Hamm 09]	
	FM-R01				
	FM-R02				
	FM-R03				
	FM-F01				
	FM-F02				
	FM-01				
	FM-02				
an KM	KM-01				
	KM-02				
	KM-03				
NFA	NF-01				
	NF-02				
	NF-03				
	NF-04	[BST 08, BST 09]	[BST 08, BST 09]	[BST 08, BST 09]	
	NF-05		[BST 08, BST 09]	[BST 08, BST 09]	
	NF-06	[GPM 08a]	[GPM 08a]	[GPM 08a]	
	NF-07		[BST 08, BST 09, GPM 08a]	[BST 08, BST 09, GPM 08a]	
	NF-08				

Tabelle 4.2.: Abdeckung der Anforderungen durch ITSM-Ansätze

4.3. Arbeiten zum Fehlermanagement allgemein

Schon in den 90er Jahren ist das Fehlermanagement zu einem der wichtigsten Bereiche des Netzmanagements geworden. Spätestens 1994, als das Open System Interconnection (OSI) Referenzmodell herausgegeben wurde [OSI 94], ist das Fehlermanagement Teil der Computernetze geworden. In [StVa 98] adressieren Stanley und Vaidhyanathan das Thema Fehlerfortpflanzung und skizzieren ein Framework für Echtzeit-Fehlermanagement großer Systeme. Dies wird mit Hilfe von Causal Directed Graph (CDG) realisiert, in dem die Fehlerfortpflanzung durch mögliche Fehlerpfade modelliert wird und dadurch eine Korrelation (Causalität) zwischen Fehler-Ereignissen (*fault events*) und Symptom-Ereignissen (*symptom events*) hergestellt werden kann.

Oppenheimer et al. untersucht in [OGP 03] allgemein die Fehlerproblematik im Internet. Dies ist eine zusammenfassende Studie bezüglich Störungen, deren Ursachen, Anwendungsfällen und Vermeidung. Vergleichbar realisieren Medeios et al. eine Analyse der Fehler, die in Grids auftreten, und deren Konsequenzen [MCBS 03]. Rahman et al. veröffentlichten in 2006 eine Studie [RBM 06] bezüglich der Fehlerursachen und -verbreitung in kritischen Infrastrukturen. Diese basiert auf Vorfälle, die in einem Zeitraum 12 Jahren veröffentlicht wurden. Die Studie wurde 2009 [RBM 09] basierend auf neuen Daten erweitert. In [KSW 07] schlagen Kapadia et al. eine generische Fehlermanagement-Methode, basierend auf einem Real-World-Modell zur Unterstützung vielfältiger Applikationen vor. Diese Methode soll für vielfältige praktische Aktivitätsbereiche zur Verfügung stehen.

Meist konzentrieren sich die Arbeiten zu Fehlermanagement nicht auf den ganzen Fehlerlebenszyklus, sondern nur spezifisch auf eine oder zwei Phasen. In diesem Sinne wird die Vorstellung der verwandten Arbeiten in den folgenden Abschnitten strukturiert.

4.3.1. Arbeiten zur Fehlerentdeckung

Im Bereich Fehlerentdeckung gibt es mehrere Ansätze. Steider realisiert in [StSe 04a] eine Bestandsaufnahme der Fehlerlokalisierungstechniken in Computernetzen. Abbildung 4.6 fasst diese unterschiedlichen Techniken zusammen. Diese umfassen viele Bereiche der Informatik wie Künstliche Intelligenz, Graphentheorie, Neuronale Netze, Informationstheorie und Automatentheorie und beinhalten Modell-basierte (engl. *model-based*) und Fall-basierte (engl. *case-based*) Werkzeuge sowie auch Techniken der Graphentheorie und den Codebook-Ansatz.

probabilistische Fehlerentdeckung

In [StSe 04d] präsentieren Steider und Sethi eine Methode zur probabilistischen Fehlerentdeckung anhand von ,,Belief Networks". Der Fehler wird durch eine Bayesianische Methode unter Verwendung auch von unklaren, unsicheren und inkorrekten Informationen über das System lokalisiert. In [StSe 04c] wird eine Erweiterung dieser Methode vorgeschlagen, indem

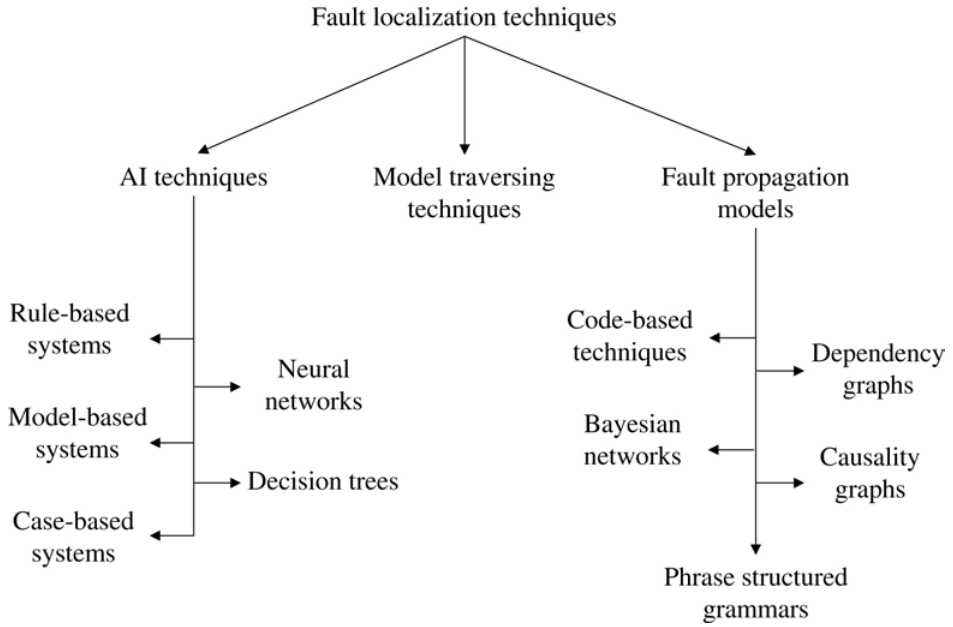


Abbildung 4.6.: Klassifizierung der Fehlerlokalisierungstechniken [StSe 04a]

man zur Fehlerlokalisierung anhand von Symptomerklärungen übergeht. Dafür wird eine sogenannte Fehlersymptomkarte (engl. symptom-fault map) als Fehlerfortpflanzungsmodell benutzt. Durch diese Methode werden alle wahrscheinlichsten Fehlerkandidaten herausgesucht. Diese stellen eine Menge an möglichen Hypothesen dar, die eine vollständige Erklärung für die bisher bemerkten Symptome darstellt. Sie werden dann entsprechend deren Güte eingestuft. Die Methode kann gleichzeitige voneinander unabhängige Fehler identifizieren und betrachtet sowohl positive als auch negative Symptome in der Analyse. Für den vorliegenden Ansatz ist das ein relevanter Beitrag zur Fehlerlokalisierung in der eigenen Domäne (FM-L01) zur Fehlerentdeckung und Fehlereingrenzung.

Fehlerentdeckung in Multi-Agenten-Systeme

In ihrer Arbeit [XuDe 05] stellen Xu und Deters einen Ansatz zur Fehlerentdeckung in Multi-Agenten-Systemen (MAS) vor. Multi-Agenten-Systeme beinhalten eine große Anzahl an lose gekoppelten, heterogenen Agenten, die auf unterschiedlichen Prozessen gehostet werden. Es wird für MAS eine nichtinvasive Event-Reporting-Technik, bei der die Agenten Report-, Statusänderungs-, Warnungs- und Fehlerereignisse senden, vorgestellt. Diese Ereignisnachrichten von den Agenten werden unter einem sogenannten Event-Stream zusammengefasst und zu einem Event-Manager zur Speicherung und Bearbeitung weitergeleitet. Auch hier wird das Thema Fehlerlokalisierung in der eigenen Domäne (FM-L01) in der Phase der Fehlerentdeckung beschrieben. Der Ansatz ist für die vorliegende Arbeit nützlich für die Ermittlung des Status der Fehlerbearbeitung/Wartung (FM-P01, FM-P02) in der Eingrenzungsphase und beim Fehlerbeheben. Der Ansatz trägt auch zu den Punkten Skalierbarkeit

(NF-04) und Datenaktualität (NF-03) sowohl für Fehlerentdeckung als auch für Fehlereingrenzung bei.

Fehlerentdeckung via Overlay Netze

Hirota et al. schlagen in [HKT+ 10] eine Methode zur Fehlerentdeckung basierend auf verteilten Komponenten vor. Dies passiert durch Ansammeln einer kleinen Anzahl an Nachrichten via multiplen Overlay Netzen. Eine fehlerhafte Komponente wird dann durch eine andere geeignete, funktionsfähige Komponente des Netzes substituiert. Es werden regelmäßig Informationen über den Zustand der Nachbarkomponenten innerhalb des Netzes angefordert. Aufgrund der Antwort bzw. dem Ausbleiben einer Antwort wird ein Fehler identifiziert. Die Fehlerentdeckung wird für unterschiedliche Komponenten beschrieben. Ebenfalls wird die Möglichkeit von Fehlalarmen betrachtet. Zusätzlich zu seiner Relevanz für die Fehlerlokalisierung und Monitoring in der eigenen Domäne bringt dieser Ansatz Zusatzinformationen zur Lokalisierung von False-Positives Fehlermeldungen (FM-F01), zu benutzten Kommunikationsprotokollen (KM-03) sowie zu Skalierbarkeit (NF-04) und Automatisierung (NF-06).

Action Integrated Fault Reasoning

Eine Methode zur Fehlerlokalisierung, die die Vorteile des aktiven und passiven Monitorings zusammenfasst, stellt Tang et al. in [TASB 05] vor. Das tatsächliche Ziel dieser Arbeit ist die Steigerung der Effizienz und Präzision der Fehlerlokalisierungstechniken. Diese Methode heißt Action Integrated Fault Reasoning (AIR). Die Autoren ziehen in Betracht, dass öfters die Fehlerlokalisierung von nicht entdeckten oder verfälschten Symptomen beeinträchtigt wird. Dafür schlagen sie einen Satz an möglichen Symptomen, die möglicherweise auftreten können, vor. Eine solche Hypothese über die gegebenen Symptome muss eine hohe Genauigkeit beweisen; diese werden entweder bestätigt und damit steigt die Genauigkeit, oder neue Erkenntnisse bzw. Symptome kommen hinzu durch die eine neue Hypothese etabliert werden kann. Für die vorliegende Arbeit sind folgende Punkte relevant: Fehlerlokalisierung in der eigenen Domäne (FM-L01), Lokalisierung von False-Positives Fehlermeldungen (FM-F01) für die Fehlereingrenzung und Vorhersage/Vorbeugung. Zusätzlich wird ein gemeinsames Datenformat (IM-01) angesprochen als auch eine mögliche gemeinsame Informationsdatenbank (NF-07) und einen mögliche Automatisierung (NF-06).

Problem Determination and Resolution

Problem Determination and Resolution (PDR) ist ein Teilbereich des Fehlermanagements. Song et al. beschreiben in [SSS 09] eine mögliche Lösung für die Automatisierung der PDR. Dafür werden Methoden des maschinellen Lernens herangezogen. Der Hauptziel ist eine effektive Kategorisierung eines Fehlers (von Nutzer bemerkt) anhand der gelernten Performancedaten bzw. den Daten aus den Logs. Sie schlagen deswegen einen hierarchischen Algorithmus für inkrementelles Lernen vor. Dieser ermöglicht die dynamische Abbildung interner Klassifikationen auf reelle Situationen. Es ist eine sehr effiziente Methode, aber nur anhand von Web-Services getestet. Nichtsdestotrotz ist dieser Ansatz für die Fehlerlokalisierung in der eigenen Domäne (FM-L01) und für eine mögliche gemeinsame Informationsdatenbank (NF-07) relevant.

In das gleiche Themengebiet reiht sich auch die Arbeit [MDKP 09] ein, in der Mitra et al. das Problem Determination mit Hilfe von spatio-temporalen Mustern angehen. Der

Spatio-temporale Muster für PDR

hier beschriebene Algorithmus wird auf Event-Streams, die von Komponenten der IT-Services generiert werden, angewendet. Die in dem Algorithmus benutzten Muster sind unterschiedlicher Natur: zeitbezogen, ortsbezogen, Bereichsmuster, Baum-Muster oder allgemeine Graphen-Muster. Auch wenn es sehr effizient ist, so kann dieses Rahmenwerk nur für hierarchisch aufgebaute IT-Services benutzt (OM-01) werden, was seine Eignung für die vorliegende Arbeit eingrenzt. Nichtsdestotrotz ist es ein wertvoller Beitrag in Bezug auf Fehlerlokalisierung in der eigenen Domäne (FM-L01), die Definition eines gemeinsamen Datenformats (IM-01) als auch für die Erstellung einer möglicher Informationsdatenbank (NF-07).

Ereigniskorrelation

Hanemann beschreibt in [Hane 07] ein Rahmenwerk für Ereigniskorrelation, das grundlegend in der Dienstfehlerdiagnose benutzt werden soll. Für die Realisierung der Root Cause Analysis werden Techniken zur Ereigniskorrelation aus dem Netz- und Systemmanagement übernommen und angepasst. Da in der vorliegenden Arbeit das Fehlermanagement auf interorganisationaler Ebene untersucht wird und dabei die Korrelation von Ereignissen bei der Fehlersuche aus unterschiedlichen Domänen (FM-L01, FM-L02) notwendig ist, ist der von Hanemann beschriebene Ereigniskorrelator relevant. Nichtsdestotrotz fokussiert das Verfahren auf hierarchische Formen der Diensterbringung (OM-01).

Zeitkorrelation

Fehlerentdeckung anhand von Zeitkorrelation in dynamischen Netzen ist untersucht worden von Natu et al. in [NaSe 08]. Ein Ansatz für adaptive Fehlerlokalisierung wird hier vorgeschlagen basierend auf dynamischen Korrelationsmodellen und der Speicherung von partiellen Fehlerlokalisierungsergebnissen. Diese zwei werden in einem weiteren Schritt zu einer vereinigten Zeitkorrelation mit Fehlersymptomanalyse zusammengeführt. Die hier benutzte Methodik kann für die Problematik der Schnittstellendefinition (IM-03) benutzt werden. Der Ansatz ist zusätzlich für die Fehlerlokalisierung in der eigenen Domäne (FM-L01) als auch für das intraorganisationale Monitoring relevant.

Alarmkorrelation

Ein offensichtliches Problem in großen Netzen ist die große Anzahl von Alarmen, die aufgrund eines aufgetretenen Fehlers in Systemen erzeugt werden. In [BCF 94] beschreiben die Autoren eine Methode zur Alarmkorrelation in Kommunikationsnetzen. Bestimmte Fehlerinformationen werden während der Fehlerlokalisierung mit einem Alarm assoziiert.

Alle in diesem Abschnitt beschriebenen Ansätze wurden bereits bzgl. der Anforderungen bewertet. Obwohl dieser Abschnitt den Arbeiten zur Fehlererkennung gewidmet wurde, betreffen manche der hier vorgestellten Arbeiten mehreren Phasen des Fehlerlebenszyklus. Tabelle 4.3 fasst die Ansätze dieses Abschnittes zusammen. Die grau hinterlegten Boxen repräsentieren die die relevanten Felder der Tabelle 3.20 der vorherigen Kapitels (Seite 93).

4.3.2. Arbeiten zur Fehlereingrenzung

Netz-Korrelations-Graphen

Gopal realisiert in [Gopa 00] ein geschichtetes Model zur Unterstützung der Fehlereingrenzung. Als Grundlage für das Fehlermanagement werden dabei sowohl Netz-Korrelations-Graphen

Anforderungen		Phasen des Lebenszyklus eines Fehlers			
		Entdecken	Eingrenzen	Beheben	Vorhersage
an IM	IM-01		[MDKP 09]		
	IM-02				
	IM-03	[NaSe 06]	[NaSe 06]		
	IM-04				
	IM-05				
	IM-06				
an OM	OM-01	[NaSe 06, Hane 07]			
	OM-02				
an FM	FM-L01	[StSe 04d, StSe 04c, XuDe 05, HKT+ 10, TASB 05, SSS 09, NaSe 06, Hane 07]	[StSe 04d, StSe 04c]		
	FM-L02	[Hane 07]			
	FM-L03				
	FM-P01		[XuDe 05]	[XuDe 05]	
	FM-P02			[XuDe 05]	
	FM-M01	[HKT+ 10]			
	FM-M02				
	FM-M03				
	FM-R01				
	FM-R02				
	FM-R03				
	FM-F01		[HKT+ 10, TASB 05]		[TASB 05]
	FM-F02				
	FM-01	[HKT+ 10]			
	FM-02				
an KM	KM-01				
	KM-02				
	KM-03	[HKT+ 10]			
NFA	NF-01				
	NF-02				
	NF-03	[XuDe 05]			
	NF-04	[HKT+ 10]			
	NF-05				
	NF-06	[XuDe 05, HKT+ 10, TASB 05]	[XuDe 05]		
	NF-07		[NaSe 06, TASB 05, MDKP 09]		
	NF-08				

Tabelle 4.3.: Abdeckung der Anforderungen durch Ansätze der Phase Fehlerentdeckung

als auch Netz-Impakt-Graphen definiert. Diese werden in der operationalen Phase des Fehlermanagements benutzt bei der Automatisierung der Ereigniskorrelation für eine Root-Cause-Analyse. Das Modell beinhaltet systematische Informationen über Netzdienste, -schichten, -knoten, und -funktionen und unterstützt nicht nur die Fehlereingrenzung, sondern auch die Fehlerentdeckung und Wiederherstellung des Dienstes. Für die Anforderungen der vorliegenden Arbeit sind folgende Punkte relevant: Fehlerlokalisierung in der eigenen Domäne (FM-L01), Definition eines Datenformats (IM-01), Schnittstellendefinition(IM-03).

Netz-Wert-Management

Eine geschichtete Architektur zum Netz-Wert-Management wird in[ScZa 00] vorgestellt. Das Dienstmodell dieses wert-orientierten Netzmanagements basiert auf einem Korrelationsgraph mit Knoten (Managed Objects) und Kanten (Dienstbeziehungen). Ereignisse von den unterliegenden Schichten entsprechen operationalen Zuständen in den Managed Objects. Zustandsänderungen materialisieren sich in Alarmen, die auf einer höheren Schicht visualisiert werden können. Für die vorliegende Arbeit ist die Vorgehensweise und die daraus entstandene geschichtete Architektur relevant. Die folgenden Anforderungen werden berührt: Schnittstellendefinition(IM-03), Visualisierung (FM-01) und Fehlerbearbeitungsstatus (FM-P01).

4.3.2.1. Fehlerdiagnose

In der Literatur wird das Konzept der Fehlerdiagnose hauptsächlich für die Fehlereingrenzung (gemäß der Terminologie dieser Arbeit) benutzt. Aus diesem Grund wurden die Arbeiten zur Fehlerdiagnose an dieser Stelle zusammengeführt.

Fehlerdiagnose in Ad-Hoc Netze

Chessa und Santi schlagen in [ChSa 01] ein Fehlerdiagnose-Modell für Ad-Hoc Netze vor. Dieses ist ein vergleichsbasiertes Diagnosemodell, das auf einem 1:n Kommunikationsmodell basiert. Zwei Instanziierungen des Modells werden realisiert: Im ersten Fall verändert sich die Netztopologie während der Diagnose nicht. Es wird dabei ein effizientes Kommunikationsprotokoll für diesen Fall vorgeschlagen. Im zweiten Fall wird eine Veränderung der Topologie während der Diagnose zugelassen. Es wird dabei herausgefunden, dass die Dynamik der Szenarios die Qualität der Fehlerdiagnose im negativen Sinne beeinträchtigt. Für die vorliegende Arbeit bringt dieser Ansatz einen Mehrwert im Bereich Kommunikationsprotokolle (KM-03).

In Bereich der dynamischen Umgebungen wird eine mögliche Methode zur probabilistischen Fehlerdiagnose in [CQM$^+$ 09] vorgestellt. Cheng et al. betrachten hier ein mehrschichtiges Modell verteilter IT-Dienste (siehe Abbildung 4.7). Es werden Ausfälle von Ende-zu-Ende Diensten auf der obersten Schicht analysiert und fehlerhafte Ressourcen auf den unterliegenden Schichten identifiziert. Hierfür werden die Methoden Active-Probing und Passive-Monitoring herangezogen. Dieser Ansatz bringt zusätzliche Informationen im Bereich Fehlerlokalisierung (FM-L01, FM-L02) sowie bzgl. des Bearbeitungsstatus von administrativen

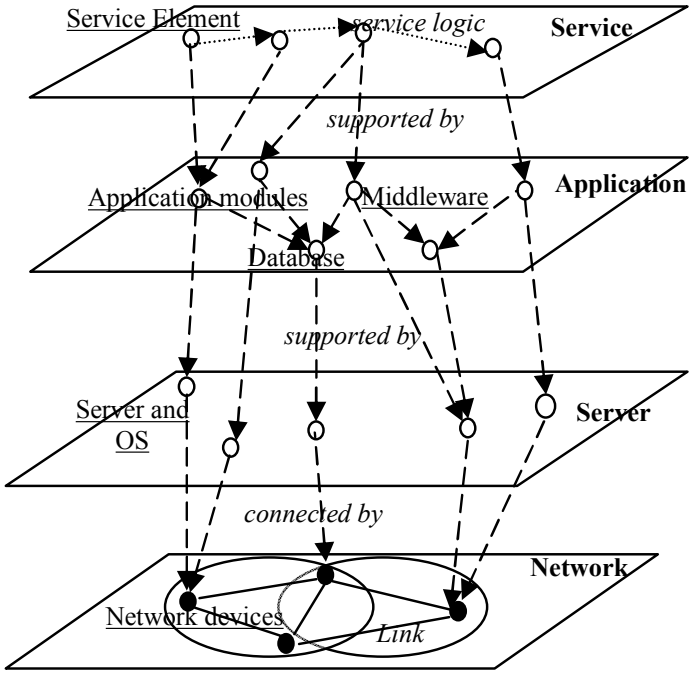


Abbildung 4.7.: Mehrschichtiges Model verteilter IT-Dienste nach [CQM$^+$ 09]

Tätigkeiten zur Fehlerbehebung oder Wartung (FM-P01, FM-P02). Zusätzlich sind organisatorische (IM-01) und Performance Aspekte (NF-05) besprochen

Mao et al. beschreiben in [MJTS 07] das NetworkMD, ein automatisches System für Fehlerdiagnose. Dieses leitet Fehlergruppen basierend auf historischen Fehlerdaten und (optional) geografischen Informationen ab. Die abgeleiteten Informationen spiegeln die ,,fehlende" Topologie wieder und tragen dazu bei, Fehler zu lokalisieren, eine Root Cause zu finden und Misskonfigurationen in bekannten Topologien herauszufinden. NetworkMD benutzt nichtüberwachtes Lernen (*engl. unsupervised learning*), ein maschinelles Lernen ohne im Voraus bekannte Zielwerte. Es weist hiermit auf die Notwendigkeit einer gemeinsamen Informationsdatenbank (NF-07) bzw. auf die Definition eines gemeinsamen Datenformats (IM-01) hin. Natürlich werden damit die intraorganisationale Fehlerlokalisierung (FM-L01) bzw. Monitoring (FM-M01) unterstützt als auch die Lokalisierung von False-Positives Fehlermeldungen (FM-F01). *NetworkMD*

Ebenso beschreiben Hofer und Fahringer in [HoFa 07] und [HoFa 08] Fehlerdiagnose-Modelle, die als maschinelles Lernen implementiert werden. Diese sind rule-based Beschreibungen, die die Automatisierung der Fehlerdiagnose verbessern. Dadurch werden die Punkte Automatisierung NF-06 und gemeinsame Informationsdatenbank NF-07 aus unser Anforderungsliste berührt.

Fehlerdiagnose in Wireless Netze

Das Thema der Fehlerdiagnose in Wireless Netzen wird in [QBRZ 05] beschrieben. Qiu et al. beschreiben eine simulationsbasierte Fehlerdiagnose. Dem Managementsystem wird ein Netzsimulator hinzugefügt, der das Entdecken und die Diagnose von Fehlern in einem operationalen Netz durchführen soll. Der Netzsimulator erfüllt zwei wichtige Voraussetzungen: Er muss sehr genau reproduzieren, was in dem operationalen Netz stattfand und führt auf dieser Basis Fehlerentdeckung und -diagnose aus. Die Autoren betrachten vier Kategorien von Netz-Fehlern: Packet dropping, Link congestion, externe Störungen und MAC-Missverhalten. Als Input für den Simulator werden Gruppen von gesammelten Daten aus der Netztopologie und Verkehrsstatistiken bereitgestellt. Obwohl es sich um Fehlermanagement in Wireless Netzen handelt, ist der Ansatz für die Fehlerlokalisierung in der eigenen Domäne (FM-L01), Statistiken (FM-R01), QoS-Parameter (FM-R02) als auch für die Performanz (NF-05) der Fehlerdiagnose relevant.

Korrelationsbäume Matrizen

Beygelzimmer et al. untersuchen in [BBMR 05] die Fehlerdiagnose unterstützt durch Korrelationsbäume und Matrizen. Um den Zustand eines Systems herauszufinden, sind vielfältige Informationen in Betracht zu ziehen. Das Problem ist die Repräsentation der korrelierten Informationen. Diese werden hauptsächlich in Ablaufdiagrammen dargestellt. Die Wartbarkeit der Ablaufdiagramme ist aber nicht so einfach. Unter diesen Umständen schlagen die Autoren eine effiziente Methode zur Optimierung dieser existierenden Ablaufdiagramme vor, basierend auf einer Hilfsmatrix. Durch diesen Ansatz werden Anhaltspunkte zu Konvertierungsmethoden (IM-02) und Aggregationsfunktionen (IM-06) in den Phasen der Fehlerentdeckung bzw. -eingrenzung gegeben.

Fehlerdiagnose in Grids

Duarte et al. beschreiben in [DBCAF 06] eine Methode zur kollaborativen Fehlerdiagnose in Grids mit Hilfe automatisierter Tests. Diese Methode zur Fehlerdiagnose in Grids besteht in Tests auf den fehlerhaften Komponenten, um herauszufinden, ob die Fehlerursache sich bei ihnen befindet oder bei einer darunterliegenden Komponente. Der Ansatz ist für die vorliegende Arbeit sehr interessant, da er eine Fehlerdiagnose auf mehreren Schichten vorschlägt, allerdings handelt sich hier hauptsächlich um das Testen von Softwarekomponenten auf unterschiedlichen Schichten und nicht auch für Netzkomponenten. Der Ansatz ist relevant für io-Fehlermanagement in Allgemeinen, aber auch speziell für die Automatisierung (NF-06).

Methoden zur Fehlerdiagnose

Venkatasubramanian et al. realisieren eine Studie über unterschiedliche Fehlerdiagnose-Methoden aus verschiedenen Perspektiven. Die Methoden werden in drei allgemeine Kategorien eingeteilt. In [VRYK 03a] werden quantitative modell-basierte Methoden analysiert. [VRYK 03b] gibt einen Überblick über Methoden, die auf qualitativen Modellen und Suchstrategien beruhen. Im letzten Teil [VRYK 03c] wird als Basis die Prozesshistorie genommen; große Mengen an historischen Prozessdaten werden transformiert und präsentiert, um ein Diagnosesystem zu realisieren. Am Ende werden die unterschiedlichen Methoden zur Fehlerdiagnose verglichen und die Grundzüge für ein Framework gesetzt. Es wird dann eine Liste mit Charakteristiken, die ein mögliches Fehlerdiagnosesystem erfüllen sollte, bereitgestellt. Die Arbeit gibt eine gute Übersicht über die möglichen Fehlereingrenzungsmethoden, die bei

Anforderungen		Phasen des Lebenszyklus eines Fehlers			
		Entdecken	Eingrenzen	Beheben	Vorhersage
an IM	IM-01		[Gopa 00, MJTS 07]		[MJTS 07]
	IM-02	[BBMR 05, VRYK 03b]	[BBMR 05, VRYK 03b]		
	IM-03	[Gopa 00]	[Gopa 00, ScZa 00]	[Gopa 00]	
	IM-04				
	IM-05				
	IM-06	[BBMR 05]	[BBMR 05]		
an OM	OM-01	[CQM+ 09, VRYK 03b, TaAS 08]	[CQM+ 09, VRYK 03b, TaAS 08]		
	OM-02				
an FM	FM-L01	[Gopa 00, CQM+ 09, MJTS 07, QBRZ 05]	[Gopa 00, CQM+ 09, MJTS 07, QBRZ 05]		
	FM-L02	[CQM+ 09, TaAS 08]			[CQM+ 09]
	FM-L03				
	FM-P01		[ScZa 00, CQM+ 09]	[CQM+ 09]	
	FM-P02			[CQM+ 09]	
	FM-M01	[MJTS 07]	[MJTS 07]	[MJTS 07]	[MJTS 07]
	FM-M02				
	FM-M03				
	FM-R01	[QBRZ 05]			
	FM-R02	[QBRZ 05]			
	FM-R03				
	FM-F01		[MJTS 07]		[MJTS 07]
	FM-F02				
	FM-01		[ScZa 00]		
	FM-02				
an KM	KM-01				
	KM-02				
	KM-03		[ChSa 01]		
NFA	NF-01				
	NF-02				
	NF-03				
	NF-04		[TaAS 08]		
	NF-05		[CQM+ 09, QBRZ 05, TaAS 08]	[CQM+ 09]	
	NF-06	[DBCAF 06]	[HoFa 07, HoFa 08, DBCAF 06]		
	NF-07		[MJTS 07, HoFa 07]	[MJTS 07]	[MJTS 07]
	NF-08				

Tabelle 4.4.: Abdeckung der Anforderungen durch Ansätze der Phase Fehlereingrenzung

der Realisierung des Informations- (IM-02, IM-06) bzw. Organisationsmodell (OM-01, OM-02) in der vorliegenden Arbeit mit einbezogen werden können.

Overlay User Diagnosis (OUD)

In Bereich der Overlay-Netze wird in [TaAS 08] eine Fehlerdiagnosemethode beschrieben, die auf einer kollaborativen Nutzerbeobachtung bzgl. Fehler beruht (Overlay User Diagnosis, OUD). OUD kann Fehler eingrenzen, ohne dass man Bezug nehmen muss auf unterliegende Schichten; das wird durch OverlayMonitoring-Agenten realisiert, die die Nutzerbeobachtungen korrelieren. Für die vorliegende Arbeit kann dieser Ansatz für die Optimierung der Fehlereingrenzung benutzt werden, da die Suche auf unterliegenden Netzschichten sehr zeitaufwendig ist (Performanz (NF-05), Skalierbarkeit (NF-04), Automatisierung(NF-06)). Ebenfalls gibt dieser Ansatz Hilfestellung für die Erstellung des Organisationsmodells (OM-01) durch overlay monitoring agents, die die Nutzerbeobachtungen sammeln und korrelieren um eine Fehlerdiagnose zu stellen. Auch der Workflow der Fehlerlokalisierung in einer Domäne (FM-L02) wird durch das OUD System unterstützt.

Die Bewertung der hier vorgestellten Ansätze ist in Tabelle 4.4 zusammengefasst. Die grau hinterlegten Boxen repräsentieren die relevanten Felder der Tabelle 3.20 des vorherigen Kapitels (Seite 93).

4.3.3. Arbeiten zur Fehlerbehebung

Für die Fehlerbehebung an sich gibt es weniger Literatur, allerdings sind die folgenden Arbeiten aus dem Bereich Recovery, Backup und Wiederherstellung hier erwähnenswert.

Backup-Service-Management System

Ein Backup-Service-Management System wird in [LSMK 99] beschrieben. Lück et al. realisieren anhand einer Client/Server-Architektur einen Backup-Service für die Speicherung und Wiederherstellung von Daten im verteilten Umfeld. Das System besteht aus zwei Teilen: Einer, der die Backups realisiert, und der andere, der für die Wiederherstellung zuständig ist. Dabei werden wichtige Parameter in Betracht gezogen: die Ausfallsicherheit, die Geschwindigkeit, mit der ein Backup realisiert und wiederhergestellt wird, der Automatisierungsgrad als auch die Heterogenität der Umgebung und Skalierbarkeit. Das Modell ist UML-basiert und beschreibt unterschiedliche Bereiche, die implementiert werden (Service Model, System Model und Services Used). Der Aufbau des Systems an sich ist für die in der vorliegenden Arbeit zur entwickelnde Architektur von Belang, insbesondere für das Informationsmodell. Ebenso werden Grundzüge des Organisationsmodells beschrieben mit den Rollen und Verantwortlichkeiten für das Managementsystem und für das gemanagte System. Von den Anforderungen an eine ioFMA werden die Automatisierung (NF-06), Skalierbarkeit (NF-04) und Performanz (NF-05) berührt.

Segment-Recovery

In [AbED 09] wird eine Technik zur Wiederherstellung von Segmenten eines hierarchisch aufgebauten Netzes beschrieben. Der vorgeschlagene Mechanismus basiert auf der Wiederherstellung von Interdomänen-Segmenten und wird von einem Recoverymodul (eines pro Domäne) durchgeführt. Bei der Durchführung des Recovery Prozesses wird auf die Einhaltung von QoS-Parameter geachtet. Durch die Segmentweise Wiederherstellung, also das

Anforderungen		Phasen des Lebenszyklus eines Fehlers			
		Entdecken	Eingrenzen	Beheben	Vorhersage
an IM	IM-01				
	IM-02				
	IM-03				
	IM-04				
	IM-05				
	IM-06				
an OM	OM-01			[AbED 09]	
	OM-02			[AbED 09]	
an FM	FM-L01				
	FM-L02				
	FM-L03				
	FM-P01				
	FM-P02				
	FM-M01				
	FM-M02			[NRS+ 10]	
	FM-M03				
	FM-R01				
	FM-R02				
	FM-R03				
	FM-F01				
	FM-F02				
	FM-01				
	FM-02				
an KM	KM-01			[AbED 09]	
	KM-02			[AbED 09, NRS+ 10]	
	KM-03			[AbED 09]	
NFA	NF-01				
	NF-02				
	NF-03				
	NF-04			[LSMK 99]	
	NF-05			[LSMK 99]	
	NF-06			[LSMK 99]	
	NF-07			[NRS+ 10]	
	NF-08				

Tabelle 4.5.: Abdeckung der Anforderungen durch Ansätze der Phase Fehlerbehebung

Reparieren der fehlerhaften Pfade in der ,,Nähe" des Fehlers, wird eine schnellere Recoveryzeit gewährleistet. Der Recovery-Mechanismus aggregiert Fehlerereignisse, um die ,,Ereignisflut" einzudämmen. Zusätzlich bewertet es die ausgefallenen Pfade, um die Funktion des hoch priorisierten Netzverkehrs als erstes zu gewährleisten. Die hier beschriebene Segmentweise Wiederherstellung ist für die vorliegende Arbeit ein guter Anhaltspunkt im Bereich Organisations- bzw. Kommunikationsmodell.

Disaster-Recovery

Nayak et al. gehen in [NRS+ 10] das Disaster-Recovery an, das über die reine Fehlerwiederherstellung herausgeht. Das implementierte System ENDEAVOUR ist ein Framework für integrierte Ende-zu-Ende Disaster-Recovery-Planung. Das System unterstützt unterschiedliche Technologien bzw. Kombinationen von Technologien über mehrere Schichten hinweg (virtuelle Maschinen, Datenbanken, Storage-Controllers u. a.). Auch wenn die vorliegende Arbeit sich nicht mit Disaster-Recovery befasst, ist es trotzdem aus konzeptionellen Gesichtspunkten (Aufbau des Systems, Interdomänen-Eigenschaft des Szenarios usw.) ein wertvoller Beitrag. Spezifisch für die Anforderungen werden der vorliegenden Arbeit Informationen bzgl. einer gemeinsamen Informationsdatenbank (NF-07) gegeben durch die ENDEAVOUR-Wissensdatenbank, die aus Technologiemodellen und Kompositionsformeln besteht und die insbesondere bei der Replikation in Anspruch genommen wird. Es wird ebenfalls die Interdomänen-Kommunikation (KM-02) angesprochen, durch ein technologieübergreifendes Replikationsmodell. Durch das ENDEAVOUR-Framework wird z. B. auch das interorganisationale Monitoring (FM-M02) berührt.

Tabelle 4.5 fasst Bewertung der Ansätze gegenüber den Anforderungen zusammen. Die grau hinterlegten Boxen repräsentieren die relevanten Felder der Tabelle 3.20 der vorherigen Kapitels (Seite 93).

4.3.4. Arbeiten zur Fehlervorhersage/-vorbeugung

Active Probing

Probing ist die Untersuchung eines Systems auf Fehler, auch wenn diese sich nicht oder noch nicht auswirken. Rish et al. stellen in [RBO+ 04] eine Echtzeit-Problemermittlung anhand Active Probing vor. Es handelt sich hier um aktive Messungen; ,,Proben" sind Transaktionen, die eine (Netz)Komponente auf ihre Funktionsfähigkeit überprüfen. Ein sehr einfaches Beispiel, um die Verfügbarkeit des Netzes zu prüfen, ist Ping. Es gibt die Möglichkeit, diese Tests vorausgeplant und auch periodisch durchzuführen, jedoch ist die Wirsamkeit einer solchen Methode eingeschränkt: Werden keine Tests benötigt, verschwenden vorausgeplante Tests Ressourcen; bei tatsächlich aufgetretenen Problemen treten Verzögerungen bei der Antwort auf und im Fall von nicht hinreichenden Tests müssen zusätzliche Untersuchungen durchgeführt werden. Die Arbeit von Rish et al. stellt eine interaktive Methode zur Durchführung von Tests vor. Diese besteht aus einem Test, um die mögliche Diagnose zu stellen, und einem zweiten Schritt, in dem, wenn nötig, zusätzliche Tests auszuwählen und durchzuführen sind. Somit selektiert und sendet das Active-Probing Proben abhängig vom aktuellen Problem. Ein ähnlicher Ansatz ist in [NaSe 06] vorgestellt,

in dem das Active-Probing zur Unterstützung der Fehlerentdeckung in Netzen benutzt wird.

Für die vorliegende Arbeit ist der Active-Probing Ansatz hauptsächlich wegen der vordefinierten durchzuführenden Messungen entsprechend der Anforderung Standard-Metriken (IM-05) wichtig. Solche Messungen können dann aggregiert (IM-06) und interpretiert werden um einen Fehler innerhalb einer bestimmten Domäne zu signalisieren (FM-M01, FM-L02).

Intelligent-Probing

Ein anderer Ansatz im Bereich Probing ist das in [KiHo 04] vorgestellte Intelligent-Probing für die Diagnose des Netzstatus. Kirmani und Hood stellen eine Probing-Methode basierend auf einer proaktiven Strategie vor. Fehler müssen in diesem Fall bereits in der Entwicklungsphase gefunden werden, so dass diese präventiv korrigiert werden können. Der Netzstatus wird ermittelt durch das Senden von Proben ins Netz. Wenn die Komponenten erfolgreich antworten, dann sind sie funktional, ansonsten gibt es auf dem untersuchten Pfad mindestens einen ausgefallenen Knoten. Man kann daher schnellstmöglich vorhersagen, welche Kunden bzw. Netzbereiche betroffen werden. Dieser Ansatz wird beim Entwurf der ioFMA miteinbezogen, insbes. für die Erfüllung der Anforderungen: Reporting (FM-R01, FM-R02) und intraorganisationales Monitoring (FM-M01).

Fehlern in Logdateien

Zusätzlich zu Probing beruht eine andere Variante zur Fehlervorhersage und-vorbeugung auf der Identifizierung von Symptomen wiederkehrender Fehler in Logdateien. In den Arbeiten [RMJW 09] und [RMJW 10] wird eine solche Methode beschrieben. Der Ansatz von Reidemeister et al. beinhaltet zwei Teile: Die lernende Phase, in der man die gekennzeichneten Stichproben nimmt und sie analysiert. Die Beobachtungsphase benutzt den trainierten Klassifikator der vorherigen Phase, um mögliche Klassen von Fehlern vorherzusagen. Diese Arbeit ist ein sehr wichtiger Beitrag zur Thema Fehlervorhersage und -vorbeugung im Allgemeinen, aber nicht spezifisch für einen bestimmte Anforderung der vorliegenden Arbeit.

In Tabelle 4.6 wird die Abdeckung der Anforderungen durch die hier beschriebenen Ansätze zusammengefasst.

4.4. Arbeiten zur Dienstkomposition

Gemeinsam erbrachte Dienste im interorganisationalen Umfeld setzten sich zumeist aus mehreren Teildiensten (Provider-spezifisch, Netz-spezifisch, Technologie-spezifisch usw.) zusammen. Hauptsächlich im Bereich der Webservices wird das Thema Dienstkomposition diskutiert und untersucht. Auch für den Bereich der Computernetze existieren Ansätze, die im Kontext dieser Arbeit nennenswert sind.

In ihrer Arbeit [VPM$^+$ 07] zur Dienstkomposition, angewandt auf das Netzmanagement, zeigen Vianna et al., dass die Dienstkomposition tatsächlich auch mit traditionellen Managementtechnologien realisiert werden kann. Die Anwendung der Technologien auf die Dienst-

Anforderungen		Phasen des Lebenszyklus eines Fehlers			
		Entdecken	Eingrenzen	Beheben	Vorhersage
an IM	IM-01				
	IM-02				
	IM-03				
	IM-04				
	IM-05				[RBO+ 04, NaSe 06]
	IM-06				[RBO+ 04, NaSe 06]
an OM	OM-01				
	OM-02				
an FM	FM-L01				
	FM-L02				[RBO+ 04, NaSe 06]
	FM-L03				
	FM-P01				
	FM-P02				
	FM-M01				[KiHo 04, RBO+ 04, NaSe 06]
	FM-M02				
	FM-M03				
	FM-R01				[KiHo 04]
	FM-R02				[KiHo 04]
	FM-R03				
	FM-F01				
	FM-F02				
	FM-01				
	FM-02				
an KM	KM-01				
	KM-02				
	KM-03				
NFA	NF-01				
	NF-02				
	NF-03				
	NF-04				
	NF-05				
	NF-06				
	NF-07				
	NF-08				

Tabelle 4.6.: Abdeckung der Anforderungen durch Ansätze der Phase Fehlervorhersage

komposition ist ein wichtiger Pluspunkt für den Netzmanagement-Bereich. In diesem Ansatz werden allerdings nur Dienste betrachtet, die auf einer hierarchischen Kompositionskette beruhen.

Klie et al. analysieren in [KGF 07] die automatische Komposition von Web-Services als eine zukünftige Möglichkeit zur Automatisierung des Netzmanagements. Basierend auf dem Vergleich einer Reihe von Kompositionstechnologien wird ein Ansatz für die Dienstkomposition im Netzmanagement beschrieben. Diese automatische Webservice-Komposition kann benutzt werden, um die komplexen Netzmanagement-Aktivitäten zu vereinfachen. Dieser Ansatz ist relevant für die vorliegende Arbeit, insbesondere bei der Implementierung der ioFMA.

In [MiTs 10] realisiert Miyazawa et al. basierend auf dem Informations-Framework (SID) ein Integrated Resource Information Model, das nicht nur Informationen über die Netzressourcen umfasst, sondern auch die Applikationsressourcen und Kundenressourcen. Anhand dieses Modells wird anschließend ein Mechanismus zur Fehlerlokalisierung definiert, dessen Funktionalität auf den Auswirkungen auf das Netz basiert.

4.5. Zusammenfassende Bewertung

In diesem Kapitel wurde der Stand der Technik im Bereich des. Fehlermanagements beschrieben und folgende Gruppierung vorgenommen: ITSM, Fehlermanagement allgemein (Fehlerentdeckung, Fehlereingrenzung, Fehlerbehebung und Fehlervorhersage bzw. -vorbeugung), interorganisationales Fehlermanagement sowie Dienstkomposition. Die Arbeiten wurden im jeweiligen Abschnitt zusammengefasst.

Als Gesamtbewertung sind einige Beobachtungen aufzuführen: Es gibt eine Reihe von Arbeiten, die sich mit Fehlermanagement auseinandersetzen, allerdings, wie schon in den vorherigen Abschnitten erwähnt, sind diese auf bestimmte Probleme fokussiert. Entweder gehen sie hauptsächlich nur auf ein oder zwei Phasen des Fehlerlebenszyklus ein oder nur auf eines der Managementmodelle bzw. auf ausgewählte Bereiche davon. Ebenfalls zu bemerken ist, dass viele dieser Arbeiten einen guten Anhaltspunkt für die Entwicklung einer ioFMA bieten, allerdings decken sie nur teilweise die in der vorliegende Arbeit angesprochene Problematik ab. Einige Beispiele dafür:

- Die perfSONAR-Architektur ist eine mehrschichtige Architektur, die für interorganisationales Performancemonitoring eingesetzt wird. Sie ist ein hilfreicher Ansatz für das in der Systemsicht zu entwickelnde plattformabhängige Modell. Sie stellt aber nur wenig Informationen für das plattformunabhängige Modell dar.
- Das E2EMon Tool ist einer der wertvollsten Ansätze im Bereich interorganisationales Monitoring mit gut beschriebenen Kommunikationsmechanismen, Aggregationsfunktionen, Statusinformationen usw. Der Nachteil ist, dass dieses System in Umgebungen

mit einer hierarchischen Form der Diensterbringung nur begrenzt eingesetzt werden kann.

- In dem Ansatz von Steider und Sethi werden sehr gute organisatorische Rahmen (bzgl. Rollen und Verantwortlichkeiten) für das interorganisationale Fehlermanagement aufgezeigt. Allerdings sind diese nur für hierarchische Formen der Diensterbringung beschrieben und instantiiert.
- Die ITSM-Ansätze sind für den Modellentwurf grundlegend; die Existenz detailliert beschriebener Prozesse bietet die Möglichkeit, darauf eine ioFMA aufzubauen. Die Tatsache, dass klar definierte Prozesse existieren, vereinfacht den Modellentwurf (siehe Abschnitt 2.3.2), da diese das CIM-Modell des in diese Arbeit gefolgten MDA-Modellierungsansatzes ausmachen. Diese Prozesse geben auch Informationen über die möglichen Rollen und Verantwortlichkeiten; es werden allerdings keine Informationen bzgl. des Kommunikationsmodells bereitgestellt.
- Aus der Wissenschaft und Forschung beschäftigen sich einige Ansätze speziell mit der Automatisierung des Incident-Management-Prozesses. Es werden dabei auch nichtfunktionale Aspekte betrachtet, die bei der Entwicklung der ioFMA zu berücksichtigen sind. Der Nachteil ist hier, dass diese Ansätze intraorganisational untersucht worden sind, eine Untersuchung auf deren Verallgemeinerung für interorganisationales Fehlermanagement steht aus.
- Einige Ansätze behandeln probabilistische Fehlerentdeckung und -diagnose. Diese sind grundsätzlich in Betracht zu ziehen, da sie sowohl auf wahrscheinliche Fehlerursachen in einem System hinweisen als auch auf das gemeinsame Auftreten von Fehlern. Dieses ist allerdings nur intraorganisational definiert. In interorganisationalen Umgebungen mit viel höherer Komplexität kommen weitere organisatorische und kommunikative Aspekte hinzu.

Was fehlt, ist ein Ansatz, der größtenteils die in Tabelle 3.20 gestellten Anforderungen gleichzeitig erfüllt. Alle oben beschriebenen Aspekte können miteinander kombiniert werden, um die aufgezeigten Defizite im Bereich des interorganisationalen Fehlermanagements zu überwinden. Das Ziel dieser Arbeit ist hiermit, eine übergreifende Lösung bzgl. Fehlermanagement zu erstellen, die mehr ist als eine Summe der schon existierenden Ansätze.

5

Entwurf einer interorganisationalen Fehlermanagementarchitektur (ioFMA)

Inhalt des Kapitels

5.1. Entwurfssystematik

Wie schon in Kapitel 2.3 kurz angeschnitten, folgt der Entwurf der interorganisationalen Fehlermanagementarchitektur der in der Softwareentwicklung benutzten Model Driven Architecture- Entwicklungsstrategie. Die MDA wurde im Jahr 2001 von der OMG verabschiedet. Abbildung 2.12 aus Abschnitt 2.3.2 auf Seite 35 zeigt den Zusammenhang zwischen dem Entwurf (bzw. der Anforderungsanalyse und den verwandten Arbeiten) in der vorliegenden Arbeit und dem MDA-Ansatz. Dies wird in diesem Abschnitt ausführlich beschrieben. Zur Umwandlung eines Modells in ein anderes werden Transformationen benötigt [Schif 07, Scha 08]. Somit besteht die Entwurfssystematik in gemäß MDA in einer Beschreibung von Modellen und deren Transformationen.

5.1.1. Computation Independent Model (CIM)

Als berechnungsunabhängiges Modell beschreibt das CIM ein System allgemein und dessen Umgebung. Der Fokus wird auf eine möglichst vollständige Beschreibung des Kontextes gelegt [GPR 06]. Dies beinhaltet ein sogenanntes Domänenmodell (Domain Model) und ein Anforderungsmodell (Requirements Modell). Auf der Ebene der berechnungsunabhängigen Modellierung beinhaltet das Modell fast ausschließlich deklarative, technologieneutrale Beschreibungen.

Domänenmodell

Für das Domänenmodell wird möglichst eine Modellierungssprache die dem Verständnis der Personen, die täglich mit dem System zu tun haben entspricht, benutzt. Dies dient in der Softwareentwicklung zur verständlicheren Kommunikation zwischen Entwicklern und Kunden. Eine angemessene Abbildung fachspezifischer Konzepte (seitens des Kunden) auf Konzepte der technischen Ebene (seitens der Entwickler) zu finden, ist dabei der entscheidende Punkt. Es bietet sich die UML als gemeinsame Sprache an. Wenn das System, das zu modellieren ist, aus dem geschäftlichen Umfeld stammt, wird das Domänenmodell Business Model genannt. In der vorliegenden Arbeit ist dieses Modell durch den Prozess-Workflow des Incident-Managements in ITIL und dem Referenzprozess Incident Management [Hamm 09] gegeben.

Anforderungsmodell

Das Anforderungsmodell bringt, zusätzlich zum Domänenmodell, Informationen hinsichtlich der zu erstellenden Anwendung wie auch der Umgebung und der Prozesse, die sie benötigt. Das Anforderungsmodell kann als Untermodell des Domänenmodells betrachtet werden, beschreibt aber detaillierter den Sachbestand der zu entwickelnden Anwendung. Das Anforderungsmodell korrespondiert in der vorliegenden Arbeit mit den in Abschnitt 3.2.2 beschriebenen Anwendungsfällen und der von ihnen abgeleiteten Anforderungen in Abschnitt 3.2.3.

5.1.2. Platform Independent Model (PIM)

Das plattformunabgängige Modell (PIM) der MDA beschreibt formal die Struktur und Funktionalität eines Systems. Dieses ist unabhängig von den benutzten Implementierungstechniken und abstrahiert von der zugrundeliegenden Plattform.

Eine Plattform aus Sicht der MDA ist eine Ausführungsumgebung für die Anwendung. Ihre Funktionalität wird anhand von Schnittstellen für die Anwendungen zur Verfügung gestellt. Wie in der von der Object Management Group (OMG) herausgegebenen MDA-Spezifikation ([OMG 03]) beschrieben, gibt es drei Kategorien von Plattformen:

- Generisch: Umfasst logische, technologieneutrale Konzepte (z. B. Objektorientierte Programmierung).
- Technologiespezifisch: Repräsentiert die Umsetzung von Konzepten innerhalb einer Technologie bzw. einen Standard (z. B. CORBA, JavaEE).
- Herstellergebunden: Beschreibt die technologiespezifischen Konzepte eines Herstellers (z. B. IBM WebShpere, IONAOrbix)

Das PIM gemäß MDA ist unabhängig von technologiespezifischen und herstellergebundenen Plattformen. Das PIM ist aber nicht völlig plattformunabhängig, wenn man die generischen Plattformen in Betracht zieht. Das PIM wird meistens mit der UML realisiert, was auf dem Konzept der Objektorientierung beruht. Dies muss aber keine einschränkende Wirkung auf die spätere Realisierung des Systems haben.

Ein sehr wichtiger Aspekt des PIMs ist der Domänenbegriff. Domänen werden als klar abgrenzbare, kohärente fachliche Bereiche eines Systems definiert. Die Domänenbildung erfolgt in der vorliegende Arbeit in Abschnitt 5.2.1.

Nach der Spezifikation der Domänen wird zur Beschreibung eines Metamodells einer Architecture Metamodel (AMM) übergegangen. Das AMM wird als Hilfestellung für die Transformation des CIM in das PIM benötigt. Dieses Metamodell beinhaltet die Beschreibung der wichtigsten Konzepte, Abstraktionen, Schnittstellen und Muster, die angewendet werden, um ein System zu modellieren. Es trägt zur Realisierung des PIMs bei, indem es die notwendigen Modellierungssprachen zur Darstellung einzelner Aspekte oder Aspektklassen angibt. In dieser Arbeit wird mit Hilfe von UML erst das Funktionsmodell (mit den Funktionsbereichen und Managementfunktionen) in Abschnitt 5.2.2 und anschließend das Informationsmodell (mit allen in der ioFMA involvierten Klassen) in Abschnitt 5.2.3 entworfen.

5.1.3. Platform Specific Model (PSM)

Durch Erweiterung des Platform Independent Model (PIM) mit plattformabhängigen Informationen erhält man das Platform Specific Model (PSM).

Die Transformation vom Platform Independent Model (PIM) in das Platform Specific Model (PSM) wird mit der Unterstützung des Platform Description Model (PDM) realisiert. Das PDM besteht aus der Erweiterung der bisher betrachteten Architekturkonzepte mit domänenspezifischen Aspekten und konkreten Realisierungsmöglichkeiten. Die Technologien, die dabei benutzt werden, werden hier unter dem Begriff Plattform verwendet. Auf dieser Plattform wird dann letztendlich die entstandene Anwendung ausgeführt. Wie beim Übergang vom Computation Independent Model (CIM) zum Platform Independent Model (PIM) werden auch hier Regeln zur Überführung benötigt. Dafür wird das Platform Description Model (PDM) eingesetzt, indem die auf dem Architecture Metamodel (AMM) erstellten UML-Profile mit technologiespezifischen Informationen angereichert werden.

5.2. Entwurf der Teilmodelle

Die Realisierung der Teilmodelle der ioFMA-PIM beruht auf den Anwendungsfällen und deren beteiligten Akteure als auch auf den Referenzprozesse in ITIL und [Hamm 09] die das CIM gemäß MDA repräsentieren. Da sich diese Arbeit mit dem interorganisationalem Fehlermanagement beschäftigt, ist es naheliegend, als erstes das Organisationsmodell zu entwerfen (Abschnitt 5.2.1). Auf diesem und auf der Basis der in Abschnitt 3.2.2 beschriebenen Anwendungsfälle wird das Funktionsmodell angefertigt (Abschnitt 5.2.2), gefolgt vom Informationsmodell im Abschnitt 5.2.3. Abschließend wird das Kommunikationsmodell bereitgestellt (Abschnitt 5.2.4).

5.2.1. Das Organisationsmodell

In Abschnitt 3.2.1 wurden die Akteure, die an den Anwendungsfällen beteiligt sind, genannt und kurz beschrieben. In diesem Abschnitt werden die Rollen, die am ioFM-Prozess beteiligt sind, beschrieben, basierend auf den in der Anforderungsanalyse beschriebenen Akteuren.

5.2.1.1. Definition der Managementdomänen

Aus der Beschreibung der Anwendungsfälle und den daraus gefolgerten Anforderungen ergibt sich die Notwendigkeit der Einteilung in Managementdomänen. Bei jedem Provider bestehen bereits intraorganisationale Fehlermanagement-Prozesse. Das ist grundlegend für diesen Ansatz, da die dafür zuständigen Personen oder Aktivitäten mit dem auf der interorganisationalen Ebene durchgeführten Fehlermanagement interagieren. Zudem kann man den Nutzer-Bereich separat als Managementdomäne betrachten. Insgesamt werden drei Managementdomänen betrachtet:

- **CustDomain:** Die Customer-Managementdomäne `CustDomain` ist die Nutzer-Domäne gemäß dem MNM-Dienstmodel [GHH+ 01, GHK+ 01].
- **SPDomain:** Die Service-Provider-Managementdomäne `SPDomain` repräsentiert die lokale Domäne eines Service Providers, der an der Diensterbringung beteiligt ist.
- **ioDomain:** Die interorganisationale Managementdomäne `ioDomain` ist diejenige Domäne, die die interorganisationalen Sachverhalte managt. Diese ist sehr eng mit den andern zwei integriert.

5.2.1.2. Definition der Rollen

Die ioFMA-Rollen basieren auf den in Abschnitt 3.2.1 definierten Akteuren. Dort wurden die notwendigen Akteure definiert, die an den allgemeinen Anwendungsfällen beteiligt sind. In diesem Abschnitt wird die Liste der partizipierenden Rollen erweitert und in die entsprechenden Managementdomänen eingeteilt.

Rollen in der SPDomain Schon in Abschnitt 3.2.1 wurden für die Provider-Domäne die Rollen Service Provider (SP), Domain-Fault-Manager (DFM), Domain-Fault-Operator (DFO) und Domain-Monitoring-System (DMS) eingeführt. In der Beschreibung des Fehlermanagementprozesses für das verallgemeinerte Szenario in Abschnitt 3.1.4.3 gab es zwei sehr allgemeine, übergreifende Rollen: Domain-Fault-Reporting-Management-Team (DFRMT) und Domain-Fault-Resolution-Team (DFRT). Wenn man diese Informationen und die Beschreibung der Anwendungsfälle zusammenfasst, ergeben sich folgende Rollen in der Provider-Domäne.

Domain Fault Manager (Dom-FM) ist diejenige Rolle, die für Koordination und Überwachung des Fehlermanagementprozesses innerhalb einer organisatorischen Domäne verantwortlich ist. Er begleitet den Fehlerlebenszyklus in allen seinen Phasen, von der Fehlerentdeckung bis zum -beheben und zur -vorhersage. Sie koordiniert die anderen intraorganisationalen Rollen (bzgl. Fehlermanagement). Ebenfalls ist sie die Kontaktperson zu anderen Managementprozessen (z. B. Konfigurations-, Change-, Accounting-Management).

Der Akteur Domain-Fault-Operator (DFO) wird hier durch zwei Rollen realisiert: Domain Service Desk (Dom-SD) und Domain Technical Specialist (Dom-TS). Die Rolle Dom-SD ist für die Aufnahme von Störungen innerhalb einer organisatorischen Domäne und die Weiterleitung zu den Dom-TS zuständig. Ebenfalls sehr wichtig für das ioFM ist die Tatsache, dass diese Rolle sowohl Störungen von anderen Domänen annehmen als auch die in dieser Domäne nicht gelösten Störungen zu anderen Domänen unter der Überwachung des Dom-FM weitergeben kann.

Der Domain Technical Specialist (Dom-TS) bekommt vom Dom-SD eine Fehlermeldung zur Lösung weitergeleitet. Falls der Fehler in der eigenen Domänen nicht zu lösen ist, wird dieser

Rolle *Domain Fault Manager*	
Bezeichner	Dom-FM
Mgmt.Domäne	SPDomain
Beschreibung	Überwachung des lokalen Fehlermanagementprozesses
Assoziierte Akteure	DFM

Tabelle 5.1.: Zusammenfassung der Rolle *Domain Fault Manager*

Rolle *Domain Service Desk*	
Bezeichner	Dom-SD
Mgmt.Domäne	SPDomain
Beschreibung	Störungen innerhalb der Domäne annehmen und weiterleiten, intern zum Dom-TS oder nach Außen, wenn nicht lösbar
Assoziierte Akteure	DFM, DFO

Tabelle 5.2.: Zusammenfassung der Rolle *Domain Service Desk*

wieder zurück zum Dom-SD gegeben mit dem Hinweis auf Weiterleitung entweder zu einer bestimmten Domäne oder zum io-SD, der es dann zu der für den Fehler zuständigen Domäne weiterleitet.

Rolle *Domain Technical Specialist*	
Bezeichner	Dom-TS
Mgmt.Domäne	SPDomain
Beschreibung	Störungen innerhalb der Domäne lösen oder weiter eskalieren
Assoziierte Akteure	DFO

Tabelle 5.3.: Zusammenfassung der Rolle *Domain Technical Specialist*

Das Domain Monitoring System (Dom-MS) realisiert innerhalb einer Domäne die Überwachung der Systeme. Durch Meldungen oder Alarme wird das Fehlverhalten des Systems angekündigt. Das Dom-MS ist ein sehr nützliches und unerlässliches Werkzeug für die frühzeitige Fehlererkennung, aber auch für die anderen Phasen des Fehlerlebenszyklus.

Rolle *Domain Monitoring System*	
Bezeichner	Dom-MS
Mgmt.Domäne	SPDomain
Beschreibung	Überwachung des Systems und Meldung von Systemfehlern und Alarmen
Assoziierte Akteure	DMS

Tabelle 5.4.: Zusammenfassung der Rolle *Domain Monitoring System*

Rollen in ioDomain Bei der Beschreibung der Akteure in Abschnitt 3.2.1 ist für das interorganisationale Fehlermanagement ein einziger Akteur definiert worden Global-Fault-Coordinator-Manager (GFCM). Im Abschnitt 3.1.4.3 wurde bei der Beschreibung des ioFM für das verallgemeinerte Szenario die übergreifende Rolle Global-Fault-Management-Team (GFMT) benutzt. Dieser Akteur wird in drei unterschiedliche spezialisierte Rollen eingeteilt: Interorganizational Fault Manager (io-FM), Interorganizational Service Desk (io-SD) und Interorganizational Fault Operator (io-FO).

Der Interorganizational Fault Manager (io-FM) ist diejenige Rolle, die auf interorganisationaler Ebene die Überwachung des Fehlermanagement-Prozesses übernimmt. Sie kontrolliert die Weiterleitung von Störungsmeldungen zu den beteiligten Domänen und überwacht den rechtmäßigen Ablauf des Fehlermanagementprozesses. Diese Rolle kann statisch oder dynamisch vergeben werden. Dynamische Vergabe heißt, dass diese Rolle durch keine bestimmte festgelegte Domäne übernommen wird, sondern jede der beteiligten Domänen diese Rolle übernehmen könnte.

Rolle *Interorganisational Fault Manager*	
Bezeichner	io-FM
Mgmt.Domäne	ioDomain
Beschreibung	Koordination, Überwachung des io-Fehlermanagement-Prozesses
Assoziierte Akteure	GFCM

Tabelle 5.5.: Zusammenfassung der Rolle *Interorganisational Fault Manager*

Der Interorganizational Service Desk (io-SD) ist für die Aufnahme der Störungen auf interorganisationaler Ebene zuständig: Selbst wenn mehrere Provider an der Diensterbringung beteiligt sind, gibt es ,,nach Außen" für den Kunden oder für die anderen Domänen eine einzige

Instanz, an die aufgetretene Störungen signalisiert werden. In Absprache mit dem io-FO werden Fehlermeldungen zu den betroffenen Domänen weitergeleitet. Diese Rolle kann sowohl statisch als auch dynamisch vergeben werden wie im vorherigen Fall.

Rolle *Interorganisational Service Desk*	
Bezeichner	io-SD
Mgmt.Domäne	ioDomain
Beschreibung	Aufnahme der Störungen von Kunden oder von anderen Domänen
Assoziierte Akteure	GFCM

Tabelle 5.6.: Zusammenfassung der Rolle *Interorganisational Service Desk*

Die Rolle Interorganizational Fault Operator (io-FO) ist für die Analyse der vom io-SD überreichten Fehlermeldungen zuständig. Diese Analyse kann schließlich zur Findung der Fehlerursache und Lösung des Fehlers führen oder es wird entschieden, die Fehlermeldung zu einer bestimmten anderen Domäne weiterzuleiten. Sie ist diejenige, die das Know-how bezüglich der technischen Gegebenheiten auf interorganisationalen Ebene besitzt, also kann sie auch entscheiden, welchen Domänen die bestehende Fehlermeldung zuzuordnen ist.

Rolle *Interorganisational Fault Operator*	
Bezeichner	io-FO
Mgmt.Domäne	ioDomain
Beschreibung	Analyse von Fehlern, Lösung und evtl. deren Weiterleitung zu den beteiligten (organisatorischen) Domänen
Assoziierte Akteure	GFCM

Tabelle 5.7.: Zusammenfassung der Rolle *Interorganisational Fault Operator*

Das Interorganizational Monitoring System (io-MS) ist für das interorganisationale Fehlermanagement sehr wichtig, da es Systemmeldungen bezüglich der interorganisational erbrachten Dienste liefert. In der Realität kann diese Rolle von einem tatsächlich existierenden interorganisationalen Monitoringtool, das Informationen über den Zustand der unterschiedlichen Domänen liefert, repräsentiert werden. Ebenso könnte es sich dabei aber auch um ein Tool handeln, das mit Informationen aus den lokalen Monitoring-Systemen beliefert wird, diese dann zusammenfasst und korreliert, um den Zustand des Gesamtsystems zu ermitteln.

Rolle *Interorganisational Monitoring System*	
Bezeichner	io-MS
Mgmt.Domäne	ioDomain
Beschreibung	Überwachung und Meldung von Systemfehlern, Alarme im io-Umfeld
Assoziierte Akteure	DMS

Tabelle 5.8.: Zusammenfassung der Rolle *Interorganisational Monitoring System*

Die unterschiedlichen Rollen können von unterschiedlichen administrativen Domänen übernommen werden. Auch kann eine Person oder ein Team mehrere Rollen übernehmen, die sowohl intra- als auch interorganisational sein können. Beispielsweise kann der Dom-SD der Domäne A für kurze Zeit die Rolle der io-SD übernehmen, während für die gleiche Zeitspanne der Dom-FM der Domäne B der io-FM ist.

Rollen in CustDomain Das hier benutzte Konzept Customer ist dem MNM-Dienstmodell von Garschammer et al. [GHH+ 01, GHK+ 01] entnommen. Dieses differenziert auf Seite des Kunden zwischen der Rolle des tatsachlichen Nutzers und der Rolle des Customers. Der Nutzer (`user`) ist derjenige, der den Dienst tatsächlich benutzt und somit auch mögliche Störungen in den normalen Betrieb des Dienstes bemerken kann und diese beispielsweise zum Service Desk meldet. Der Customer hingegen übernimmt die Managementaufgaben

Rolle *Nutzer*	
Bezeichner	user
Mgmt.Domäne	CustDomain
Beschreibung	Der tatsächliche Nutzer des Dienstes
Assoziierte Akteure	user

Tabelle 5.9.: Zusammenfassung der Rolle *Nutzer*

seitens des Kunden. Über eine CSM-Schnittstelle (vgl. [Nerb01]) kann er die Schnittstelle der Managementfunktionen des Service-Providers ansprechen.

Rolle *Customer*	
Bezeichner	customer
Mgmt.Domäne	CustDomain
Beschreibung	Führt alle Managementaktivitäten bzgl. des Dienstes auf der Kundenseite
Assoziierte Akteure	user

Tabelle 5.10.: Zusammenfassung der Rolle *Customer*

5.2.1.3. Interaktion von Rollen

Nachdem im vorherigen Abschnitt die Rollen der ioFMA definiert und beschrieben wurden, gibt dieser Abschnitt eine Übersicht über die Interaktionen zwischen den Rollen. Es werden dabei Interaktionskanäle definiert, als Schnittstelle zwischen Rollen beschrieben, die miteinander in irgendeiner Form kommunizieren.

Abbildung 5.1 gibt eine Übersicht über die Domänen, Rollen und Interaktionskanäle. Da der Kunde/Nutzer derjenige ist, der den Dienst beansprucht, wird die `CustDomain`-Domäne zentral, über den Domänen `SPDomain` und `ioDomain` angeordnet. Innerhalb jeder Domäne – repräsentiert als abgerundetes Rechteck – befinden sich die dazugehörigen Rollen (als Ellipsen dargestellt). Die Rollen sind miteinander durch Linien verbunden, die die Interaktionskanäle repräsentieren. Die Namensgebung der Interaktionskanäle ergibt sich durch eine Kombination aus zwei Buchstaben (für die beiden verbundenen Rollen) und eine Zahl (als Zuordnung, falls mehr als eine Buchstabenkombination existiert). Die Buchstaben sind vergeben abhängig von der Domäne, der die Rolle angehört: **P** für Rollen in `SPDomain`, **C** für Rollen in `CustDomain` und **I** für Rollen in `ioDomain`. Beispielsweise ist der Interaktionskanal zwischen customer (Domäne `CustDomain`) und Dom-FM (`SPDomain`) **CP** genannt. Die Darstellung der Rollen innerhalb der Domänen hat keine bestimmte Bedeutung, wurde aber so gewählt um die Übersichtlichkeit der Abbildung zu bewahren.

Die Interaktionskanäle werden in Rahmen verschiedener Funktionalitäten angeboten, die aber erst in Abschnitt 5.2.2 genauer beschrieben werden. Hier werden sie lediglich kurz definiert und die Funktionalitäten bzw. Kommunikationsmechanismen werden lediglich angeschnitten.

Interaktionskanäle CP und CI2. Über die Interaktionskanäle CP und CI2 stellt der Customer in seiner Management-spezifischen Rolle auf der Kundenseite dem Domain-Fault-Manager (DFM) bzw. Interorganizational Fault Manager (io-FM) Informationen über Änderungen der SLAs und über andere Managementaktivitäten auf der Kundenseite zur Verfügung.

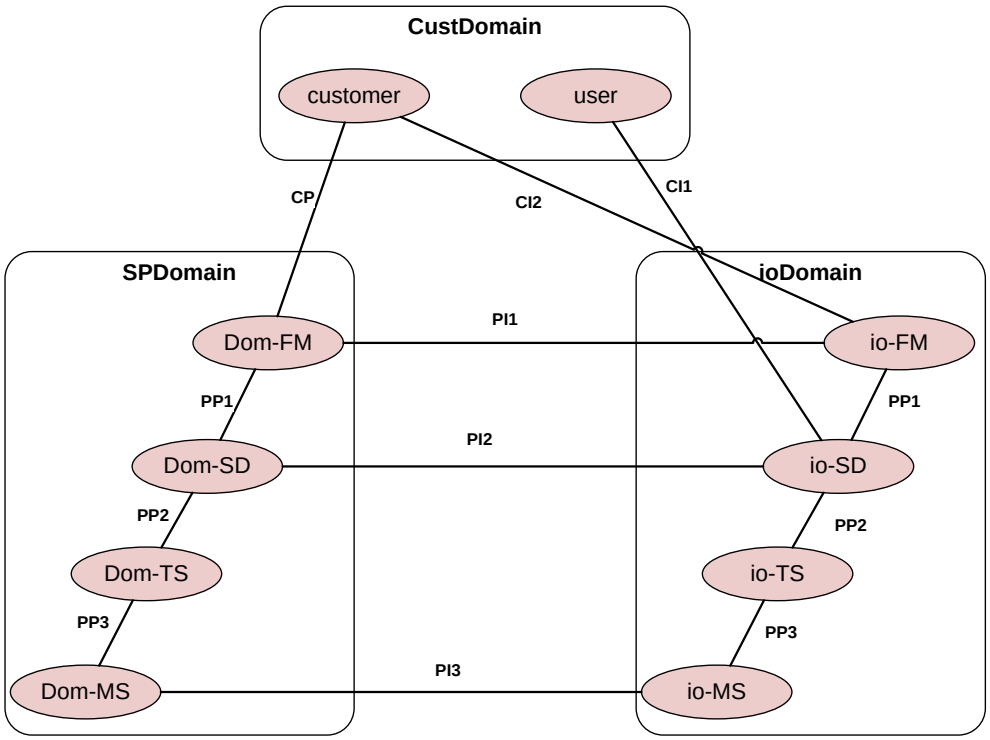


Abbildung 5.1.: Die Interaktionen zwischen den Rollen der ioFMA

Interaktionskanal CI1 Über den Interaktionskanal CI1 meldet der User dem Interorganizational Service Desk (io-SD) Störungen in seinem Dienst oder ändert Informationen bezüglich einzelner Störungen. In der anderen Richtung benachrichtigt der io-SD den User über den Status der Fehlerbearbeitung, bzw. beliefert ihn mit statistischen Daten, QoS-Parameter usw.

Interaktionskanal PP1 Über den Interaktionskanal PP1 werden Informationen bezüglich des Fortschritts in der Fehlerbearbeitung bzw. in der Wartung vom Domain Service Desk (Dom-SD) zum Domain Fault Manager (Dom-FM) transportiert wie auch Meldungen über Änderungen innerhalb dieser organisatorischen Domäne wie beispielsweise geplante Wartungen, Netzzustand usw. In der entgegengesetzten Richtung stellt der Domain Fault Manager (Dom-FM) Informationen über mögliche vertragliche Änderungen gegenüber den Kunden oder interorganisational bedingte Änderungen zur Verfügung. Ebenso fordert er vom Domain Service Desk (Dom-SD) regelmäßige Aktualisierungen des Zustands an.

Interaktionskanal PP2 Über den Interaktionskanal PP2 stellt der Dom-SD dem Domain Technical Specialist (Dom-TS) alle Informationen über eine vorliegende Störung zur Verfügung. Der letztere informiert in der entgegengesetzten Richtung den Dom-SD über die mögliche Lösung oder Weiterleitung der Fehlermeldung.

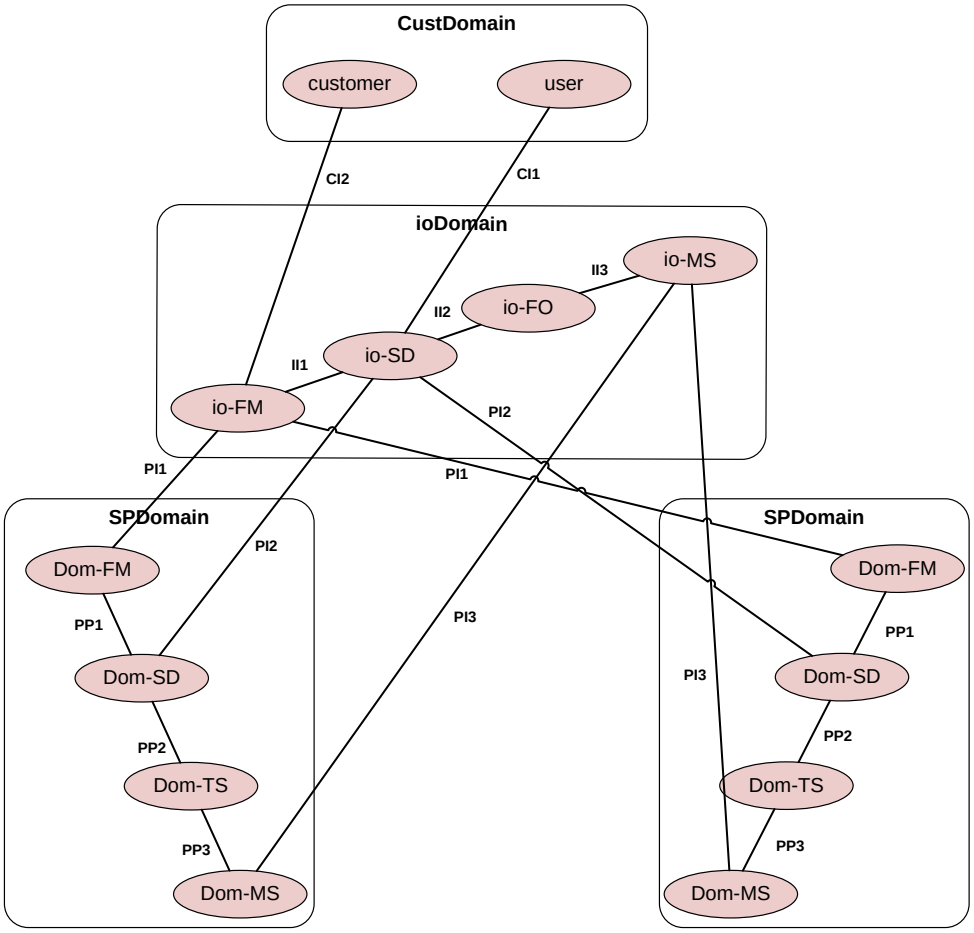


Abbildung 5.2.: Die Interaktionen zwischen den Rollen der ioFMA für die hierarchische Form der Diensterbringung

Interaktionskanal PP3 Über diesen Interaktionskanal PP3 ist es für den Dom-TS möglich, Informationen vom Domain Monitoring System (Dom-MS) zu beziehen, oder (als mögliche Variante) kann das Monitoringsystem proaktiv dem Dom-TS Monitoringdaten zur Verfügung stellen.

Interaktionskanal PI1 Über den Interaktionskanal PI1 werden in beiden Richtungen Informationen über den Fehlermanagementprozess allgemein bereitgestellt.

Interaktionskanal PI2 Über den Interaktionskanal PI2 leitet der Interorganizational Service Desk (io-SD) Informationen über eine bestehende Fehlermeldung auf interorganisationaler Ebene weiter und fordert von der betroffenen Domäne eine Lösung. In der umgekehrten Richtung liefert der Domain Service Desk (Dom-SD) Informationen über die Lösung oder Nicht-Lösung eines Fehlers.

Interaktionskanal PI3 Über den Interaktionskanal PI3 werden Monitoringdaten aus den organisatorischen Domänen auf interorganisationaler Ebene zur Verfügung gestellt. Dieser Interaktionskanal ist optional, da nicht immer die Domänen diese Daten zur Verfügung stellen können und/oder auf interorganisationaler Ebene ein dediziertes Montoring-System existiert.

Interaktionskanal II1 Über den Interaktionskanal II1 stellt der Interorganizational Fault Manager (io-FM) dem io-SD Informationen über mögliche Management-bezogene Änderungen bereit. Der io-SD beliefert den io-FM mit Informationen beispielsweise über den Zustand der Fehlerbearbeitung, Wartungsarbeiten sowie über mögliche geplante Änderungen in den Domänen o.ä.

Interaktionskanal II2 Über den Interaktionskanal II2 liefert der io-SD die relevanten Informationen bezüglich einer gemeldeten Störung dem Interorganizational Fault Operator (io-FO), der das Know-how besitzt, um zu entscheiden, ob die Störung sofort behoben werden kann oder ob sie einer bestimmten Domäne zugewiesen werden muss. Diese Informationen kehren dann zum io-SD auf demselben Interaktionskanal zurück.

Interaktionskanal II3 Über den Interaktionskanal II3 werden Monitoringdaten über das gesamte interorganisationale Providernetzwerk dem io-FO geliefert, um diesem die Entscheidung, welcher Domäne eine Störung zugewiesen werden kann, zu ermöglichen.

5.2.1.4. Besonderheiten des Organisationsmodells bzgl. Diensterbringungsformen

Die oben genannten Rollen und Interaktionen sind allgemein gültig für beide Grundformen der interorganisationalen Diensterbringung, die in der vorliegenden Arbeit betrachtet werden. Dennoch können manche Rollen spezialisiert werden, um den Anforderungen der verschiedenen Formen der Diensterbringung gerecht zu werden.

In einer Hierarchie werden die Rollen der `ioDomain` immer in der organisatorischen Domäne des ,,Wurzel"-Providers vergeben (vgl. Abbildung 5.3). Diese können gleichzeitig auch Rollen der `SPDomain` sein, wenn der Wurzel-Provider selbst Teile der Dienstfunktionalität erbringt. Die darunterliegenden Provider hingegen besitzen nur `SPDomain`-spezifische Rollen. Die unterschiedlichen `SPDomain`s haben untereinander keine Verbindung. Die Interaktionskanäle laufen immer zwischen `ioDomain` und jeder `SPDomain` separat.

In einer Heterarchie kann man innerhalb einer `SPDomain` die Rolle Dom-SD für die organisatorischen Domänen, die sich an den Enden einer ,,Providerkette" befinden, verfeinern. Diese Domänen werden als leicht veränderte `SPDomain` modelliert, die als `EndSiteDomain` gekennzeichnet ist (vgl. Abbildung 5.3). Im Unterschied zur regulären Domäne besitzt sie eine Rolle End-Site-SD, die der Repräsentant für jede der beiden End-Sites ist. Dies vereinfacht die Kommunikation des Kunden mit dem Providernetzwerk. Somit entstehen hier drei

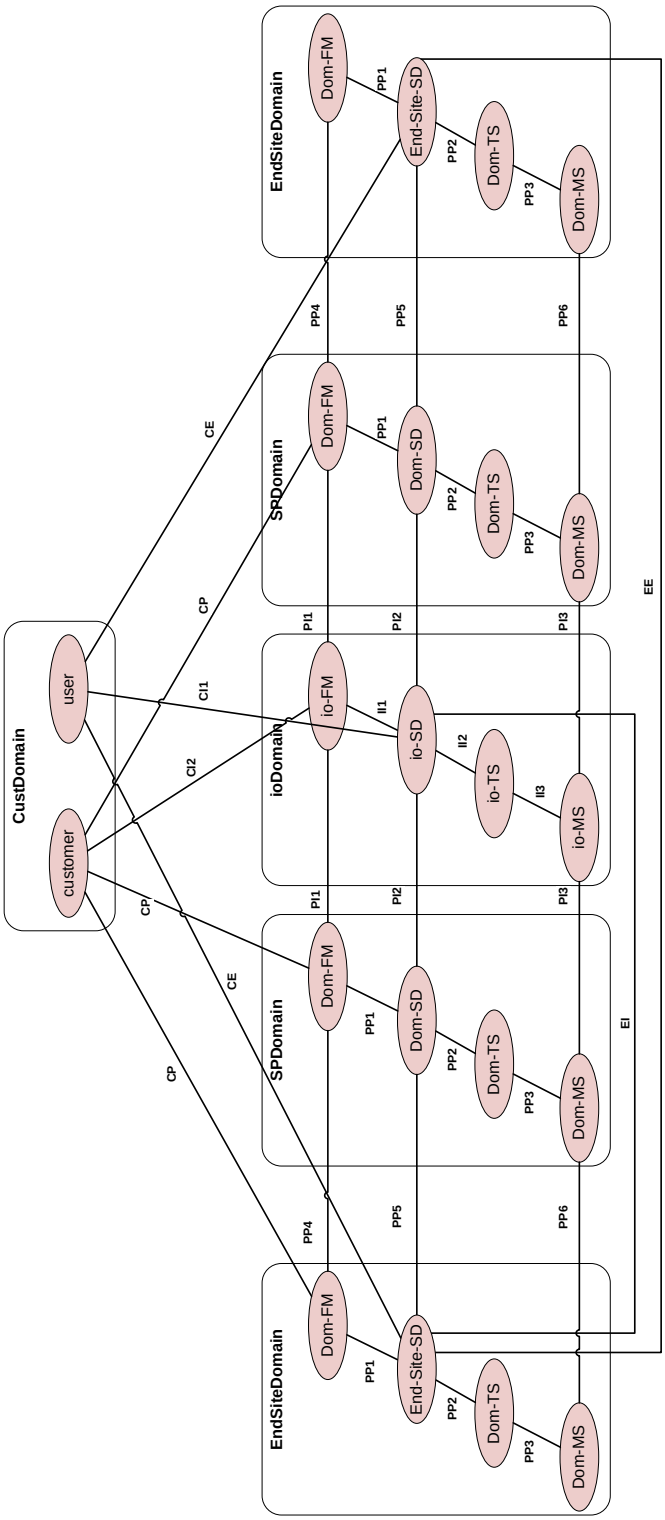


Abbildung 5.3.: Die Interaktionen zwischen den Rollen der ioFMA für die heterarchische Form der Diensterbringung

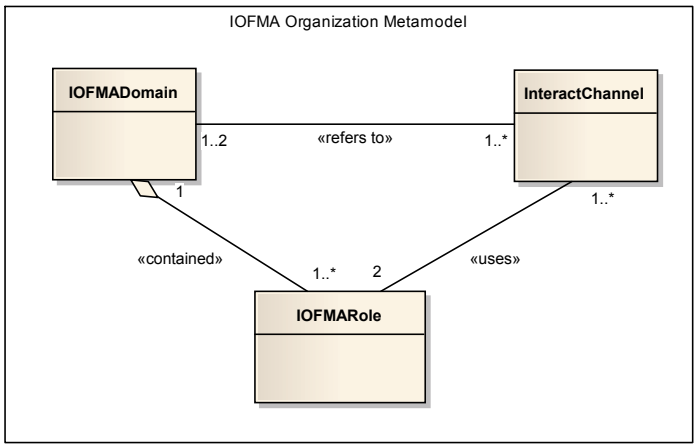


Abbildung 5.4.: Das Metamodell des ioFMA-Organisationsmodells

neue Interaktionskanäle: **CE**, über den die Nutzer mit der jeweiligen End-Site-SD kommunizieren, **EI**, über den Informationen zwischen der End-Site-SD und der `io-SD` ausgetauscht werden, und **EE**, der für die Kommunikation zwischen den zwei End-Site-SDs zuständig ist. Zusätzlich zu den hier beschriebenen Interaktionskanällen, die spezifisch für E2E-Links sind, können noch andere hinzukommen abhängig von den organisatorischen Beziehungen zwischen den Provider.

Im Fall der Heterarchie gibt es zwischen den Domänen (`SPDomain`) und deren benachbarten Domänen eine Verbindung. Wie schon in Abschnitt 2.1.3.4 beschrieben, ist das eine charakteristische Eigenschaft der heterarchischen Form der Diensterbringung. In diesem Fall gibt es eine spezielle Form von Interaktion, eine `SPDomain`-zu-`SPDomain` Interaktion. Somit werden noch drei zusätzliche **PP** Interaktionskanäle definiert: **PP4** zwischen den benachbarten `Dom-FM`s, **PP5** zwischen den benachbarten `Dom-SD`s und **PP6** zwischen den benachbarten `Dom-MS`.

5.2.1.5. Spezifikation des Organisationsmodells

In den vorangehenden Abschnitten wurden informell die Rollen und Interaktionskanäle des Organisationsmodells beschrieben. Bevor in Abschnitt 5.2.3 das Informationsmodell realisiert werden kann, müssen die Entitäten und das Domänenkonzept des ioFMA-Organisationsmodell in UML spezifiziert werden. Eine Vertiefung wird während der Spezifizierung des Informationsmodells vorgenommen.

Als erstes wird das Metamodell des ioFMA-Organisationsmodells beschrieben. Es beinhaltet die Metaklassen `IOFMADomain`, `IOFMARole` und `InteractChannel`. Die Klasse `IOFMADomain` besteht aus mehreren Rollen (`IOFMARole`), die über einen oder mehrere Interaktionskanäle (`InteractChannel`) Informationen austauschen. Die Klasse `IOFMARole` repräsentiert einen

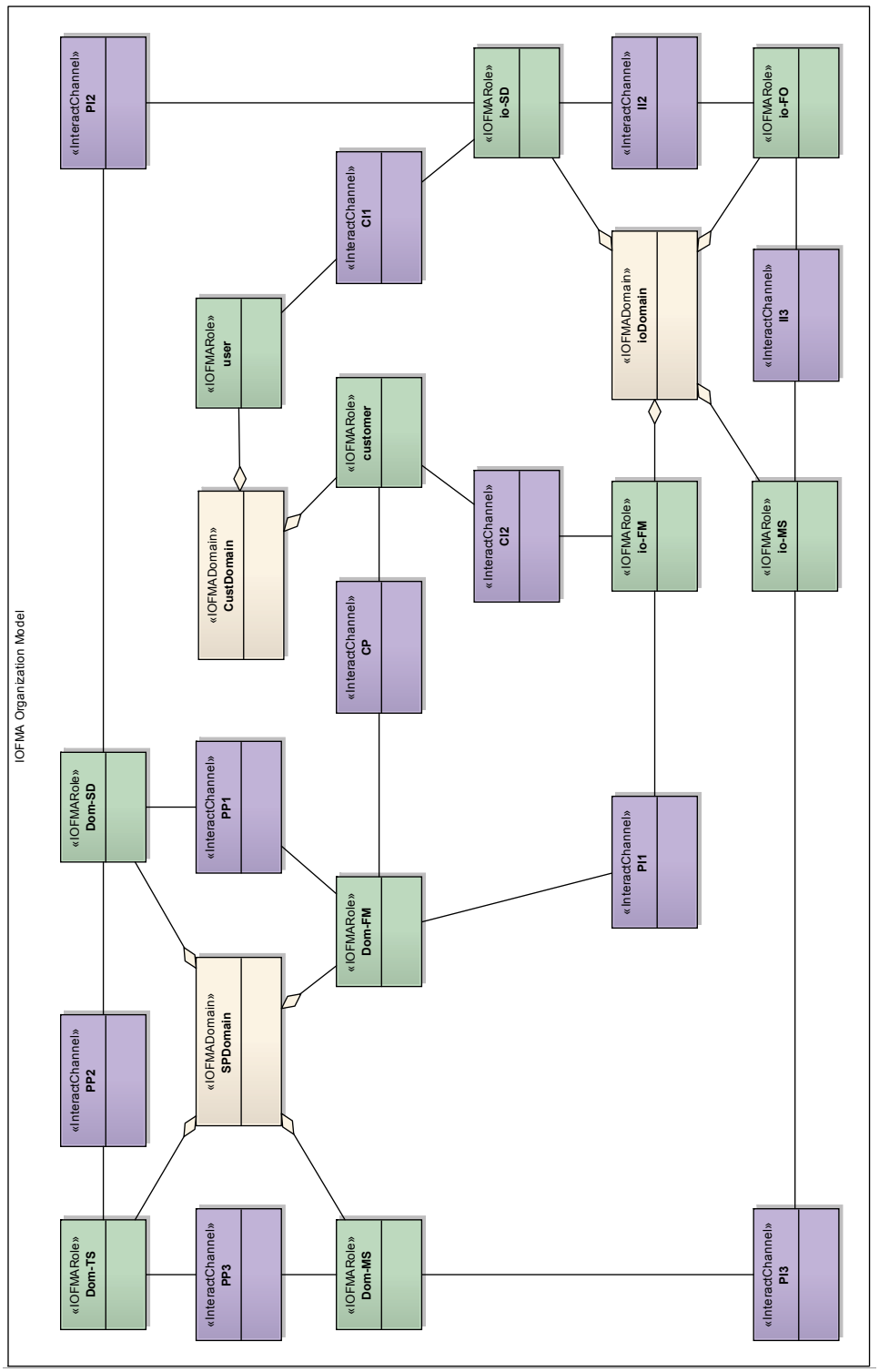


Abbildung 5.5.: Das ioFMA-Organisationsmodell

Funktionsumfang, der von einer Organisation, einer Person, oder einer Entität, die mehrere Rollen gleichzeitig annehmen kann, erbracht wird (mehr in Abschnitt 5.2.3). Die Klasse `InteractChannel` repräsentiert die Verbindung zwischen zwei Rollen, die sich in derselben oder in unterschiedlichen Managementdomänen befinden. Abbildung 5.4 fasst das Metamodell des ioFMA-Organisationsmodells zusammen.

Unter Benutzung des dargestellten Metamodells kann jetzt das ioFMA-Organisationsmodell spezifiziert werden. Der UML-Erweiterungsmechanismus des Stereotyps kann verwendet werden, um einzelnen Modellelementen – hier Domänen, Rollen und Interaktionskanälen – eine spezifische Semantik zu geben (mehr dazu in Abschnitt 5.2.3). Abbildung 5.5 stellt in einem statischen UML-Klassendiagramm das Organisationsmodell der ioFMA dar. Um die graphische Übersichtlichkeit zu steigern, wurden die Kardinalitäten der Beziehungen zwischen den Klassen des Organisationsmodells weggelassen. Dennoch sind diese sehr wichtige Elemente des Modells. Das ioFMA-Organisationsmodell besteht aus drei Klassen, die die oben beschriebenen Managementdomänen formal darstellen: `SPDomain`, `ioDomain` und `customer`. Diese sind als Stereotypen der Klasse `IOFMADomain` modelliert. Die Managementdomänen sind statisch den entsprechenden Rollen zugewiesen. Stereotypen der Klasse `InteractChannel`, konform zum Metamodell, realisieren die logische Verbindung zweier Rollen.

5.2.2. Das Funktionsmodell

Das Funktionsmodell der ioFMA strukturiert das Fehlermanagement als Ganzes in mehrere ioFMA-Funktionsbereiche auf (gemäß [HAN 99]) und legt allgemeine Managementfunktionen für die ioFMA fest. Im Abschnitt 3.2.2 wurden anhand der Anwendungsfälle Funktionalitäten, die die ioFMA zu erfüllen hat, beschrieben. Hier wird die Brücke zum Funktionsmodell gebaut, indem für jeden einzelnen Funktionsbereich die Funktionalitäten und Dienste sowie die Managementobjekte zur Erbringung der Funktionalität definiert werden. Ansätze für dies wurden in [MaHo 11] beschrieben.

5.2.2.1. Festlegung der Funktionsbereiche

Die Einteilung in die fünf Funktionsbereiche Fault, Configuration, Accounting, Performance, Security Management (FCAPS) wie im Fall des OSI-Referenzmodells [OSI 94] kann hier nicht realisiert werden, da diese Arbeit als Fokus ausschließlich das Fehlermanagement hat. Zusätzlich handelt es sich hier um ein Fehlermanagementkonzept, das in interorganisationalen Umbebungen einsetzbar ist. Wie auch beim Organisationsmodell bzgl. der Domänen, sind auch hier die Funktionalitäten aus einem lokalen bzw. einem interorganisationalen Gesichtspunkt zu betrachten. Die Anwendungsfälle (vgl. Abschnitt 3.2.2) und die Funktionalitäten, die diese repräsentieren, betreffen drei unterschiedliche Ebenen: die des Nutzers, die des einzelnen beteiligten Service-Provider und den interorganisationalen Bereich. Die

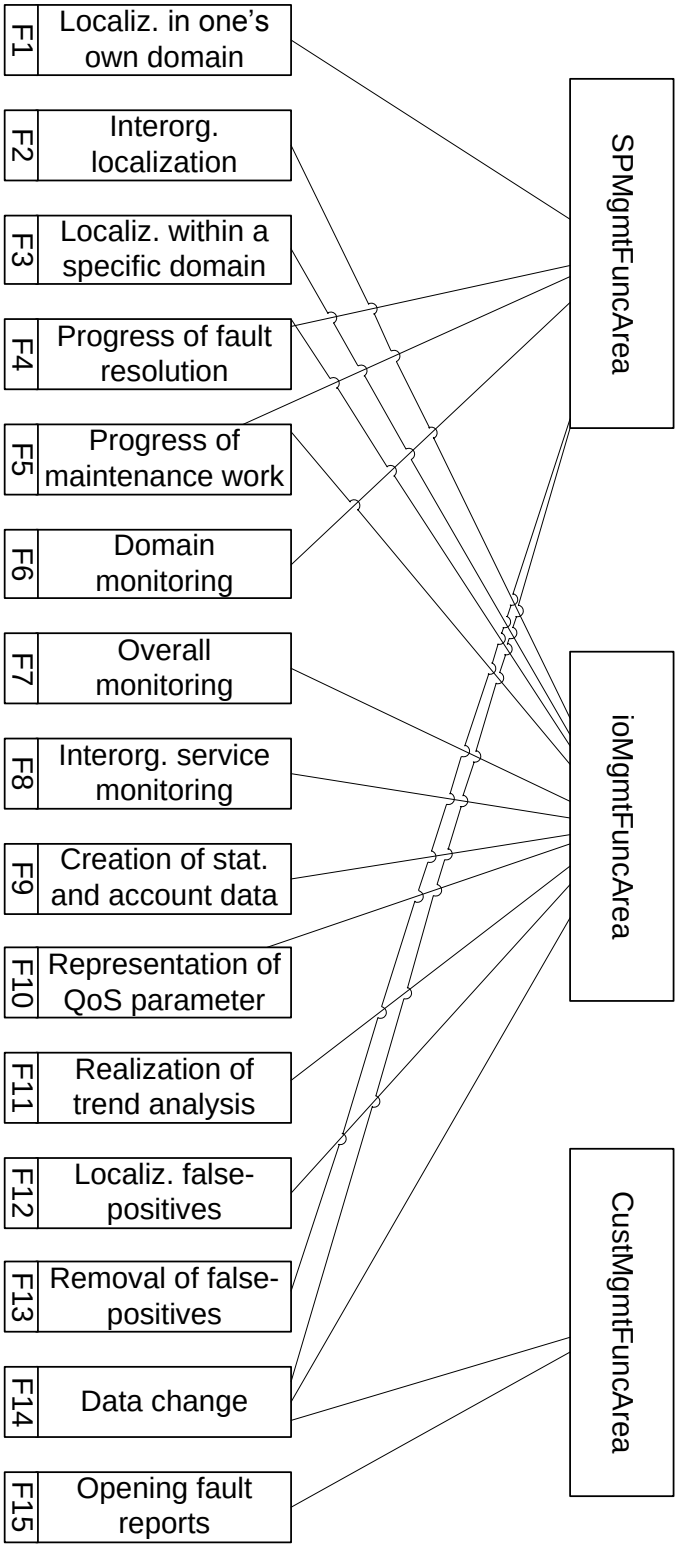


Abbildung 5.6.: Die Funktionsstruktur der ioFMA

Einteilung des Fehlermanagements in Funktionsbereiche orientiert sich daher an dieser drei Ebenen.

Es werden drei Funktionsbereiche definiert: Customer-Management (`CustMgmtFuncArea`), Service-Provider-Management (`SPMgmtFuncArea`) und interorganisationales Management (`ioMgmtFuncArea`). Der Kern des Managements ist der Funktionsbereich interorganisationales Management, aber ohne die beiden anderen Funktionalbereiche ist ein interorganisationales Fehlermanagement nicht denkbar.

5.2.2.2. Festlegung der Managementfunktionen

Die oben definierten Funktionsbereiche nutzen für die Realisierung des io-Fehlermanagements bestimmte allgemeine Managementfunktionen. Zur Festlegung der Funktionalitäten wurden im Abschnitt 3.2.2 dreizehn Anwendungsfälle beschrieben. Diese waren in fünf Kategorien (Fehlerlokalisierung, Status der Fehlerbearbeitung/Wartungsarbeit, Überwachung, Reporting und Falschfehlermeldungen) zusammengefasst. Somit werden, abhängig von den in Abschnitt 3.2.2 beschriebenen Anwendungsfällen, die Managementfunktionen definiert. Die Funktionen F1-F13 können 1:1 auf deren gleichnamige Anwendungsfälle abgebildet werden. Die beiden letzten Funktionen F14 - Datenänderung, F15 - Erstellen (Öffnen) einer Fehlermeldung sind Managementfunktionen, die bei der Beschreibung der Anwendungsfälle nicht betrachtet wurden, jedoch sind diese als Funktionen für die ioFMA unerlässlich. Die Funktionsstruktur der ioFMA wird in Abbildung 5.6 zusammengefasst. In dieser Abbildung wird auch die Zugehörigkeit der unterschiedlichen Managementfunktionen zu den Funktionsbereichen angemerkt.

Fehlerlokalisierung

Funktion Fehlerlokalisierung Die Funktion Fehlerlokalisierung besteht gemäß dem gleichnamigen Anwendungsfall aus drei Funktionen: Fehlerlokalisierung in der eigenen Domäne (F1), in irgendeiner Domäne (F2) und einer bestimmten anderen Domäne (F3).

in der eigenen Domäne

Abbildung 5.7 verschafft einen Überblick über die auszuführende Funktion Fehlerlokalisierung in der eigenen Domäne (F1). Der user meldet eine Störung (*report(failure)*) an den Dom-SD. Dieser versucht Informationen über diese Störung zu bekommen. Diese kann er aus einer CMDB oder aus einem TTS beziehen. Wenn keine Informationen bereitstehen, wird die Störung an den technischen Spezialisten (Dom-TS) weitergeleitet (*fwd(failure)*). Die Bearbeitung wird mit einem der Ergebnisse ,,gelöst" oder ,,nicht-gelöst" abgeschlossen, das zum Service Desk zurückgesendet wird. Der Service Desk teilt anschließend dem User diese Antwort mit. Im Falle, dass die Störungsursache nicht gefunden werden konnte, wird die Störungsmeldung zum io-SD weitergeleitet, der die Weiterbearbeitung übernimmt. Der User wird darüber ebenfalls informiert.

interorganisational

Bei der interorganisationalen Fehlerlokalisierung (F2) wird die Aktion vom (Dom-SD) angestoßen, indem es dem io-SD eine Störung aus der eigenen Domäne meldet (*report(failure)*).

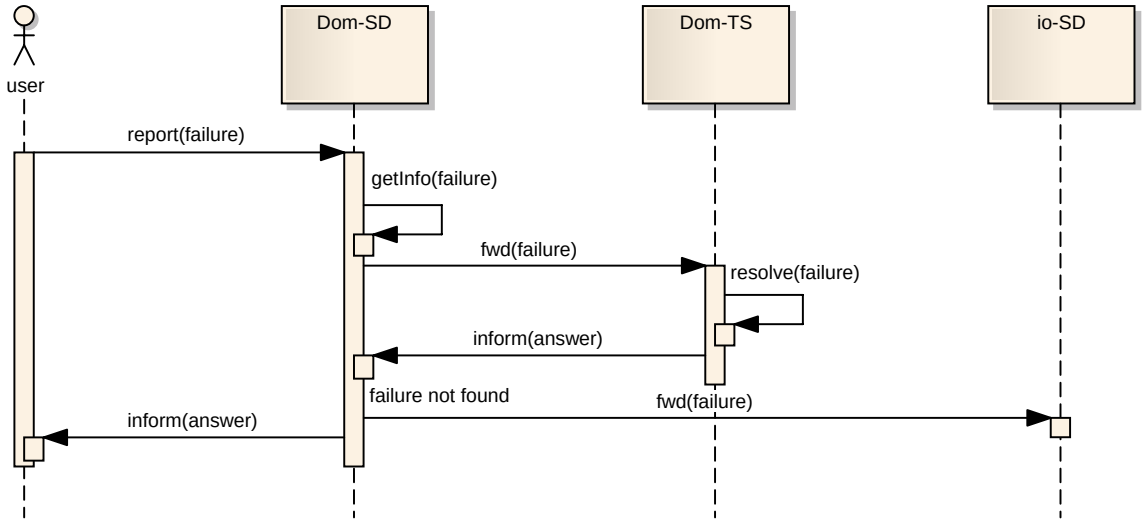


Abbildung 5.7.: Sequenzdiagramm für die Managementfunktion Ferhlerlokalisierung in der eigenen Domäne

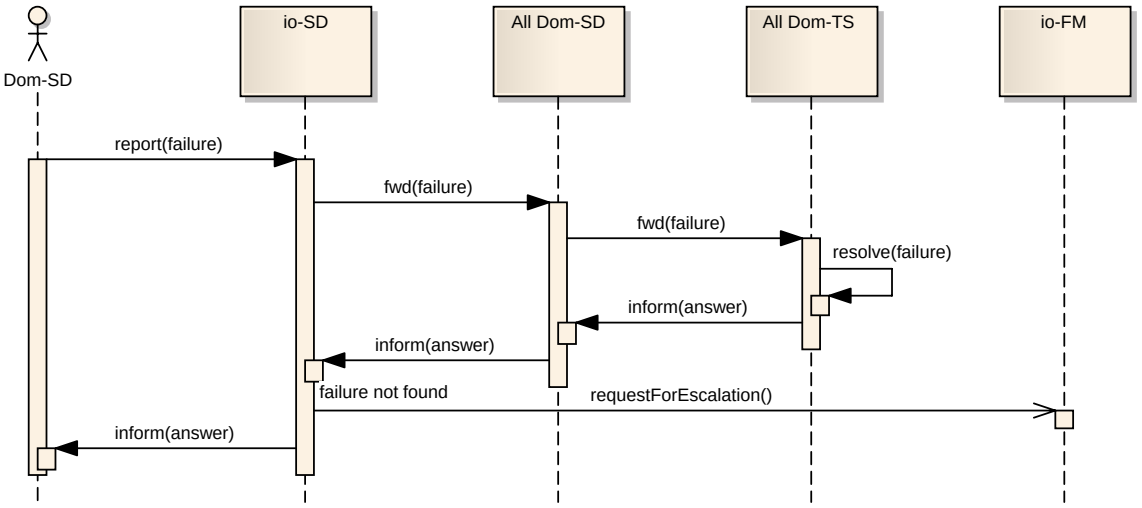


Abbildung 5.8.: Sequenzdiagramm für die Managementfunktion interorganisationale Fehlerlokalisierung

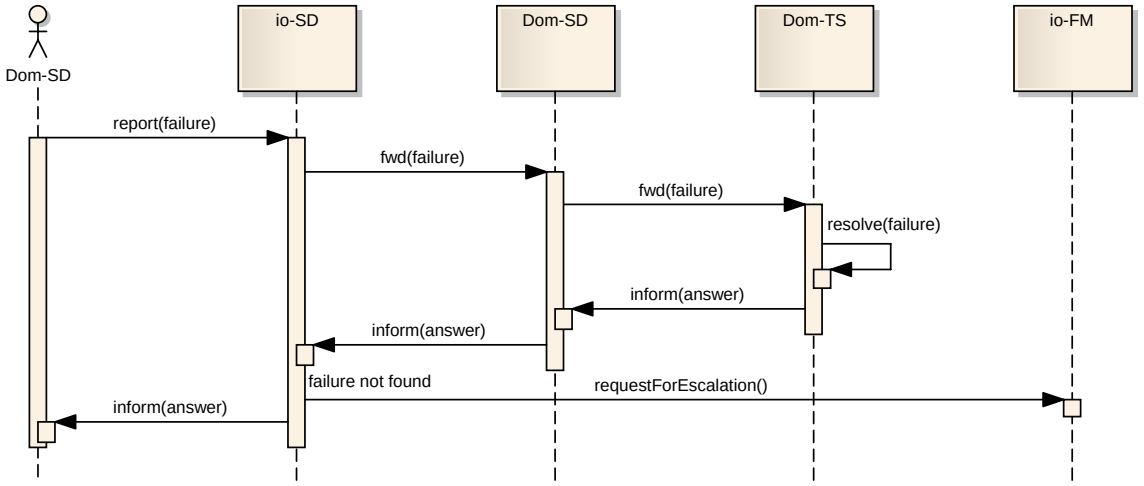


Abbildung 5.9.: Sequenzdiagramm für die Managementfunktion Fehlerlokalisierung in einer anderen Domäne

Diese wird vom io-SD (falls diesem noch keine Störungsursache für den gemeldeten Fehler bekannt ist) an alle an der Erbringung des Dienstes beteiligten Dom-SDs weitergeleitet (*fwd(failure)*). Die Domänen suchen nach einer möglichen Störungsursache in der eigenen Domäne (möglicherweise durch Weiterleitung an den entsprechenden technischen Spezialisten). Wenn das realisiert ist, melden sich die Domänen am io-SD zurück mit dem entsprechenden Ergebnis. Dabei wird im Idealfall die Fehlerursache in einer Domäne lokalisiert. Wird der Fehler nicht entdeckt, muss eine spezielle Eskalationsprozedur vom io-FM gestartet werden (*requestForEscalation()*). Die Domäne, die die Störung gemeldet hat, wird entsprechend informiert. Abbildung 5.8 stellt diese Managementfunktion in einem Sequenzdiagramm dar.

in einer anderen Domäne

Wenn die gemeldete Störung gleich bestimmten Domäne zugewiesen werden kann, kann diese die Störung intern in dieser Domäne bearbeitet werden, beispielsweise durch den technischen Spezialisten dieser Domäne (Dom-TS *resolve(failure)*). Nach der Lösung der Störung wird diese zurück zum Dom-SD bzw. io-SD und den meldenden Dom-SD mitgeteilt. Falls die Störung nicht gelöst wurde, werden die genannten Rollen darüber informiert und der io-SD leitet diese Information zum io-FM weiter, mit der Bitte zum Einleiten der Eskalationsprozedur (siehe Abbildung 5.9).

In den obigen Sequenzdiagrammen wurden einige Nachrichten für die Interaktionen zwischen den unterschiedlichen beteiligten „Objekten" benutzt. Diese werden bei der Beschreibung der Objekte im Informationsmodell mit einbezogen.

Funktion Statusaktualisierung Bei der Funktion Statusaktualisierung werden zwei Aspekte betrachtet: Der Status der Fehlerbearbeitung (F4) und der Status von Wartungsarbeiten

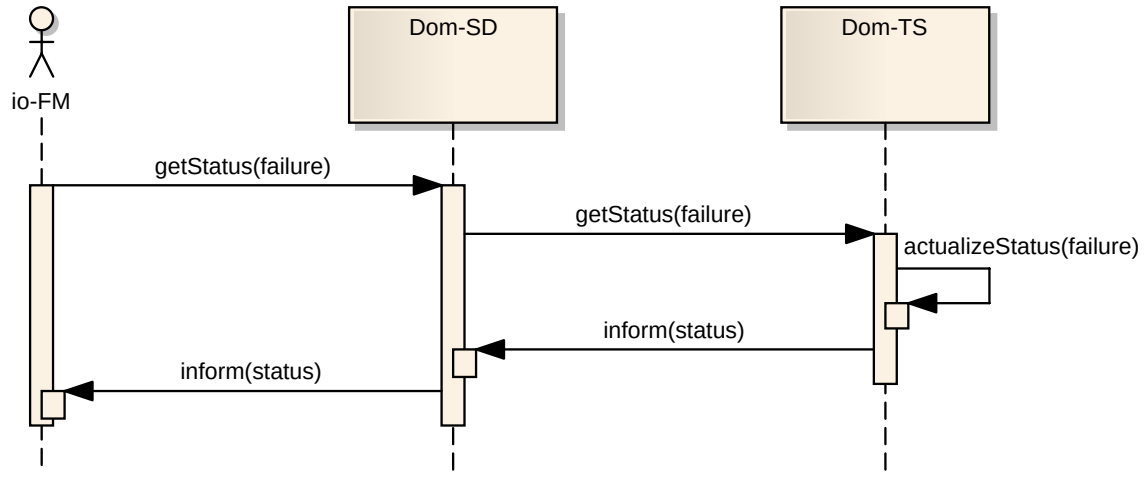


Abbildung 5.10.: Sequenzdiagram für die Managementfunktionen Statusaktualisierung

(F5). Obwohl diese zwei als separate Funktionen zu sehen sind, ist der Ablauf fast identisch.

Abbildung 5.10 stellt das Sequenzdiagramm für die Statusaktualisierung dar. Obwohl die Statusaktualisierung von mehreren Rollen (user, Dom-FM, Dom-SD, io-FM, io-SD) benötigt wird, wird es hier beispielhaft für den io-FM dargestellt. Alle anderen Varianten sind als Untermengen dieser zu betrachten. Der io-FM fragt den Status der Fehlerbearbeitung (*getStatus(failure)*) am Service Desk (Dom-SD) der betroffenen Domäne nach. Der Dom-SD informiert sich darüber beim Dom-TS, der sich mit der Berabeitung befasst. Dieser aktualisiert den Status der Fehlerbearbeitung und übermittelt ihn zurück bis zum io-FM. Ideal wäre, wenn der Status vom Dom-TS in regelmäßigen Abständen aktualisiert wird, so dass er jedem, der diesen ansehen möchte, auch zur Verfügung steht. Fast identisch läuft die Statusaktualisierung von Wartungsarbeiten; der einzige Unterschied besteht im Parameter maintenWork, der anstatt des Parameters failure übergeben wird.

Funktion Fehlermonitoring Die Funktion Fehlermonitoring läßt sich auch gemäß den Anwendungsfällen in Abschnitt 3.2.2.3 in drei Kategorien teilen: Domänenüberwachung (F6), Gesamtstatusüberwachung (F7) und Überwachung eines interorganisationalen Dienstes (F8).

Die Domänenüberwachung (F6) wird realisiert, wenn Domänen-intern Monitoringinformationen benötigt oder wenn von einer an der Diensterbringung beteiligten Domäne Monitoringinformationen über eine bestimmte Domäne benötigt werden. Abbildung 5.11 stellt den Ablauf der Domänenüberwachung, angestoßen vom io-FM, dar. Die Domänenüberwachung kann ebenfalls von user oder von einer bestimmten Domäne angestoßen werden. Dann fordert der io-FM Monitoringdaten über eine bestimmte Domäne an. Jedoch sind

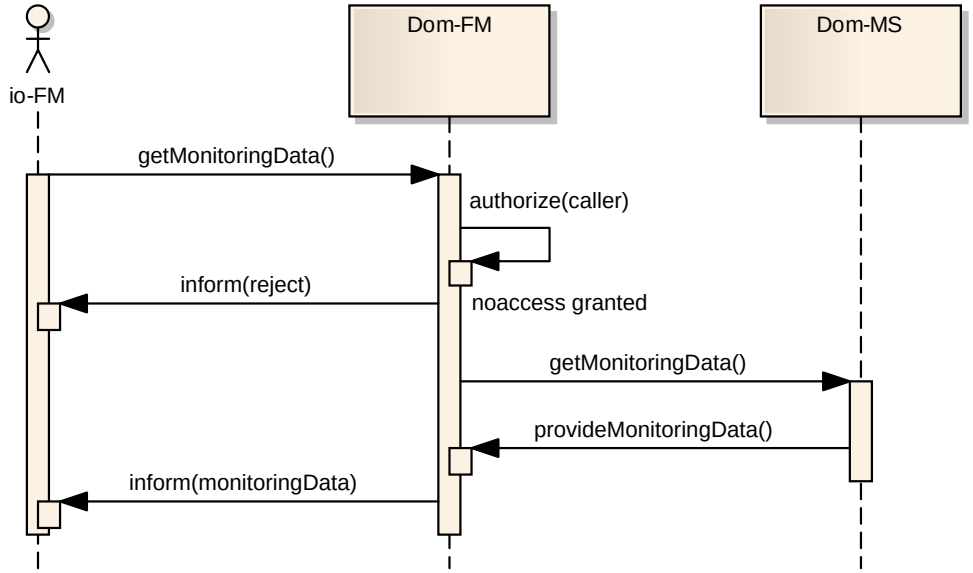


Abbildung 5.11.: Sequenzdiagramm für die Managementfunktion Domänenüberwachung

diese Daten nicht frei zugreifbar, so dass die Domäne (hier stellvertretend der Dom-FM) erst prüfen muss, ob die anfordernde Instanz die benötigten Zugriffsrechte für die besagte Domäne besitzt. Ist dies der Fall, werden die Monitoringdaten vom lokalen Monitoringsystem angefordert und diese anschließend dem io-FM oder der anfordernden Rolle bereitgestellt.

Die zweite Funktion ist die Gesamtstatusüberwachung (F7). Dies erfordert entweder die Abholung von Monitoringdaten von den beteiligten Domänen und die Zusammenfassung anhand eines vereinbarten Algorithmus (durch Korrelation, Aggregation usw.) oder den Einsatz eines Monitoringwerkzeugs auf interorganisationaler Ebene. Der Vorteil eines interorganisationalen Monitoringwerkzeugs ist, dass die Daten automatisch in eine gemeinsame Semantik transformiert werden. Dies ist aber u. U. schwer zu realisieren, aufgrund der Autonomie der Provider. Die Gesamtstatusüberwachung muss nicht unbedingt vom io-FM angefordert werden, sondern kann auch von einer der Domänen angestoßen werden. Abbildung 5.12 zeigt einen möglichen Ablauf, in dem der io-FM die Gesamtüberwachung anfordert. Alle anderen Fälle kann man als Untermenge von diesem betrachten.

Der io-FM (oder eine andere Instanz) fordert von allen beteiligten Domänen Monitoringdaten an (*getMonitoringData()*). Die jeweilige Domäne überprüft die Zugriffsrechte (*authorize(caller)*) für diese Instanz und nur, wenn diese autorisiert ist diese Daten zu bekommen, wird die Aktion weitergeführt. Die Dom-MS der jeweiligen Domäne stellen die Monitoringdaten zur Verfügung (*provideMonitoringData()*), die daraufhin zentral nach bestimmten Regeln (z. B. Aggregation) zusammengefasst werden (*centralize(monitoringData)*).

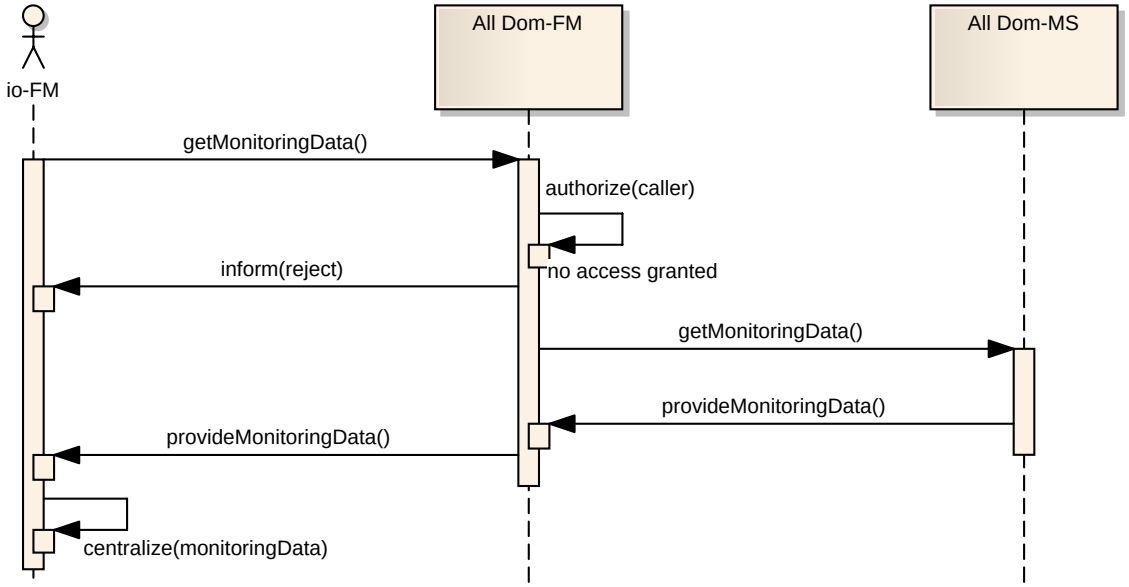


Abbildung 5.12.: Sequenzdiagramm für die Managementfunktion Gesamtstatusüberwachung

Bei der Überwachung eines interorganisationalen Dienstes (F8) geht es um einen Spezialfall der Gesamtstatusüberwachung. Es werden gegenüber der in Abbildung 5.12 dargestellten Gesamtüberwachung nicht die Monitoringdaten aller Domänen des Providernetzwerks, sondern nur von den an einer bestimmten Dienstinstanz beteiligten Domänen angefordert. Ansonsten bleibt der Ablauf unverändert.

Funktion Reporting Die Reporting Funktion besteht aus den drei Teilfunktionen: Statistiken und Accountingdaten (F9), QoS-Parameter (F10) und Trendanalysen (F11) und ist diejenige Funktion, die die Quantifizierung von dienstbezogenen Parametern realisiert, so dass diese verglichen und visualisiert werden können.

Die ersten zwei Funktionen können vom Ablauf her ähnlich wie die Gesamtmonitoring-Funktion dargestellt werden. Der wichtigste Unterschied besteht im Nachrichtenaufruf. Die übermittelten Nachrichten heißen *getStatData()*, *getAccountData()* und *getPartQoS()*. Im Fall der Trendanalyse, werden auf Basis der in den vorherigen zwei Fällen gesammelten Daten Einschätzungen über potentielle Fehler gemacht,wie beispielsweise, wann und unter welchen Umstände Fehler auftreten.

Funktion False Positives Die Lokalisierung (F12) und Markierung von False Positives (F13) wird im Sequenzdiagramm in Abbildung 5.13 zusammengefasst. Diese Funktion kann von

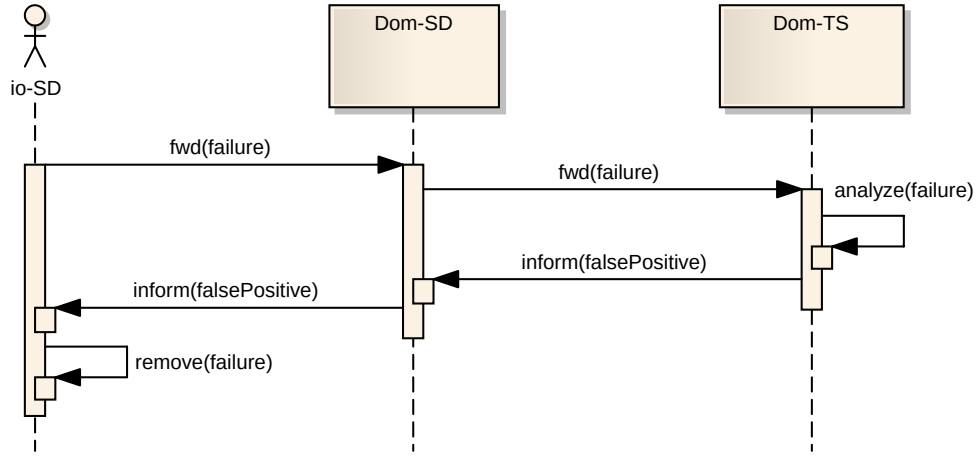


Abbildung 5.13.: Sequenzdiagramm für die Managementfunktionen bzgl. Falschfehlermeldungen

mehreren Rollen benutzt werden, hier wird aber nur die Variante der io-SD behandelt. Die anderen sind als Untermenge hiervon zu betrachten. Der Anfang dieser Aktion ist identisch mit der Funktion Lokalisierung von Fehlern in einer bestimmten Domäne. Somit leitet der io-SD eine Störungsmeldung zu einer bestimmten Domäne (*fwd(failure)*) weiter, diese wird vom Dom-SD angenommen und zu den technischen Spezialisten zur Überprüfung (*analyze(failure)*) geschickt. Diese stellen fest, dass kein Fehler vorliegt und dass es sich nur um einen Fehlalarm handelt. Diese Information wird an den Dom-SDs bzw. io-SD weitergegeben (*inform(falsePositive)*). Der io-SD als io-Koordinator stellt den Status dieser Fehlermeldung auf *noFault* zurück (*remove(failure)*).

Funktion Datenänderung Die Funktion Datenänderung (F14) wird für jegliche Änderungen im Datenbestand benutzt, z. B. bei Konfigurationsänderungen oder spezifisch fehlerbezogenen Änderungen (z. B. Fehlerreports, fehlerhafte Komponenten usw.). Die Änderungen der Konfigurationsdaten wird von dem Änderungs- bzw. Konfigurationsmanager (Change- resp. Configuration-Manager) über die ,,Schnittstelle" io-FM realisiert. Für fehlerbezogene Änderungen ist ausschließlich das Fehlermanagement zuständig. Das Sequenzdiagramm in Abbildung 5.14 stellt diese Datenänderungsfunktion dar.

Es wird hier nur die Funktion, die von user angestoßen wird, beschrieben. Diese kann aber auch von anderen Rollen gestartet werden. Der user muss eine Änderung durchführen (beinhaltet auch das Löschen von Objekten) und leitet daher eine Anfrage (*requestChange(data)*) an den customer weiter. Dieser überprüft in seiner Managementrolle auf Kundenseite die Machbarkeit der Änderung aus Kundensicht (*verify(data)*) und leitet sie dann, wenn die Änderung gerechtfertigt ist, an den io-FM weiter. Nachdem der io-FM diese Änderung

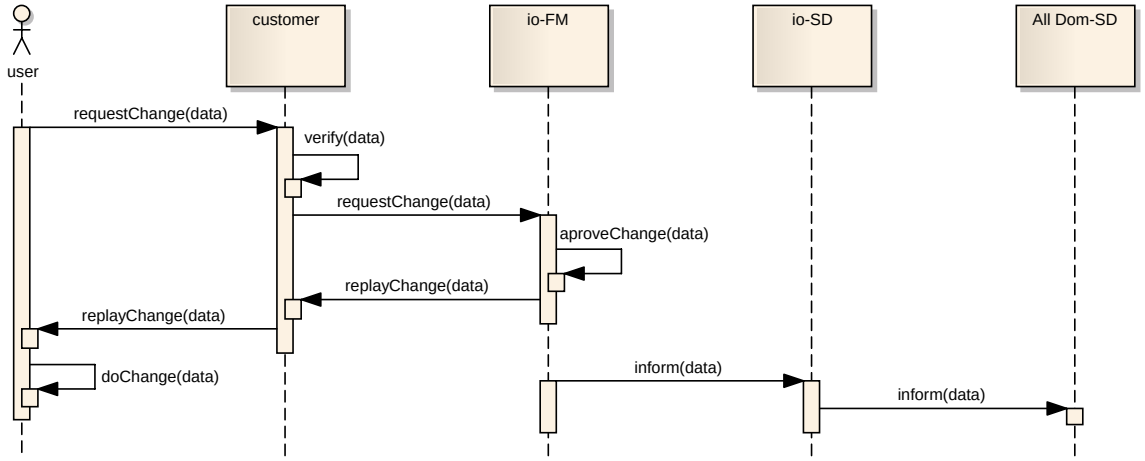


Abbildung 5.14.: Sequenzdiagram für die Managementfunktionen Datenänderung

genehmigt hat, wird die Zustimmung auf demselben Weg zurück zum user gesendet (*replayChange(data)*), und dieser führt die Änderung durch (*doChange(data)*).

Das Erstellen (Öffnen) einer Fehlermeldung (F15) ist eine sehr triviale Funktion, die hier der Vollständigkeit halber angegeben wird. Dies wird in der Customer Domäne vom Nutzer ausgeführt, indem er ein Trouble Ticket im Trouble Ticket System (TTS) der Domäne öffnet.

5.2.2.3. Besonderheiten des Funktionsmodells bzgl. Diensterbringungsformen

Im Fall der hierarchischen Form der Diensterbringung wird eine vereinfachte Variante des Funktionsmodells verwendet. Da einige Interaktionskanäle in diesem Fall nicht existieren (z.B. ist der io-FM in diesem Fall Dom-FM innerhalb der Domäne des Wurzel-Providers), werden auch die Funktionen viel einfacher. Bei der heterarchischen Form der Diensterbringung ist die Komplexität des Funktionsmodells ähnlich hoch wie im Allgemeinfall. Die einzige Variable, die beim Ablauf der Managementfunktionen noch dazukommt, ist die zusätzliche Rolle End-Site-SD, die dann mit dem io-FM bzw. io-SD kommuniziert, sowie mit den anderen Domänen.

Wie sich die zwei Formen der interorganisationalen Diensterbringung auf das Funktionsmodell auswirken, wird beispielhaft anhand der Funktion Fehlerlokalisierung in einer anderen Domäne (F3) (vgl. Abbildung 5.9) beschrieben.

Im hierarchischen Fall, wie Abbildung 5.15 bildlich darstellt, wird eine stark vereinfachte Variante der Funktion angewendet. Der io-SD, der für den Nutzer eigentlich auch direkt der Dom-SD ist, leitet den Fehler aus seiner Domäne (Wurzel-Domäne) zu der Domäne, in der

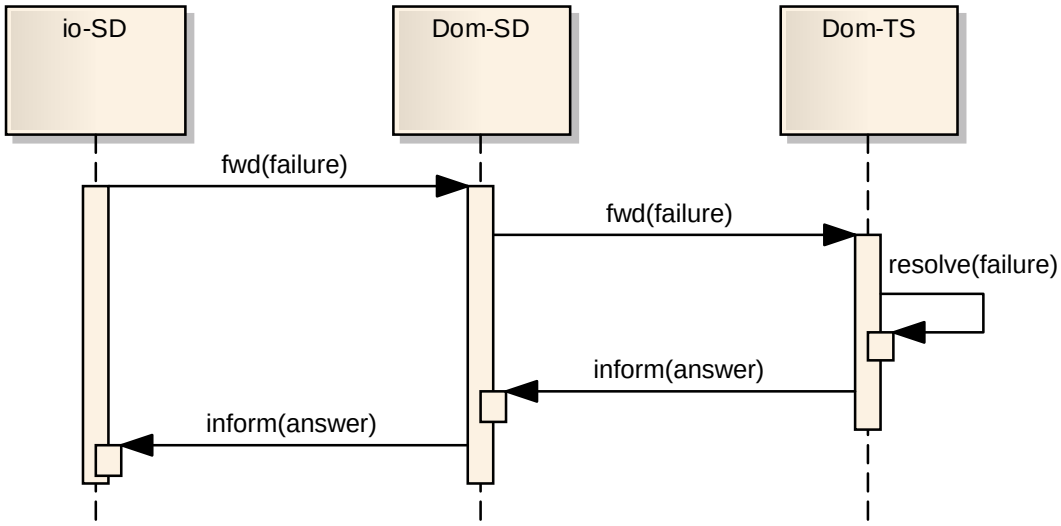


Abbildung 5.15.: Sequenzdiagramm für die Managementfunktion Fehlerlokalisierung in einer anderen Domäne (hierarchische Form)

der Fehler vermutet wird, weiter. Der Fehler wird hier von dem zuständigen Dom-SD angenommen und zum Dom-TS zur Behebung weitergeleitet. Dieser behebt den Fehler oder nicht und informiert entsprechend den Dom-SD, der die Antwort zum io-SD weitergibt. Im negativen Fall leitet der io-SD selbst die Eskalationsprozedur ein.

Im Gegensatz dazu wird im heterarchischen Fall eine ausführlichere Prozedur durchgeführt, da noch die erwähnten interorganisationalen Rollen hinzukommen und deshalb die Interaktionskanäle komplizierter aufgebaut sind (vgl. Abbildung 5.3). Ein Fehler wird grundsätzlich vom End-Site-SD erkannt (entweder über Monitoringwerkzeuge oder durch Meldungen seitens des Kunden). Als erstes wird unter den zwei End-Site-SDs festgelegt, ob es sich tatsächlich um einen Fehler handelt (also wird die Bestätigung der beiden Randdomänen benötigt) oder nicht (vgl. Abbildung 5.16). Falls es sich tatsächlich um einen bestätigten Fehler handelt, leiten die zwei Randdomänen (End-Site-SD) einen Broadcast zu den anderen an der Diensterbringung beteiligten Domänen weiter, um die fehlerhafte Domäne zu finden. Nach diesem Broadcast – wenn die Domäne, auf die der Fehler zurückzuführen ist, bekannt ist – wird sowohl der io-FM als auch der io-SD informiert. Der erstere ist ab diesem Zeitpunkt für die Lösung des Fehlers innerhalb der fehlerhaften Domäne zuständig, dennoch bleibt der io-SD der Ansprechpartner für den Kunden. Es folgen dann innerhalb dieser Domänen die schon aus dem Allgemeinfall bekannten Schritte.

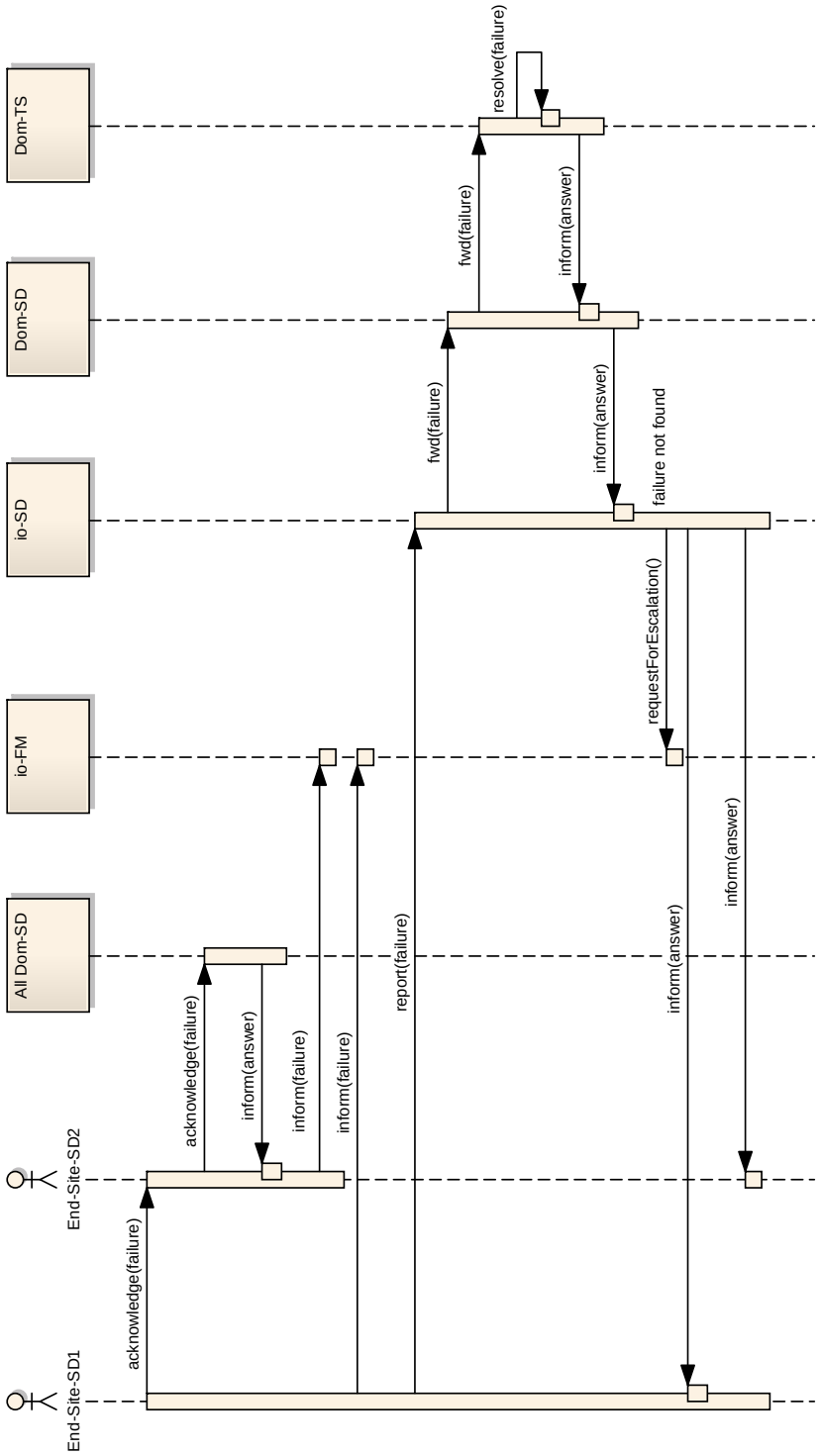


Abbildung 5.16.: Sequenzdiagramm für die Managementfunktion Fehlerlokalisierung in einer anderen Domäne (heterarchische Form)

5.2.3. Das Informationsmodell

In einer Managementarchitektur bestimmt das Informationsmodell einen eindeutigen Beschreibungsrahmen für die Gesamtheit der Managementobjekte. Nachdem in den Abschnitten 5.2.1 und 5.2.2 die Rollen und Domänen bzw. Funktionsbereiche und Funktionen beschrieben wurden, werden in diesem Abschnitt die Managementobjekte zur Erbringung der Funktionalität definiert. Um das zu realisieren, muss einerseits eine geeignete Spezifikationssprache für die Informationsmodellierung festgelegt werden (hier UML). Auf der anderer Seite ist die Strukturierung des untersuchten Sachgebietes (z. B. in Modelldomänen) sowie die Definition von abstrakten und generischen Wurzel-Klassen als Basis für die Spezialisierungs- und Vererbungshierarchie zur Ableitung der ioFMA-Klassen und deren Beziehungen notwendig. Das Informationsmodell der ioFMA wurde in [MaSc 11] teilweise beschrieben, hier folgt aber einen vollständige Abhandlung dieses Modells.

5.2.3.1. Das Domänenmodell der ioFMA

Das Informationsmodell besteht aus separaten Domänen (Begriff aus der objektorientierten Entwicklung), die als thematisch orientierte Gruppierungen von Entitäten zur Strukturierung von Modellen fungieren [Larm 01]. Der hier benutzte Begriff ist nicht derselbe wie der Domänen-Begriff als Teil des Organisationsmodells nach Hegering et al. [HAN 99]. In einer ioFMA-Domäne sind die Entitäten, die einem spezifischen, konzeptionell abgeschlossenen Bereich des ioFMs zugeordnet werden können, beinhaltet.

Basierend auf dem Organisationsmodell wird zunächst die Domäne Role eingeführt, um formal die unterschiedlichen (organisatorischen) Domänen, Rollen und Interaktionskanäle (vgl. Abschnitt 5.2.1), die im interorganisationalen Fehlermanagement benötigt werden, übergreifend zusammenzufassen.

Ressourcen, Dienste und Fehler im interorganisationalen Umfeld sind die Managementobjekte dieses ioFMA-MIBs. Entsprechend werden die drei Modelldomänen Resource, Service und Fault modelliert. Die Domäne Resource muss eine Spezialisierung der Ressourcen beinhalten (ob die Ressource intra- oder interorganisational benötigt wird). Sie muss außerdem die möglichen aufgetretenen Ressourcenalarme, aber auch die Informationen über der Zustand der jeweiligen Ressource abbilden können. Analog wird die Domäne Service definiert, indem man die Identifikation des Dienstes, die Art des Dienstes (intra- oder interorganisational), mögliche Dienstprobleme als auch Dienstinformationen abbildet. Die Fault Domäne fasst zusätzlich zu einer Spezialisierung in intra- und interorganisationale Fehler auch alle für den Dienst oder die Ressourcen relevanten Störungsinformationen zusammen.

Die Spezifikation aller im ioFMA-Informationsmodell definierten Entitäten (Ressourcen, Dienste, Fehler, Rollen u. a.) wird in einer separaten Domäne definiert (siehe IOFMA-Specification in Abschnitt 5.2.3.8). So werden auch die für das Fehlermanagement relevanten

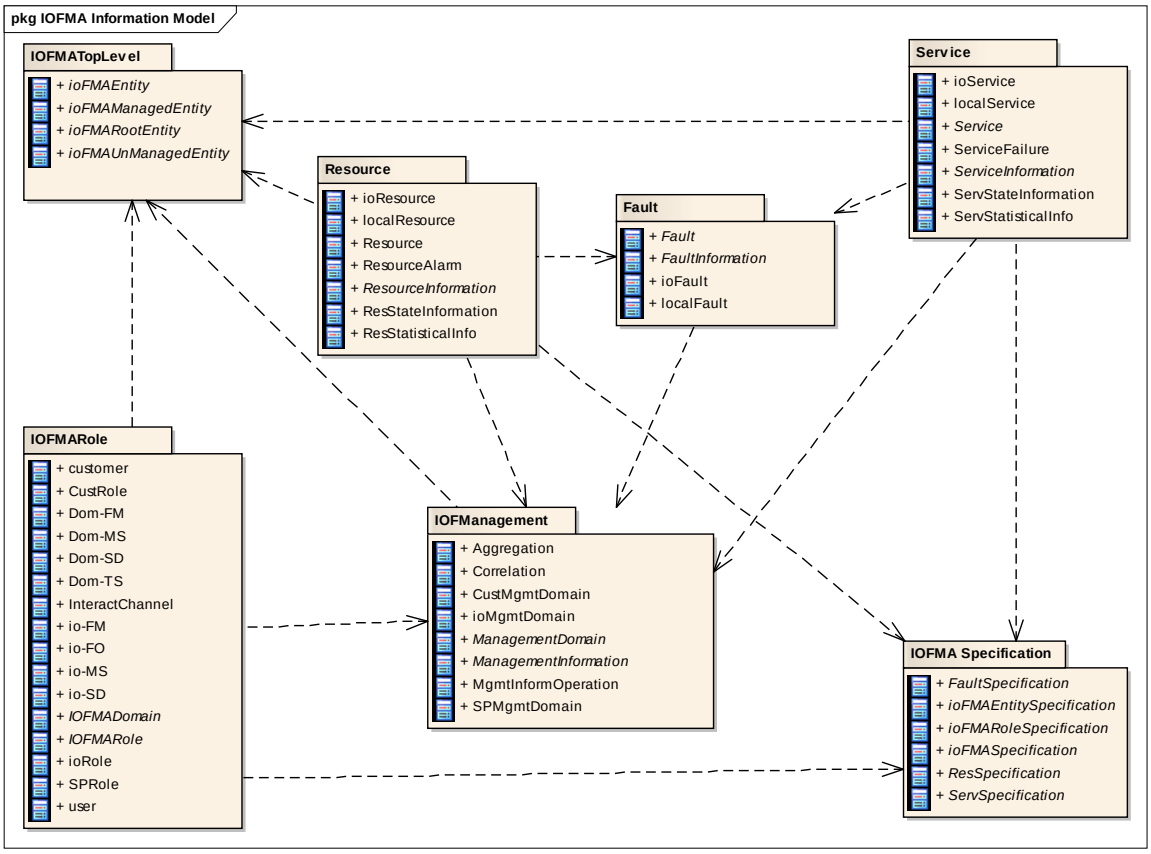


Abbildung 5.17.: Domänen des ioFMA-Informationsmodells

Informationen, Operationen und Funktionsbereiche in der Domäne IOFManagement zusammengefasst.

Um einen Satz allgemeiner ioFM-Entitäten zu definieren, ergibt sich die Notwendigkeit einer übergreifenden Domäne, die aber gleichzeitig als Grundlage für die ioFMA fungiert. Hiermit wird die Domäne IOFMATopLevel eingeführt, die die Wurzel-Entitäten und abstrakten Oberklassen der ioFMA beinhaltet.

Abbildung 5.17 fasst diese Domänen im ioFMA-Informationsmodell zusammen: Die Domänen `IOFMATopLevel` repräsentieren die allgemeinsten Klassen des Modells, beinhalten also die Wurzel-Entitäten und die weitgehend abstrakten Oberklassen. Die Domäne `IOFManagement` modelliert das interorganisationale Fehlermanagement mit Bezug auf die Managementinformation. Damit hängen die drei anderen Domänen `Service`, `Resource` und `Fault` zusammen, die geeignete Datenmodelle über die Managementobjekte der ioFMA bereitstellen. Um Rollen und Domänen (administrativ) tiefer als im Organisationsmodell definieren zu können, wird die Domäne `IOFMARole` benutzt. Übergreifend wird die Domäne `IOFMA-`

`Specification` eingeführt, die die Spezifikationen unterschiedlicher Dienste, Komponenten, Rollen und Managementinformationen bereitstellt.

5.2.3.2. Die Modelldomäne `IOFMATopLevel`

In der Domäne `IOFMATopLevel` liegt der Fokus auf der Bereitstellung eines allgemeinen Satzes an ioFMA-Entitäten als Basis des ioFMA-Informationsmodells. `IOFMATopLevel` beinhaltet abstrakte und generische Wurzelklassen, von denen sich fast alle Klassen des ioFMA-Informationsmodells ableiten lassen. Eine Zusammenfassung der `IOFMATopLevel` Domäne ist in Abbildung 5.18 dargestellt.

Basierend auf einem klassischen obiektorientierten Modellierungsansatz, wird die Klasse `ioFMARootEntity` als Wurzelklasse der Klassenhierarchie des ioFMA-Informationsmodells eingeführt. Es werden dabei ein Satz von Attributen und Methoden, die allen ioFMA-Entitäten gemeinsam sind, definiert. Das Attribut `objectID` wird für die Identifizierung von Objektinstanzen eingesetzt. Die Benennung und Beschreibung der Entität wird anhand der Attribute `entityName` und `description` realisiert. Über die Methoden `get` (*getDescription()*, *getEntityName()*, *getObjectID()*) und `set` (*setDescription()*, *setEntityName()*, *setObjectID()*) können diese Informationen gelesen und verändert werden.

Die Klasse `ioFMARootEntity` wird durch vier Klassen spezialisiert: `ioFMAEntity`, `IOFMARole`, `IOFMADomain` und `ioFMASpecification`. Mit Ausnahmen der Klasse `ioFMAEntity` werden die anderen Klassen bei den entsprechenden Domänen detaillierter erklärt.

`ioFMAEntity` ist eine abstrakte Klasse, die die Gesamtheit der Managementobjekte darstellt, seien diese ,,gemanaged" oder ,,nicht-gemanaged". Daher werden zwei Klassen eingeführt, die die Klasse `ioFMAEntity` spezialisieren.

- `ioFMAManagedEntity` repräsentiert die Entitäten, auf die das io-Fehlermanagement direkt wirkt. Die gemanagten Entitäten sind interorganisationale Entitäten wie `ioService`, `ioResource` und `ioFault`.

- Die Klasse `ioFMAUnManagedEntity` enthält die Entitäten, auf die das io-Fehlermanagement keinen direkten Einfluss hat, die aber Managementinformationen liefern, die zur Realisierung des io-Fehlermanagements beitragen. Die nicht gemanagten Entitäten im Rahmen der ioFMA sind die sogenannten lokalen Objekte, also intraorganisationale Entitäten, die nicht explizit Teil der zu managenden Objekte sind, aber trotzdem eine wesentliche Rolle bei der Realisierung des io-Fehlermanagements haben: `localService`, `localResource` und `localFault`.

Diese werden bei der entsprechenden Domäne (Service, Resource, Fault) weiter beschrieben.

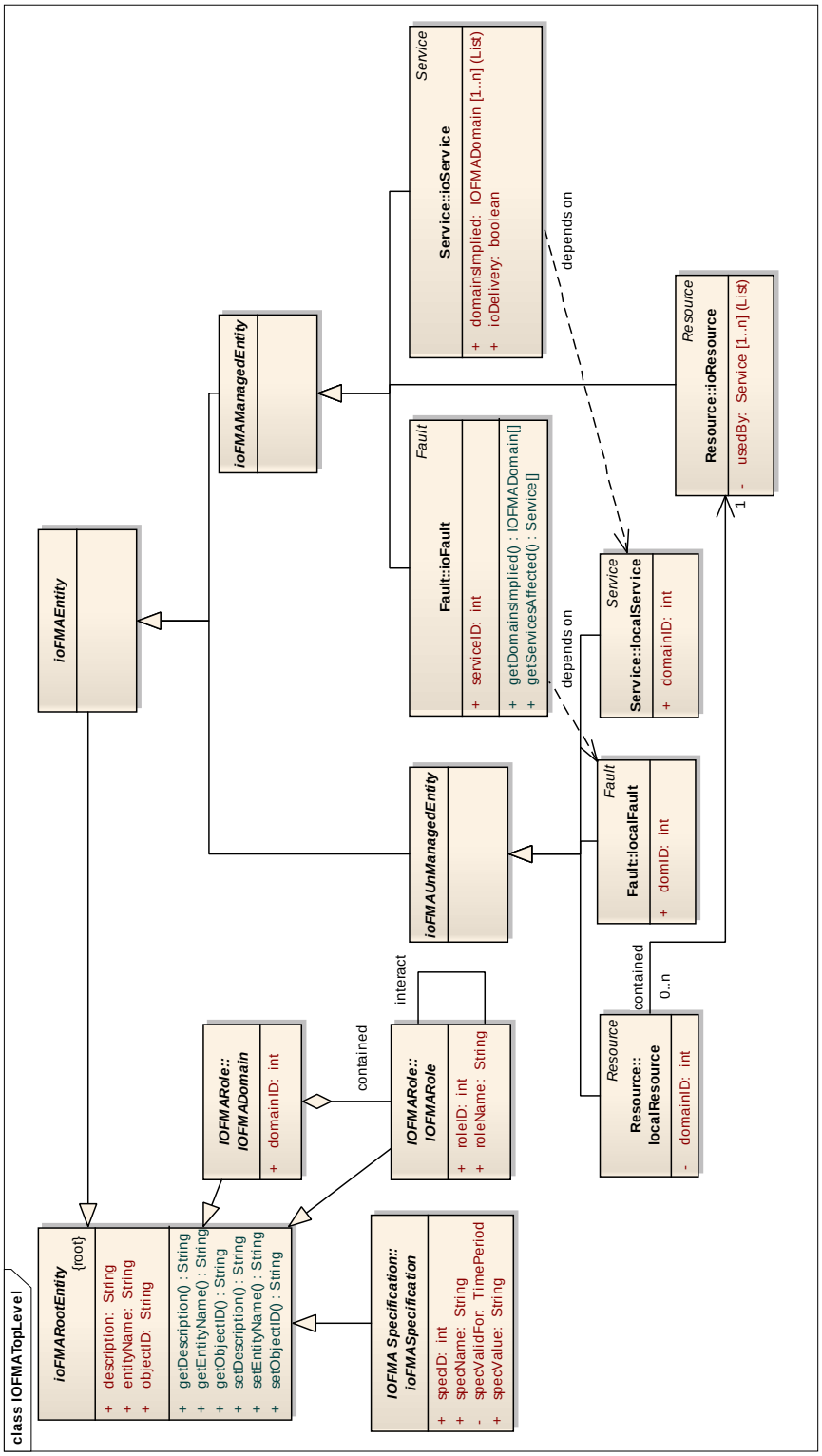


Abbildung 5.18.: Die Domäne `IOFMATopLevel`

5.2.3.3. Die Modelldomäne `IOFMARole`

Die Domäne `IOFMARole` stellt die Modellierung der organisatorischen Domänen und deren zugehörigen Rollen dar. Wie das Diagramm in Abbildung 5.19 zeigt, sind die zwei abstrakten Klassen `IOFMARole` und `IOFMADomain` Spezialisierungen der `ioFMARootEntity`. Die Klasse `IOFMARole` hat als Attribute eine `roleID` und einen `roleName`. Die Klasse `IOFMADomain` hat ebenfalls ein Attribut `domainID`. Die `IOFMARole` wird anhand von einer `ioFMARoleSpecification`-Klasse der Domäne `IOFMASpecification` spezifiziert.

Customer Rollen

Die Klassen `CustRole`, `SPRole` und `ioRole` spezifizieren die Klasse `IOFMARole` für die Kunden-bezogenen, Service Provider-bezogenen und interorganisationalen Rollen. Deren Subklassen sind aus dem Organisationsmodell, wo sie nur namentlich erwähnt wurden, bekannt.

Die Klasse `CustRole` wird durch die Klassen `user` und `customer` spezialisiert. Sie hat zwei Methoden *requestChange* zur Anfrage und *doChange()* zur Durchführung einer gewünschten Änderung bezüglich eines Fehlerreports oder einer Komponente. Die Klasse `user` besitzt zusätzlich die Methoden *createFaultReport()* und *report(Fault)* zur Meldung bzw. Eröffnung eines Fehlerreports (z. B. Trouble Ticket).

SP Rollen

Die Klasse `SPRole` repräsentiert die intraorganisationalen Rollen innerhalb einer Service-Provider-Domäne. An dieser Stelle ist das Attribut `domainID` sehr wichtig um die Domäne, der diese Rollen zugehören, kenntlich zu machen. `SPRole` wird von den Klassen `Dom-SD`, `Dom-FM`, `Dom-TS` und `Dom-MS` spezialisiert.

Die Klasse `Dom-SD` repräsentiert Domänen Sevice Desks. Diese hat folgende Methoden:

- *fwd(Fault)* zur Weiterleitung eines Fehlers.
- Durch die Methode *report(Fault)* wird ein Fehler aus dieser Domäne interorganisational (beispielsweise an den io-SD) gemeldet.
- Durch *getStatus(Fault)* wird der Status der Fehlerbearbeitung zurückgegeben.
- Durch *inform()* werden bestehende Informationen (Zustand der Bearbeitung, Antworten von Domänen usw.) weitergegeben.

Die Klasse `Dom-FM` repräsentiert den Fehlermanager einer bestimmten Domäne mit folgenden Methoden:

- *checkAccessRights()* zur Prüfung der Zugriffsdaten
- *inform()* zur Weiterleitung der zusammengefassten Informationen über das Fehlermanagement innerhalb der Domäne

Die technischen Spezialisten innerhalb einer Domäne sind durch die Klasse `Dom-TS` repräsentiert. Diese besitzt die Methoden:

- *analyze(Fault)* zur Analyse eines vermuteten Fehlers

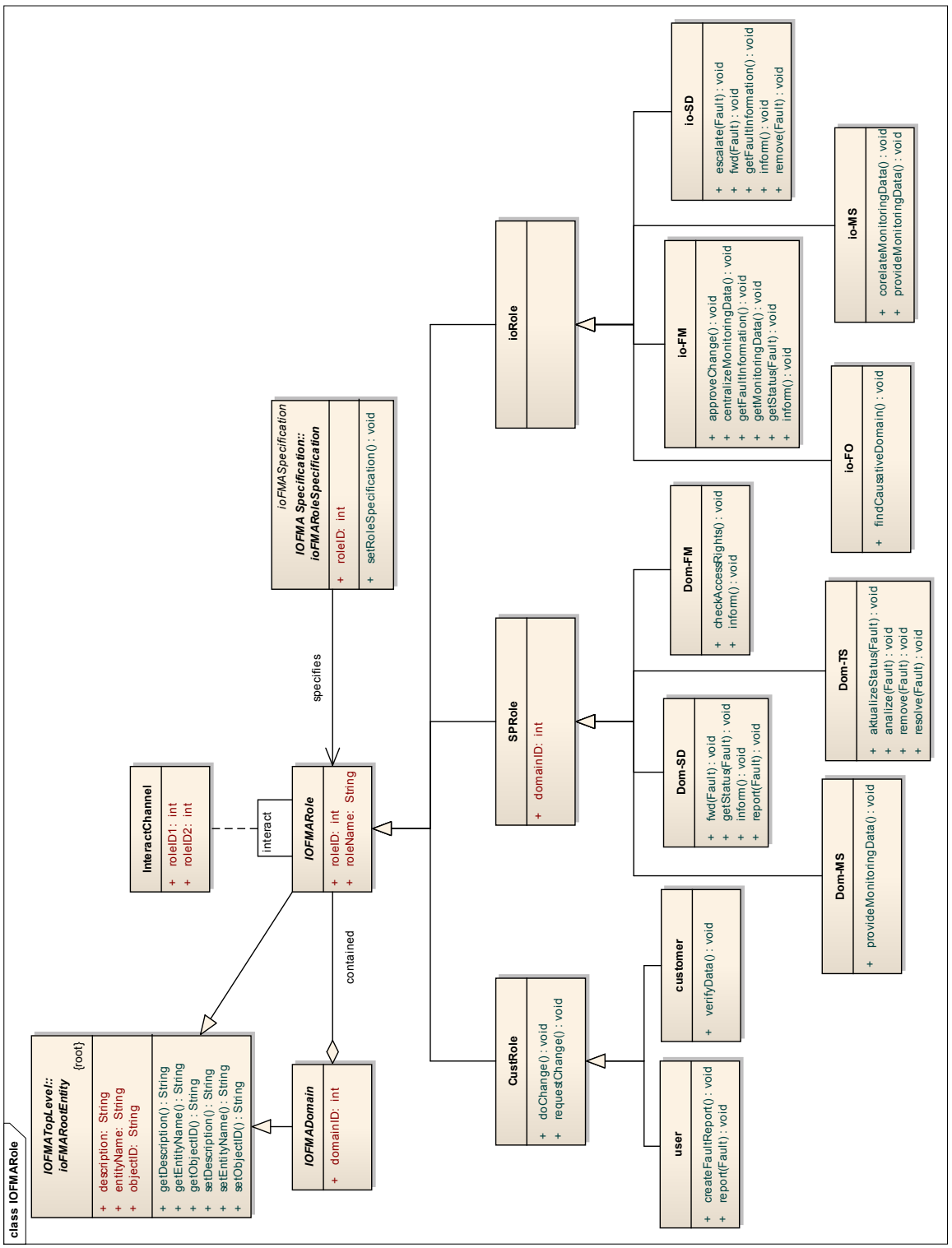


Abbildung 5.19.: Die Domäne `IOFMARole`

- *resolve(Fault)* zur Lösung eines Fehlers
- *remove(Fault)* zur Entfernung eines Fehlers, im Fall eines Fehlalarms
- *actualizeStatus(Fault)* zur Aktualisierung des Status der Fehlerbearbeitung

Die Klasse `Dom-MS` steht stellvertretend für Monitoringsysteme. Über die Methode *provideMonitoringData()* liefert ein Monitoringsystem Monitoringdaten.

io-Rollen Die letzte Kategorie von Rollen ist die der Klasse `ioRole`. Diese wird in vier unterschiedliche Klassen zerlegt.

Die Klasse `io-FM` repräsentiert den interorganisationalen Fehlermanager (wie in Abschnitt 5.2.1 beschrieben). Für diese Klasse sind mehrere Methoden definiert:

- *centralizeData()* fasst die Daten, seien diese Fehlerinformationen oder Monitoringdaten, zusammen und bewertet sie.
- *approveChange()* erteilt die Genehmigung für unterschiedliche Änderungen im System oder in den Daten.
- Mit *getFaultInformation()* werden Fehlerinformationen abgefragt.
- Mit *getMonitoringData()* werden Monitoringinformationen abgefragt.
- *inform()* leitet die zusammengefassten Daten weiter.

`io-SD` ist die Klasse des interorganisationalen Service Desk, mit folgenden Methoden:

- *fwd(Fault)*, leitet einen Fehler zur Lösung zu einer anderen Instanz weiter.
- *escalate(Fault)* eskaliert einen Fehler, wenn dieser sowohl von den lokalen als auch von den bestehenden interorganisationalen Instanzen nicht gefunden oder behoben wurde.
- *getFaultInformation()* dient zur Anforderung von Fehlerinformationen.
- *inform()* dient als Methode zur Weiterleitung von Informationen, entweder an die Domänen oder an den io-FM.
- *remove(Fault)* gestattet mögliche Fehlalarmen zu löschen.

Die Klasse `io-FO` repräsentiert den interorganisationalen Fault-Operator mit der Zusatzmethode *findCausativeDomain()*, die das Lokalisieren einer Störungsursache im Providernetzwerk realisiert.

`io-MS` ist die Klasse der interorganisationalen Monitoringsysteme mit den Methoden *correlateMonitoringData()* zur Korrelation von Monitoringdaten der lokalen Monitoringsysteme und *provideMonitoringData()* für die Bereitstellung von Monitoringdaten.

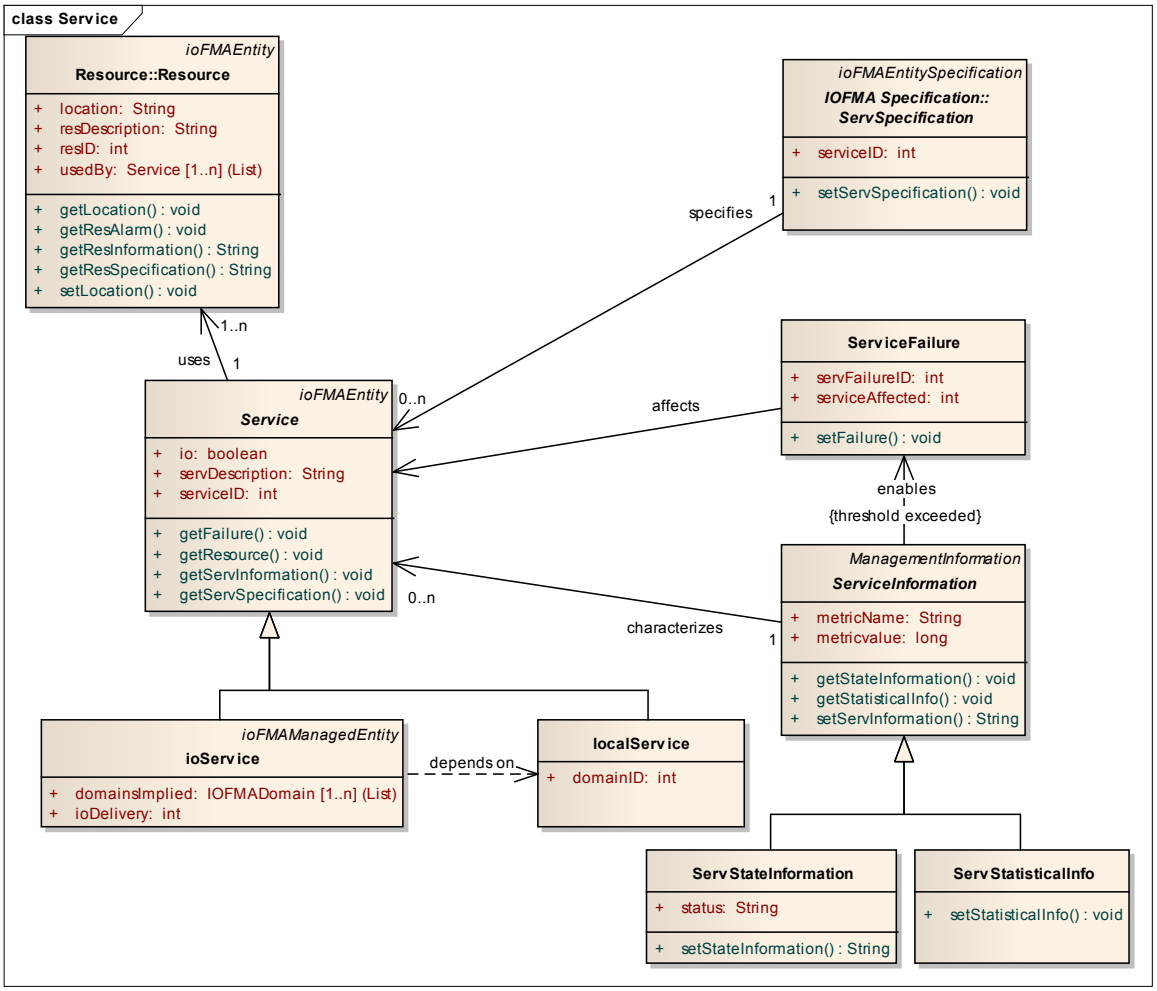


Abbildung 5.20.: Die Domäne `Service`

5.2.3.4. Die Modelldomäne `Service`

Die Domäne `Service` beschreibt, im Kontext des Informationsmodells der ioFMA, den intra- oder interorganisationalen Dienst, der den Kunden zur Verfügung gestellt wird. Die Klasse `Service` ist eine Spezialisierung `ioFMAEntity` mit den Attributen:

- `serviceID` zur eindeutigen Identifikation des Dienstes
- `servDescription` zur Kurzbeschreibung des Dienstes
- `io` zur Differenzierung der Diensterbringung (durch eine Domäne oder interorganisational)

Die Getter-Methoden dieser Klasse stellen relevante Informationen über einen Dienst zur Verfügung. Diese sind folgendermaßen definiert:

- Durch die Methode *getServInformation()* werden Informationen über einen Dienst angefordert.
- *getServSpecification()* stellt die Dienstspezifikation bereit.
- Über *getFailure()* wird die Liste der Ausfälle des Dienstes angefordert.
- Über *getResource()* wird die Liste der Ressourcen des Dienstes angefordert.

Spezialisiert wird diese Klasse durch zwei Klassen von Diensten, `ioService` und `localService`. `ioService` ist zugleich Subklasse von `ioFMAManagedEntity`, `localService` hingegen Subklasse von `ioFMAUnManagedEntity` (siehe Seite 160).

Die Klasse `ioService` besitzt zusätzlich zu der generalisierten Klasse noch ein Attribut `ioDelivery`, das darauf hinweist, ob es sich um einen hierarchischen, einen heterarchischen oder ein gemischten Dienst handelt; und ein mehrwertiges Attribut `domainsImplied` als eine Liste von IDs der IOFMADomains, die an der Erbringung dieses ioService beteiligt sind.

Die Klasse `localService` besitzt ein zusätzliches Attribut `domainID`, welches die ID der Service-Provider Domäne, in dem dieser Dienst erbracht wird, repräsentiert.

Die Klasse `Service` befindet sich in Beziehung mit mehreren anderen Klassen, die Teil der Service Domäne sind, aber teilweise auch Teil folgender Domänen sind: `ServiceFailure`, `ServiceInformation`, `ServSpecification` und `Resource`. Die Klasse `Resource` wird ausführlich bei der Domäne `Resource` besprochen.

Die Klasse `ServSpecification`, als Spezialisierung der Klasse `ioFMAEntitySpecification` in der Domäne `IOFMASpecification`, wird zu einem späteren Zeitpunkt beschrieben. Wichtig ist hier, dass sie sich in *specifies*-Beziehung mit der Klasse `Service` befindet, d. h. durch diese Klasse wird die Spezifikation des Dienstes realisiert.

Die `ServiceFailure`-Klasse befindet sich in Beziehung *affects* (beeinflusst) mit der `Service`-Klasse und besitzt zwei Attribute:

- `servFailureID` als eindeutige Identifizierung des Dienstausfalles
- `serviceAffected` für die Zuordnung des Ausfalls an einen Dienst

Die Klasse `ServiceInformation` charakterisiert einen Dienst durch die Angabe von Zusatzinformationen. Diese Klasse ist ebenso in der Domäne `IOFManagement` als Spezialisierung der Klasse `ManagementInformation` vorhanden.

Innerhalb der Domäne Service repräsentiert die Klasse `ServiceInformation` Informationen bzgl. eines erbrachten Dienstes, die für das Fehlermanagement relevant sind. Die Attribute

`metricName` und `metricValue` repräsentieren die Beziehung einer Kennzahl bzw. den Kennzahlwert, den diese Dienstinformation besitzt. Die drei Methoden werden folgendermaßen eingesetzt:

- *getStateInformation()* liefert Informationen bzgl. des Status.
- *getStatisticalInfo()* liefert statistischen Kennzahlen.
- *setStateInformation()* setzt Managementinformationen, die von der Klasse `Service` benötigt werden.

Die Klasse `ServiceInformation` wird von zwei Klassen spezialisiert: `ServStateInformation`, die den aktuellen Status des Dienstes repräsentiert, und `ServStatisticalInfo`, der statistische Kennzahlen über dem Dienst bereitstellt.

Diese zwei Klassen besitzen Setter-Methoden (*setStateInformation()* und *setStatisticalInfo()*), um die entsprechenden Managementinformationen zu setzen.

Die Klasse `ServStateInformation` zeigt mit Hilfe des Attributs `status`, ob der Dienst voll funktionsfähig (up), ausgefallen (down) oder teilweise betroffen (partial) ist.

5.2.3.5. Die Modelldomäne `Resource`

Die Domäne `Resource` definiert und umfasst alle Klassen der Ressourcen, die für das ioFM relevant sind. Die Domäne besteht aus der Klasse `Resource`, die mit folgenden Klassen in Assoziationsbeziehung steht: `ResourceAlarm`, `ResourceInformation`, `ResSpecification` und `Service`.

Die Klasse `Resource` ist eine Spezialisierung der Klasse `ioFMAEntity` mit zwei Spezialisierungen `ioResource` und `localResource`. Diese hat drei Attribute:

- `resID`: eindeutiger Bezeichner der Ressource
- `resDescription`: Beschreibung der Ressource
- `location`: Die Stelle, an der sich diese Ressource befindet
- `usedBy`: Die Liste der Dienste, die diese Ressource benutzen (mehrwertiges Attribut)

Es werden folgende Methoden definiert:

- *getLocation()* zur Abfrage des Standortes
- *getResInformation()* zur Abfrage der Managementinformationen
- *getResSpecification()* zur Abfrage der Spezifikationen
- *getResAlarm()* zur Abfrage der zusammenhängenden Alarme
- *setLocation()* zur Änderung des Standortes der Ressource

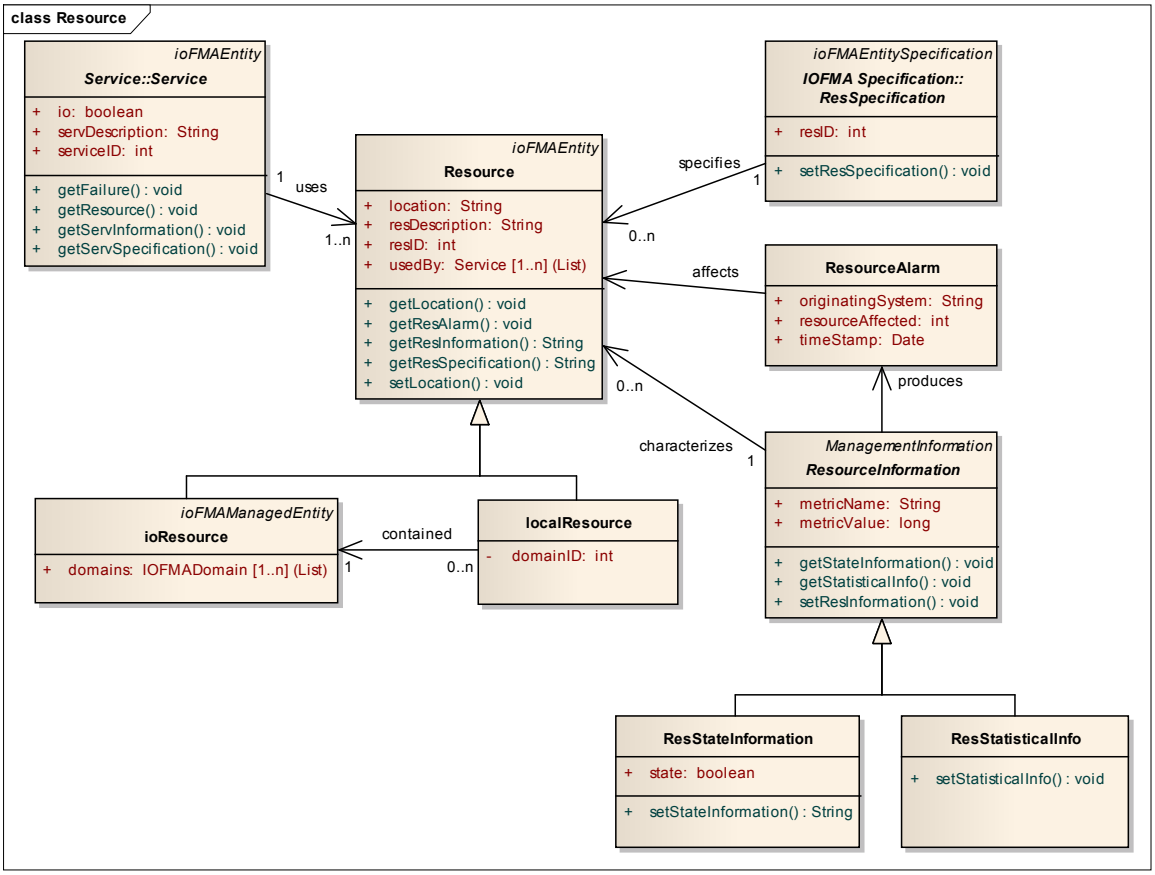


Abbildung 5.21.: Die Domäne `Resource`

Die Klasse `ioResource` spezialisiert die Klasse `Resource` zur Repräsentation von interorganisationalen Ressourcen (wie z. B. ein E2E-Link über mehreren Domänen hinweg). Diese zusätzlich ein mehrwertiges Attribut `domains` als Liste der beteiligten Domänen. Die Klasse `localResource` besitzt das Attribut `domainID`, um die Ressource zu einer bestimmten Domäne abzubilden.

Die Klasse `ResSpecification` spezifiziert die Ressourcen und wird in Domäne `IOFMASpecification` im Detail beschrieben. `ResourceAlarm` repräsentiert die Klasse der Ressourcenalarme mit folgenden Attributen: `resourceAffected`, gibt die ID der betroffene Ressource an, `originatingSystem` gibt das System an, in dem der Alarm aufgetreten ist, und `timeStamp` enthält den Zeitstempel des Auftretens des Alarms.

Die Klasse `ResourceInformation` ist eine Spezialisierung der Klasse `ManagementInformation`, die später beschrieben wird. Sie besitzt zwei Attribute: `metricName` und `metricValue`. Die hier definierten Methoden sind:

- *getStateInformation()* zur Anforderung der Statusinformationen einer Ressource
- *getStatisticalInfo()* zur Anforderung von statistischen Kennzahlen einer Ressource
- *setResInformation()* setzt die Ressourceninformationen, um die gesammelten Daten auf dem aktuellen Stand zu halten

Diese Klasse wird von den zwei Klassen `ResStateInformation` und `ResStatisticalInfo` spezialisiert. Diese zeichnen sich durch jeweils eine Setter-Methode aus:

- *setStateInformation()*, zur Aktualisierung der Statusinformationen der Ressource
- *setStatisticalInfo()*), zur Änderung der statistischen Kennzahlen

Die Klasse `ResStateInformation` besitzt noch ein zusätzliches Attribut `status` zur Anzeige des Status der Ressource als voll funktionsfähig (up), ausgefallen (down) oder teilweise betroffen (partial).

5.2.3.6. Die Modelldomäne `Fault`

Die Domäne `Fault` fasst alle fehlerbezogenen Klassen, die meist schon in den vorherigen Domänen beschrieben wurden, zusammen. Die `Fault`-Klasse ist eine Spezialisierung der `ioFMAEntity`. Sie ist durch mehrere Attribute definiert:

- `faultID` bezeichnet eindeutig einen Fehler
- `description` repräsentiert die Beschreibung des Fehlers
- `faultCategory` ist die Fehlerkategorie, zu der der Fehler gehört
- `priority` zeigt die Priorität des Fehlers (je höher die Priorität, desto wichtiger ist dessen Lösung)
- `originatingSystem` weist auf das System, das den Fehler gemeldet hat, hin
- `faultResState` ist der Status der Fehlerbearbeitung
- `io` zeigt, ob es sich um einen interorganisationalen oder um einen lokalen Fehler handelt
- `firstAlert` der Zeitstempel des ersten Alarms
- `timeRaised` der Zeitstempel der ersten Meldung
- `timeChanged` der Zeitstempel der letzten Änderung

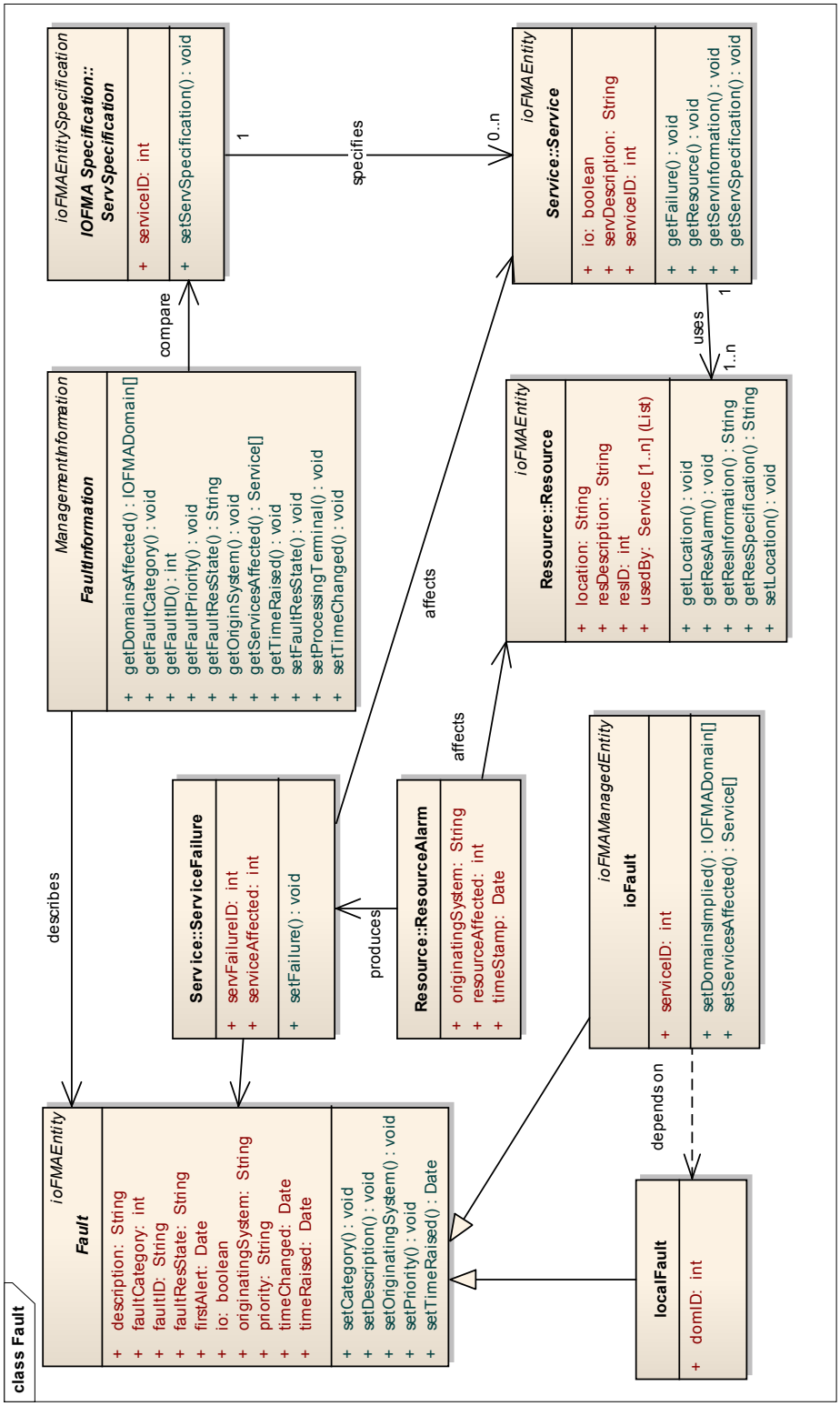


Abbildung 5.22.: Die Domäne `Fault`

Die Klasse `Fault` wird durch `localFault` und `ioFault` spezialisiert. Die Klasse `localFault` repräsentiert einen intraorganisationalen Fehler, der im Kontext der ioFMA als ein nicht-gemanagtes Objekt betrachtet wird (siehe die Domäne `IOFMATopLevel`). Sie besitzt zusätzlich zu der generalisierten Klasse noch das Attribut `domID`, welches auf die Domäne, in welcher der Fehler aufgetreten ist, hindeutet.

Für den interorganisationalen Bereich wird die Klasse `ioFault` definiert, die ein gemanagtes Objekt im Sinne der ioFMA ist. Sie hat das Attribut `serviceID`, das den vom Fehler betroffenen interorganisationalen Dienst kennzeichnet. Die Methoden *setDomainsAffected()* und *setServicesAffected()* werden zur Verfügung gestellt, um die Liste der betroffenen Domäne bzw. Dienste bereitzustellen.

Die Klasse `FaultInformation` liefert und setzt Managementinformationen für die `Fault`-Klasse. Es werden dabei folgende Setter- und Getter-Methoden benutzt:

- *getFaultID()* liefert die Fehler-ID
- *getFaultCategory()* liefert die Fehlerkategorie
- *getFaultPriority()* liefert die Fehlerpriorität
- *getTimeRaised()* liefert den Zeitpunkt der ersten Meldung
- *getOriginSystem()* liefert das meldende System
- *getDomainsAffected()* liefert die Liste der betroffenen Domänen
- *getServicesAffected()* liefert die Liste der betroffenen Dienste
- *getFaultResState()* liefert den Status der Fehlerbearbeitung
- *setFaultResState()* setzt den Status der Fehlerbearbeitung
- *setProcessingTerminal()* setzt das für die Fehlerbearbeitung zuständige Bearbeiter fest
- *setTimeChanged()* setzt den Zeitpunkt, an dem die bestehende Fehlermeldung geändert wurde

Die Klasse `ServSpecification` wird im Rahmen der Domäne `IOFMASpecification` beschrieben. Da die Objekte dieser Klasse Informationen darüber, wie ein Fehler für einen bestimmten Dienst und bestimmten Ressourcen definiert ist, bereitstellen, greift die Klasse `FaultInformation` darauf zu, um die existierenden Informationen mit der Spezifikation zu vergleichen (*compare*) und den Fehler als solchen zu kennzeichnen.

Die Klasse `Fault` befindet sich in Beziehung mit der Klasse `ServiceFailure` aus der `Service`-Domäne und damit auch implizit mit der Klasse `Service`. Da ein Ressourcenalarm normalerweise einen Dienstfehler verursacht, ist die Klasse `ResourceAlarm` der Domäne `Resource` auch in Beziehung (*produces*) mit der `ServiceFailure`-Klasse gesetzt. Im Kontext

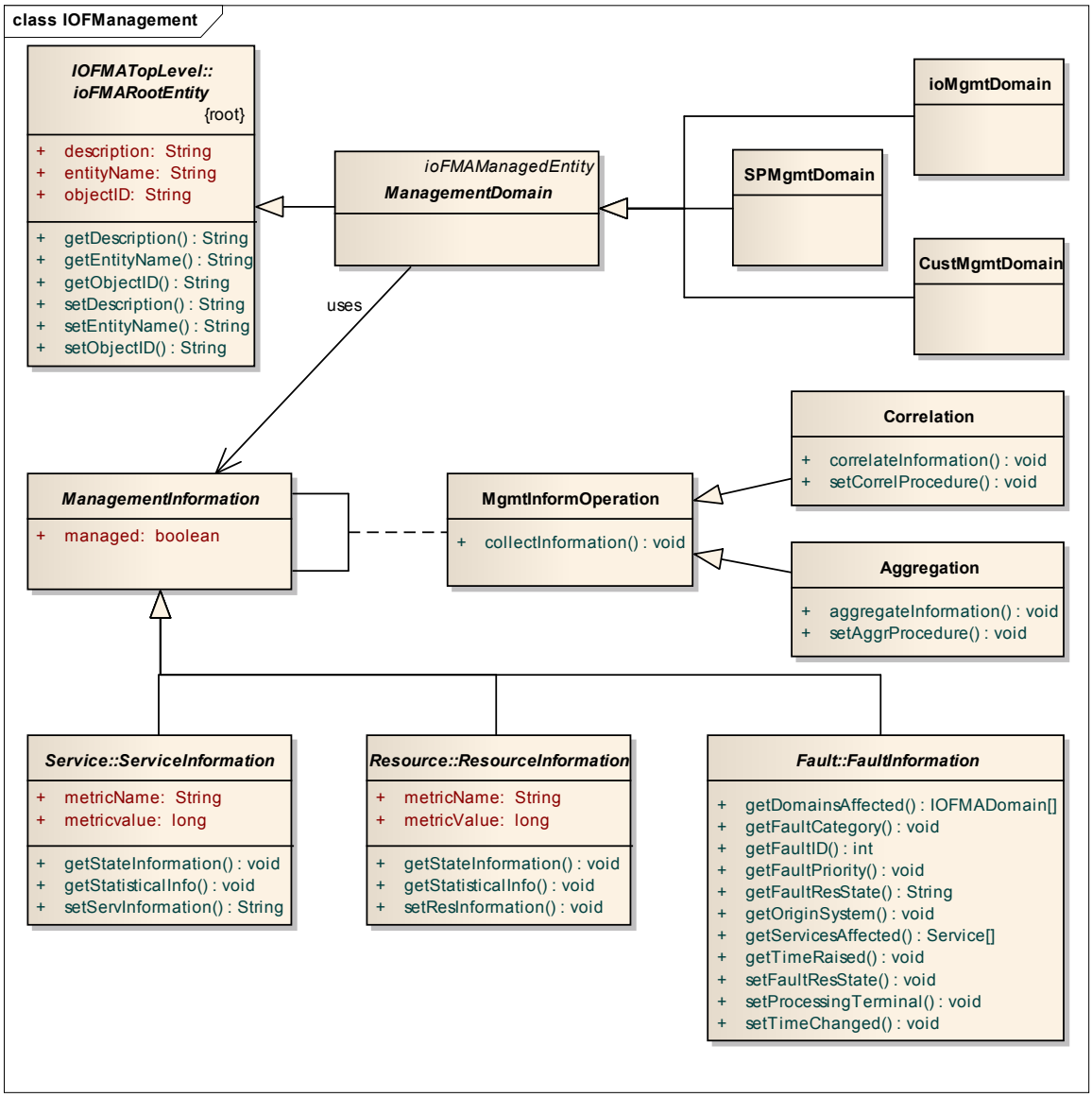


Abbildung 5.23.: Die Domäne `IOFManagement`

des Fehlermanagements spielt auch der Zusammenhang und die Abhängigkeit zwischen einem Dienst und den Ressourcen, die darunter liegen, eine wichtige Rolle. Zwischen den Klassen `Resource` und `Service` befindet sich daher eine gerichtete Assoziationsbeziehung (*produces* – d. h. ein Ressourcenalarm provoziert eine Dienststörung).

5.2.3.7. Die Modelldomäne `IOFManagement`

Die Domäne `IOFManagement` stellt die Fehlermanagement-bezogenen Sachverhalte dar. Die abstrakte `ManagementDomain`-Klasse ist eine Spezialisierung der `ioFMARootEntity`-Klasse. Diese wird wiederum durch die Klassen `CustMgmtDomain`, `SPMgmtDomain` und `ioMgmtDomain` spezialisiert. Diese werden hier nicht mehr weiter ausgeführt, da deren Inhalt fast identisch mit den Bereichen des Funktionsmodells ist.

Die Klasse `ManagementDomain` benutzt die Klasse `ManagementInformation` für die Bereitstellung von Managementinformationen. Ein Attribut `managed` hält fest, ob diese Information einer gemanagten Entität zugeordnet ist oder nicht.

Die Managementinformationen werden nach der Kategorie von gemanagten Objekten in `ServiceInformation`, `ResourceInformation` und `FaultInformation` eingeteilt. Diese Klassen wurden bei den jeweiligen Entitäten-Klassen beschrieben.

Mehrere Managementinformationen können zusammengefasst werden. Dies wird im Modell als eine Assoziationsklasse (`MgmtInformOperation`) repräsentiert. Diese Klasse besitzt die Methode *collectInformation()*, um Informationen aus den Klassen (`ServiceInformation`, `ResourceInformation` und `FaultInformation`) zu sammeln und diese entsprechend zusammenzufassen. Die zwei Klassen `Correlation` und `Aggregation` spezialisieren die `MgmtInformOperation`-Klasse.

Die `Correlation`-Klasse besitzt zwei Methoden: *setCorrelProcedure()*, die den Korrelationsalgorithmus festlegt, und *correlateInformation()*, die anhand des Algorithmus die Informationen tatsächlich korreliert. Ein solch möglicher Korrelationsmechanismus ist in [MSGL 09a] (intraorganisational) und [MSGL 09] (interorganisational) ausführlich beschrieben.

Die andere Operationsklasse (`Aggregation`) besitzt ebenfalls zwei Methoden, *setAggrProcedure()* und *aggregateInformation()*, um die Aggregationsmethode festzulegen und sie dann auf die vorliegenden Informationen anzuwenden. Es wurden an dieser Stelle nur zwei Klassen von Operationen beschrieben, aber es können nach Bedarf weiter Spezialisierungen definiert werden.

5.2.3.8. Die Modelldomäne `IOFMASpecification`

Die Domäne `IOFMASpecification` stellt die gesamten Spezifikationen, die in der ioFMA benötigt werden, bereit. Die abstrakte Klasse `ioFMASpecification` wird über folgende Attribute definiert:

- `specID` eindeutiger Bezeichner der Spezifikation
- `specName` eindeutiger Name der Spezifikation
- `specValidFor` die Zeitspanne, in der diese Spezifikation gültig ist

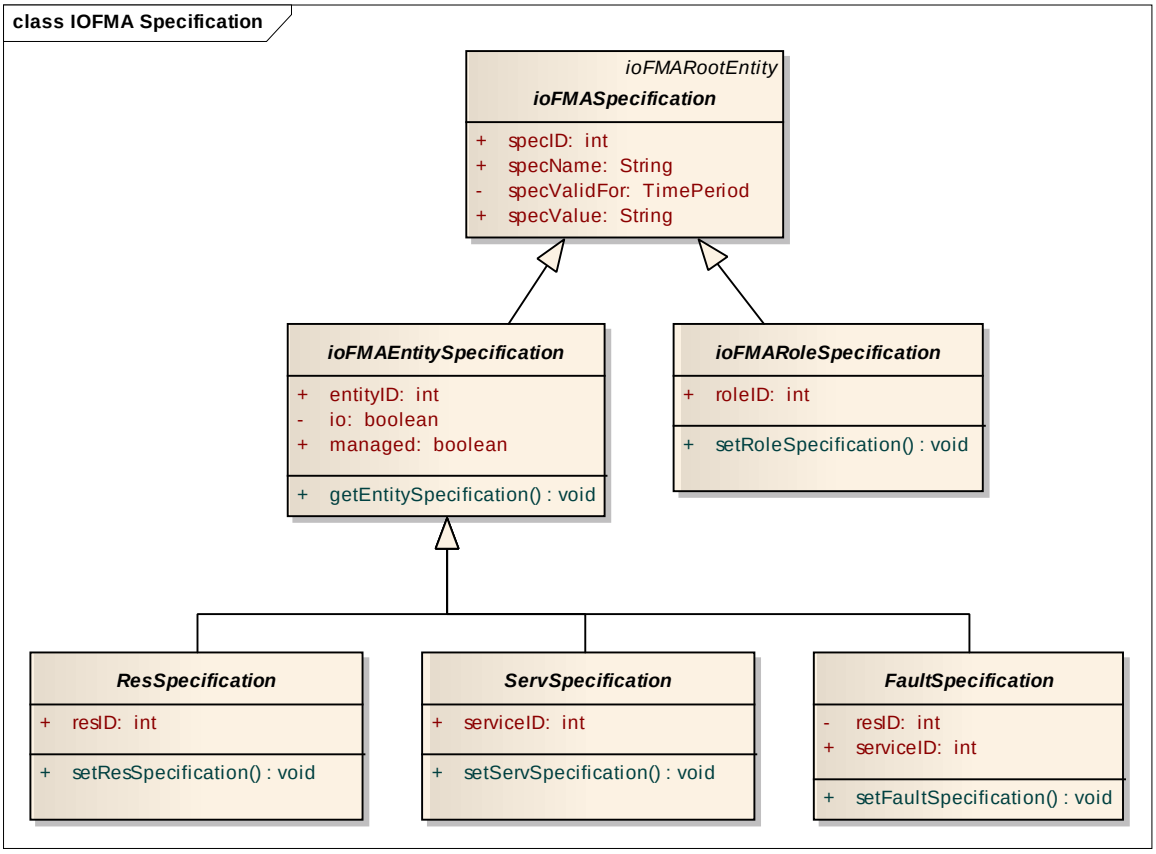


Abbildung 5.24.: Die Domäne `IOFMASpecification`

- `specValue` der tatsächliche Inhalt der Spezifikation, z.B. einer Dienstspezifikation im Servicekatalog

Die Klasse `ioFMASpecification` wird von den Klassen `ioFMAEntitySpecification` und `ioFMARoleSpecification` spezialisiert.

Die Klasse `ioFMARoleSpecification` repräsentiert Rollenspezifikationen. Sie hat ein Attribut `roleID`, um die Rolle für die Spezifikation zu kennzeichnen, und eine Methode *setRoleSpecification()*, um diese Spezifikation festzulegen.

Die Klasse `ioFMAEntitySpecification` spezialisiert ebenfalls die Klasse `ioFMASpecification` für Entitätenspezifikation. Sie besitzt noch zusätzlich die Attribute `entityID` und `managed`, um auf die zu spezifizierende Entität und ob diese gemanagt ist oder nicht, hinzuweisen. Die hier definierte Methode *getEntitySpecification()* ruft Spezifikationen unterschiedlicher, miteinander verbundenen Entitäten ab. In den jeweiligen spezialisierenden Klassen `ServSpecification`, `ResSpecification`, `FaultSpecification` werden entsprechenden Setter-Methoden definiert.

Die Klasse `ResSpecification` besitzt ein Attribut `resID`, mit der eindeutigen ID der Ressourcen, die zu spezifizieren sind. Die Methode *setResSpecification()* stellt die Ressourcenspezifikation bereit.

Die Klasse `ServSpecification` besitzt ein Attribut, um den Service, dessen Spezifikation bereitgestellt wird, eindeutig zu identifizieren (`serviceID`). Zur Bereitstellung der Spezifikation steht die Methode *setServSpecification()* zur Verfügung.

Für die Spezifikation von (möglichen) Fehlern wird die Klasse `FaultSpecification` benutzt. Sie beinhaltet die Attribute `resID` und `serviceID`, die die Ressource bzw. den Dienst, die von einem möglichen Fehler betroffen werden könnten, eindeutig kennzeichnen.

5.2.3.9. Interoperabilität mit bestehenden Informationsmodellen

Das ioFMA-Informationsmodell (die ioFMA-MIB) beinhaltet Entitäten und deren Beziehungen, die im interorganisationalen Umfeld für das Thema Fehlermanagement von Bedeutung sind. Eine wesentliche Voraussetzung für die Nutzung des ioFMA-Informationsmodells ist aber seine Interoperabilität mit bereits verwendeten Schemata, wie z.B. dem CIM(Common Information Model) bzw. Standards, oder dem SID (Shared Information Data Model).

Interoperabilität der ioFMA mit CIM

Das CIM [CIM 10] stellt ein objektorientiertes Informationsmodell für Managementzwecke bereit. Das Ziel ist die Unterstützung des plattform- und technologieunabhängigen Austauschs von Managementinformationen für Computer-Netze, Systeme und Dienste und die Gewährleistung der Kooperation von Managementsystemen. Grundsätzlich sind bei einer möglichen Interoperabilität der ioFMA mit CIM folgende Punkte zu beachten:

- Da CIM einen sehr strikten Spezialisierungs- und Subklassenansatz benutzt, kann dieser mit Erweiterungsklassen der ioFMA angereichert werden. Es handelt sich um einen aufwändigen Erweiterungsmechanismus (vgl. [KHS 01]), der ohne ein streng angewandtes Regelwerk nicht leicht durchzuführen ist.
- CIM kennt keine Assoziationsklassen. Im Gegensatz dazu benutzt die ioFMA Assoziationsklassen zur Modellierung der Abhängigkeiten.
- An manchen Stellen bei CIM wird ein sogenannter ,,UML-Dialekt" und das Managed Object Format(MOF), benutzt um bestimmte Beschreibungen zu realisieren. Die Modellierung der ioFMA hingegen entspricht vollständig dem UML-Standard.
- CIM ist fokusiert auf die Betriebsphase. Die ioFMA ist übergreifender, da sie mehr oder weniger in allen Phasen des Lebenszyklus benutzt wird.
- CIM sieht in seinem Schema nur intraorganisationale Sachverhalte vor. Das ioFMA ist per Definition in interorganisationalen Umgebungen einsetzbar.

- In der ioFMA wird eine spezielle Entität `Fault` definiert, CIM kennt diese nicht.

Angesichts der oben genannten Punkte ist die Interoperabilität der ioFMA mit CIM durch die Allgemeingültigkeit der ioFMA gegeben. Größtenteils der Klassen der ioFMA können zu CIM portiert werden.

Interoperabilität der ioFMA mit SID

Das von SID dargestellte Informationsmodell ist auf Managementprozesse ausgerichtet, aber auch mehr auf die Anforderungen des Telekommunikationbereiches zugeschnitten [BGSS 06]. Da aber die Grundprinzipien von SID auf CIM-Schemata beruhen, ist eine klare Differenzierung der System- und Geschäftsprozessorientierten Managementinformationen nicht gegeben. Der Vorteil vom SID ist, die Schwächen von CIM durch eine mächtigere Ausdrucksweise zu kompensieren. In diesem Sinne ist die Einordnung der ioFMA in SID viel einfacher zu realisieren als in CIM, da beide auf denselben Modellierungsparadigmen beruhen. Da SID auf CIM beruht ist hier durch Transitivität die Interoperabilität auch gegeben.

5.2.3.10. Besonderheiten des Informationsmodells bzgl. Diensterbringungsformen

Die Anwendung des ioFMA-Informationsmodells für die hierarchische bzw. heterarchische Form der Diensterbringung ist leicht unterschiedlich, dies ist auf Unterschiede im Organisations- bzw. Funktionsmodell zurückzuführen. Insbesondere in der Domäne Role zeigen sich Unterschiede in der Anwendung. Als Unterklasse von `ioRole` kommt bei heterarchischen Formen die Klasse `End-Site-SD` hinzu. Zusätzliche Abweichungen ergeben sich in der Implementierung der Methoden für die beiden Formen der Diensterbringung. Diese sind aber auch nur auf die Unterschiede im Funktionsmodell zurückzuführen. Andere Differenzen in den Klassen des Informationsmodells bestehen nicht.

5.2.4. Das Kommunikationsmodell

Das Kommunikationsmodell ist derjenige Teil einer Managementarchitektur, der die Konzepte zum Austausch von Managementinformationen zwischen den Akteure festlegt [HAN 99]. Bei der Realisierung des Kommunikationsmodells der ioFMA müssen folgende Aspekte betrachtet werden:

- Welche Entitäten tauschen miteinander Managementinformationen aus?
- Wie sehen die Austauschformate für das Managementprotokoll aus?
- Welche zusätzlichen Dienste werden zur Unterstützung der Kommunikationsmechanismen angeboten?

5.2.4.1. Generisches Kommunikationsmodell

In Abschnitt 5.2.2 wurde bei der Beschreibung der unterschiedlichen Funktionen erläutert, dass verschiedene Nachrichten zwischen den Rollen ausgetauscht werden müssen, um die Funktionen durchzuführen. Das ioFMA-Kommunikationsmodell basiert auf einem Manager/Agenten-Ansatz (wie in [HAN 99] beschrieben). Die in dem Organisations- bzw. Informationsmodell beschriebenen Rollen nehmen je nach Bedarf die Rolle des Managers oder des Agenten, an. Dabei wird über die Interaktionskanäle kommuniziert (siehe Abschnitt 5.2.1). Ebenso wurden im vorherigen Abschnitt 5.2.3 bei den Klassendefinitionen innerhalb des Informationsmodells unterschiedliche Interaktionen eingeführt.

- *create*: um Managementobjekte (Fehlermeldungen) zu generieren
- *get*: um Managementinformationen vom Managementobjekte zu lesen
- *set*: um Managementinformationen zu setzen (zu ändern)
- *inform*: um andere Managementinstanzen zu informieren
- *fwd*: um Managementinformationen weiterzuleiten
- *provide*: um Managementinformationen zur Verfügung zu stellen
- *remove*: um Managementinformationen zu entfernen
- *update*: um Managementinformationen (regelmäßig) zu aktualisieren

Um die im Funktionsmodell beschriebenen Managementfunktionen zu realisieren, müssen Basisdienste wie ein AA-Dienst und ein Ereignisdienst im Kommunikationsmodell definiert werden.

Authentifizierung und Autorisierung

Der AA-Dienst ist im Rahmen der ioFMA für die Autorisierung und Authentifizierung von User oder anderen Rollen innerhalb einer an der Diensterbringung beteiligten Domäne zuständig. Unter diesem Gesichtspunkt muss die ioFMA diesen Dienst nicht neu erfinden, sondern kann bestehende Ansätze benutzen. Hier sind die Realisierungen in [Homm 07] (in Form eines föderierten Identitätsmanagementwerkzeugs), in [Lang 01] (unterstützend für das Customer Service Managememnt (CSM) System) oder in [Schif 07] (als Security und Restriction Service (SRS) für das Management virtueller Organisationen) zu erwähnen. Mit Hilfe dieses Dienstes wird sichergestellt, dass eine bestimmte Rolle gegenüber ioFMA authentifiziert wird und zusätzlich für den Zugang in bestimmten administrativen Domänen bzw. auf bestimmten Informationen autorisiert wird.

Um einen Interaktionskanal (siehe Abschnitt 5.2.1.3) nutzen zu können, muss sich eine bestimmte Rolle erst gegenüber der Gegenseite authentifizieren. Dies kann man beispielhaft anhand des Interaktionskanals PI1 darstellen (siehe Abbildung 5.25). Dieser Interaktionskanal ist unter anderem für die Abholung von Monitoringdaten von den Domänen (z.B. im Fall der Gesamtüberwachung) zuständig. Der io-FM sendet seine eigene sowie die Identität des jeweiligen Aufrufers (*caller*) – z. B. user, ein Dom-SD usw. – an den Dom-FM der Domäne

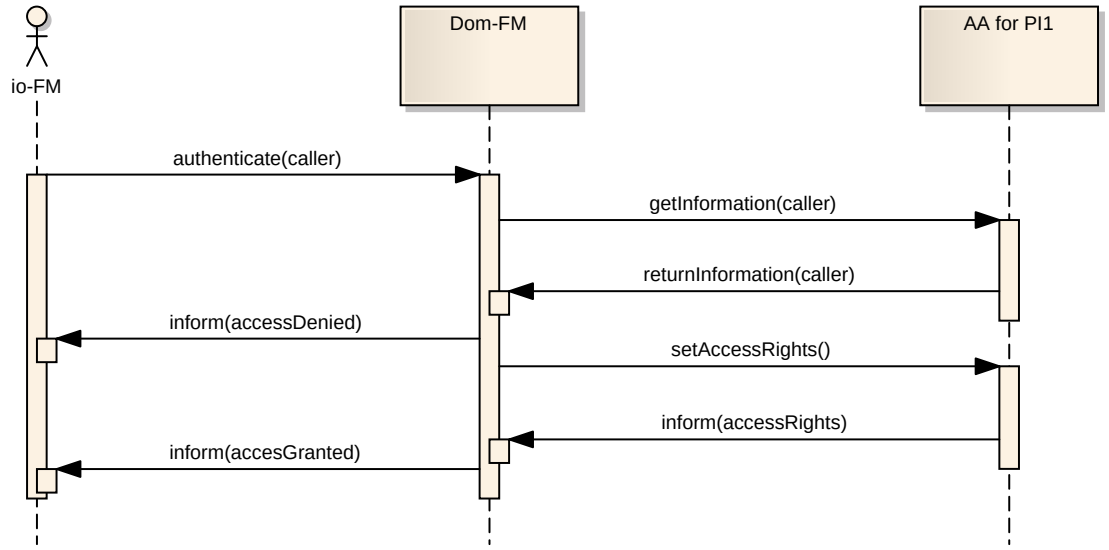


Abbildung 5.25.: Der AA-Dienst: Verwendung für den Interaktionskanal PI1

auf die der Zugriff stattfinden soll. Diese Anfrage wird dann zum AA-Dienst weitergeleitet. Dieser überprüft, ob die ID bzw. Rolle (und deren Zuordnung) gültig sind. Wenn diese Prüfung fehlschlägt, wird dem io-FM mit einem Authentifizierungsfehler geantwortet. Wenn aber die Authentifizierung erfolgreich war, legt der Dom-FM die Zugriffsrechte für den caller fest. In Einklang mit diesen Zugriffsrechten kann dementsprechend ein Dienst benutzt oder die Benutzung unterschiedlicher Daten gesteuert werden.

Ereignisdienst

Im Fall des Ereignisdienstes ist es wichtig zu beachten, dass unterschiedliche Rollen zeitnah über den Ausfall von Diensten oder Ressourcen, über Monitoringdaten oder über andere mögliche Fehler, die auftreten könnten, benachrichtigt werden. Dies geschieht über die Interaktionskanäle. Die Bereitstellung dieser Informationen wird entweder über synchrone (PULL) oder asynchrone (PUSH) Benachrichtigungsmechanismen realisiert.

5.2.4.2. Besonderheiten des Kommunikationsmodells bzgl. Diensterbringungsformen

Die unterschiedlichen Kommunikationsstrukturen in Hierarchien und Heterachien wurden schon in den Abschnitten 2.1.3.3 und 2.1.3.4 eingeführt. Der Aufbau dieser Kommunikationsstruktur ist dabei ausschlaggebend. Als Teil des Koordinationsmusters wurden die grundlegenden Kommunikationsstrukturen (Baum-Struktur, Stern-Struktur und Peer-to-Peer-Struktur) schon in Abschnitt 2.1.3.2 beschrieben. Eine Hierarchie weist eine Baum-Struktur auf, es gibt eine ,,Wurzel"-Organisation, die mit einer oder mehreren Organisationen kommuniziert. Diese kommunizieren wiederum mit mindestens einer Organisation. Zyklen sind

im Baum nicht zulässig. In einer Heterarchie gibt es keine einheitliche Ordnung. Es gibt vielfältige Kommunikationsbeziehungen zwischen den Organisationen, die zudem unabhängig von der Dimension der Steuerung sind. Peer-to-Peer- und Stern-Strukturen können hier also nebeneinander existieren.

Da die Kommunikationsstruktur die Instantiierung des Kommunikationsmodells leitet, gibt es entsprechende Unterschiede beim Einsatz des Kommunikationsmodells für Heterarchie und für Hierarchie. Als Beispiel kann der Ereignisdienst betrachtet werden: Bei einer hierarchischen Form der Diensterbringung mit einem PULL-Benachrichtigungsmechanismus wird die Information eines Providers (oder eines Managementsystems) nur dem damit verbundenen Providern zur Verfügung gestellt. Im heterarchischen Fall werden Ereignisinformationen in der Regel allen beteiligten Domänen mitgeteilt. Der PUSH-Benachrichtigungsmechanismus weist eine einheitliche Anwendung im Fall der Hierarchie und Heterarchie auf. Dabei ruft eine bestimmte Domäne von einer andern Domäne(n) Ereignisinformationen ab.

5.3. Diskussion

Das Ziel, das bei der Entwicklung der ioFMA verfolgt wurde, war, einen Rahmen in Form eines Baukasten-ähnlichen modularen Systems zu definieren. Die einzelnen Teile des Rahmens, die die jeweiligen Managementbereiche abdecken, können möglichst flexibel und interoperabel zusammengesetzt in jeder Umgebung angepasst werden, um die jeweils bestmögliche Managementlösung zum Einsatz zu bringen. Der erste Schritt in Richtung eines integrierten Fehlermanagements für heterogene interorganisationale Umgebungen wurde mit der Entwicklung des plattformunabhängigen PIM der ioFMA realisiert. Folgende Aspekte wurden dabei in herstellerübergreifender Weise spezifiziert [HAN 99]:

- Beschreibung von Managementobjekten (vgl. das Informationsmodell in Abschnitt 5.2.3)
- Behandlung und Unterstützung von Organisationsaspekten, Rollen und Kooperationsformen (vgl. das Organisationsmodell in Abschnitt 5.2.1)
- Beschreibung von Kommunikationsvorgängen zu Managementzwecken (vgl. das Kommunikationsmodell in Abschnitt 5.2.4)
- Strukturierung der Managementfunktionalität (vgl. das Funktionsmodell in Abschnitt 5.2.2)

Die in Abschnitt 3.2 ausführlich geschilderten Anwendungsfälle, die zu den Anforderungen an eine ioFMA führten, sind in Tabelle 3.19 und 3.20 zusammengefasst. Durch die Definition der plattformunabhängigen ioFMA, sind viele der in diesen Tabellen genannten Anforderungen erfüllt oder teilweise erfüllt.

Tabelle 5.11 fasst die Erfüllung der Anforderungen zusammen. Das Organisationsmodell betrachtet alle möglichen Rollen und Beziehungen, sowie die Einteilung in Domänen. Darüber hinaus stellt es unterschiedliche Modelle für die beiden interorganisationalen Formen der Diensterbringung bereit. Damit sind die Anforderungen an das Informationsmodell erfüllt.

Die Anforderungen an das Funktionsmodell sind in hohen Grad erfüllt. Das ist auf die Systematik im Aufbau des Funktionsmodells zurückzuführen, das auf den in der Anforderungsanalyse beschriebenen Anwendungsfällen basiert. Trotzdem gibt es einige Ausnahmen: Die Anforderungen FM-R01, FM-R02, FM-R03 (bzgl. Reporting) und FM-02 (Datenänderung) sind nur teilweise durch die entsprechenden Funktionen erfüllt, die nur grob beschrieben sind und eine Erweiterung benötigen (beispielsweise auf eine spezifische Plattform). Die Anforderung FM-01 (Visualisierung) wurde in dieser Phase der Entwicklung noch nicht betrachtet und sie ist deswegen auch nicht erfüllt.

Die an das Informationsmodell gestellten Anforderungen sind nur zu einem gewissen Grad erfüllt. Die Schnittstellendefinition (IM-03) wurde über die ausführliche Beschreibung des `InteractChannel`-Konstrukts realisiert. Ebenso wurden durch das entwickelte Informationsmodell alle Phasen des Fehlerlebenszyklus unterstützt (IM-04). Weder ein gemeinsames Datenformat (IM-01) noch Konvertierungsmethoden (IM-02) wurden durch das ioFMA-PIM realisiert, da diese von der Umgebung, in der die ioFMA eingeführt wird, abhängig sind. Da diese aber ein wichtiger Bestandteil des Informationsmodells und der ioFMA allgemein sind, werden diese in einer plattformspezifischen Abbildung der ioFMA miteinbezogen. Die Anforderungen IM-05 (Standard-Metriken) und IM-06 (Aggregationsfunktionen) wurden bislang nur erwähnt und kurz angeschnitten, eine ausführlichere Betrachtung dieser Fälle bleibt für die plattformspezifische ioFMA bzw. für deren Implementierung offen.

Beim Kommunikationsmodell der ioFMA-PIM wurden die Kommunikationsmechanismen und Kommunikationsprotokolle nicht betrachtet, da diese ebenfalls stark von der Plattform, in der die ioFMA einzuführen ist, abhängig sind. Die Interdomänen-Kommunikation wurde jedoch betrachtet, wenn auch nicht vollständig im Kommunikationsmodell, sondern lediglich im Funktions- und Informationsmodell.

Bezüglich der nicht-funktionalen Anforderungen ist erkennbar, dass diese nicht oder oder nur teilweise erfüllt wurden. Diese Kategorie von Anforderungen ergänzt die funktionalen Anforderungen, deren Erfüllbarkeit oben beschrieben wurde. Diese hängen sehr stark von der Plattform, in der die ioFMA einzuführen ist, und deren tatsächlicher Implementierung ab. Erfüllt ist die Anforderung NF-02 (Zugriffskontrolle), die jedoch beim Kommunikationsmodell eingeführt wurde, da sie als Grundlage eine Interdomänen-Kommunikation hat. Die Skalierbarkeit und eine gemeinsame Informationsdatenbank sind nur indirekt in den Teilmodellen angeschnitten worden, diese werden für den plattformspezifischen Fall ausführlicher betrachtet. Die Anforderungen NF-02 (Datenintegrität), NF-03 (Datenaktualität), NF-05 (Performanz), NF-06 (Automatisierung) und NF-08 (Dokumentation) werden durch das ioFMA-PIM nicht erfüllt.

Anforderungen		Phasen des Lebenszyklus eines Fehlers			
		Entdecken	Eingrenzen	Beheben	Vorhersage
an IM	IM-01		–		–
	IM-02	–	–		–
	IM-03	+	+	+	
	IM-04	+	+	+	+
	IM-05	0			0
	IM-06	0	0		0
an OM	OM-01	+	+	+	+
	OM-02	+	+	+	+
an FM	FM-L01	+	+		
	FM-L02	+			+
	FM-L03	+	+		
	FM-P01		+	+	
	FM-P02			+	
	FM-M01	+	+	+	+
	FM-M02	+	+	+	+
	FM-M03	+	+		+
	FM-R01	0			0
	FM-R02	0			0
	FM-R03				0
	FM-F01		+		+
	FM-F02			+	+
	FM-01	–	—		–
	FM-02		0	0	0
an KM	KM-01	–	–	–	–
	KM-02	0	0	0	0
	KM-03	–	–	–	–
NFA	NF-01		+	+	
	NF-02		–	–	
	NF-03	–			–
	NF-04	0	0	0	0
	NF-05		–	–	
	NF-06	–	–	–	
	NF-07		0	0	0
	NF-08		–	–	

Legende:

Anforderung ist:
+ erfüllt
0 teilweise erfüllt
- nicht erfüllt

Tabelle 5.11.: Erfüllung der Anforderungen durch die ioFMA-PIM

Durch das ioFMA-PIM wurde die Fehlermanagementdomäne vollständig plattformunabhängig gestaltet. Allerdings wird ein plattformspezifisches Modell benötigt, um einige der verbliebenen Anforderungen erfüllen zu können. Es wurde die Mächtigkeit des MDA-Ansatzes und die Möglichkeit der Realisierung einer plattformunabhängigen und einer darauf aufbauenden plattformspezifischen ioFMA genutzt. Der nächste Schritt, der in dieser Arbeit folgt, ist die Transformation des ioFMA-PIM auf das ioFMA-PSM. Es gibt natürlich abhängig von der ausgewählten Plattform (J2EE, Web Services, .NET usw.) (vgl. [Lope 05]), mehrere PSMs für das hier entwickelte ioFMA-PIM.

Die Transformation der Modelle kann innerhalb der MDA durch das Konzept Plattform definiert werden. Mit Hilfe einer Plattform lassen sich Transformationen flexibel beschreiben und durchführen. Modelltransformationen dienen letztendlich der schrittweisen Entwicklung von Software. Diese Eigenschaft ist auch diejenige, die bei der Modellierung der ioFMA ausschlaggebend war: Die ioFMA konnte erst allgemein dargestellt (PIM) und dann mit Hilfe einer Plattform auf ein PSM transformiert werden. Die Plattform ist unabhängig von einer konkreten Anwendung, die die Funktionaliät dieser Plattform nutzt [PeMe 06].

Die Auswahl der Plattform ist allerdings vom Anwendungs- bzw. Aufgabenbereich abhängig. Im Fall des interorganisationalen Fehlermanagements ist das Attribut ,,interorganisational" ausschlaggebend. Die Plattform, auf der die ioFMA entwickelt werden soll, muss übergreifend auf mehrere organisatorisch unterschiedliche Domänen zugreifen können, muss eine bestimmte einheitliche Datenmodellierung anbieten, muss ,,online" erreichbar, leicht wartbar und erweiterungsfähig sein. Für das Fehlermanagement ist zusätzlich eine koordinierte, auf einem gemeinsamen Datenformat basierende, zuverlässige Datensammlung der Domänen obligatorisch. Im interorganisationalen Bereich (Grids und in letzter Zeit Clouds) haben sich Web Services in den letzten Jahren mehr und mehr als Implementierungstechnologie durchgesetzt. Web Services können sehr leicht durch die gemeinsame Beschreibungssprache Web Services Description Language (WSDL) miteinander ,,gekoppelt" werden. WSDL ist eine XML-basierte, maschinell lesbare Sprache, die die angebotenen Funktionen (Methoden und Parameter) beschreibt und wie man darauf zugreift. Für Web Services spricht also auf der einer Seite die einheitliche Dienstbeschreibungssprache (WSDL) und auf der anderen Seite die automatisierte, auf XML basierende Nachrichtenvermittlung mit Hilfe des Simple Object Access Protocol (SOAP). Darüber hinaus wird die Möglichkeit geboten, automatisierte Interaktionen zwischen Applikationen auf verschiedenen Servern, an verschiedenen Standorten und in unterschiedlichen Domänen, zu realisieren. Im Sinne des interorganisationalen Einsatzes von Tools vereinfachen SOAP-Implementierungen die Programmierung, da von einzelnen HTTP-Aufrufen und XML-Codierungen abstrahiert wird. Aus diesem Grund wurden Web Services als spezifische Plattform für die Transformation auf das ioFMA-PSM gewählt.

5.4. Zusammenfassung

Dieses Kapitel hat sich mit der Erstellung der vier Teilmodelle der interorganisationalen Fehlermanagementarchitektur (ioFMA), basierend auf den in Kapitel 3 abgeleiteten Anforderungen, beschäftigt. Als erstes befasste sich das Organisationsmodell mit der Definition von Domänen und Rollen im interorganisationalen Fehlermanagement. Es wurde zwischen lokalen, interoganisationalen und kundenspezifischen Rollen (und Domänen) unterschieden. Es folgte das Funktionsmodell, das die Managementbereiche und die Managementfunktionen der ioFMA, bezogen auf die Anwendungsfälle aus Kapitel 3, definiert. Im Informationsmodell wurden die Managementobjekte in Form von Entitäten (Dienste, Ressourcen, Fehler) und anderen Objekten, die im interorganisationalen Fehlermanagement mitwirken, beschrieben. Das Kommunikationsmodell legt die Grundlagen für den Austausch der Managementinformationen fest.

Das hier entwickelte Modell repräsentiert das PIM im Sinne des MDA-Ansatzes sowie die Architektursicht der ioFMA. Die Abdeckung der Anforderungen an die ioFMA durch das PIM wurden aufgewiesen. In den nächsten Kapiteln werden durch das zu realisierende ioFMA-PSM sowie eine bestimmte Instantiierung noch weitere Anforderungen realisiert: In Kapitel 6 wird die Transformation auf eine Plattform sowohl allgemein beschrieben, als auch die PSM-spezifische Umsetzung auf Basis der WSDL erklärt. Darauf aufbauend werden in Kapitel 7 Implementierungsaspekte des PSM für perfSO/-NAR (vgl. Abschnitt 4.1.1) vorgestellt.

6

Plattformspezifische Transformation der Architektur

Inhalt des Kapitels

Im vorherigen Kapitel wurden innerhalb der Architektursicht die Teilmodelle der ioFMA als plattformunabhängige Spezifikation beschrieben (ioFMA-PIM). In diesem Kapitel wird eine plattformspezifische Sicht auf der ioFMA anhand einer geeigneten Transformation gegeben.

Die Transformation des PIM auf ein PSM wird anhand der in [GPR 06] beschriebenen Methodik realisiert. Laut dieser sind alle Modelle allgemein als Instanzen eines Metamodells aufzufassen. Um eine Transformation zu realisieren, wird eine Beschreibung benötigt, in der klargestellt wird, wie die Elemente des neuen Modells mit den Elementen des ursprünglichen Modells in Beziehung stehen. Die Menge dieser Beziehungen sind hier (vgl. Abbildung 6.1) als Abbildung (Mapping) zwischen den zwei Modellen benannt. Die Regeln eines solchen

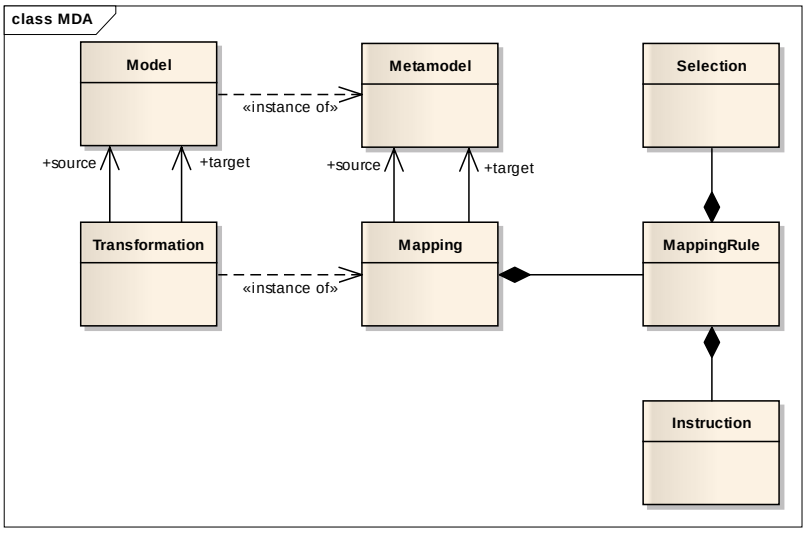


Abbildung 6.1.: Die Transformation vom PIM auf PSM [GPR 06]

Mappings bestehen aus Bedingungen, die ein Teil der Elemente (*Selection*s) aus dem Metamodell erfüllen müssen, und Anweisungen (*Instruction*s), die auf dieser Menge an Elementen durchzuführen sind.

Dieses Kapitel befasst sich hauptsächlich mit der Beschreibung der Transformation und der Methodik. Zuerst sind ein paar Grundlagen bezüglich des Transformationsmodells in MDA einzuführen. Die Problematik der Abbildung bzw. von Transformation wurde gründlich in [Lope 05, LHBJ 05, SLCA 09, SH 10] u. a. untersucht; ebenfalls wurden semi-automatisierte Mechanismen entwickelt, die hier aufgegriffen und weiter benutzt werden. Wie schon im vorherigen Kapitel beschrieben, trennt die MDA die Modellierung von der Implementierung, aber gleichzeitig integriert sie das Modell und dessen Entwicklung auf einer Zielplattform [OMG 01].

6.1. Transformationen und Abbildungen in MDA

Es werden unterschiedliche Techniken benutzt, die die Modellierung, das Design, die Implementierung und die Integration von Systemen vereinfachen. Die Grundidee in MDA ist, dass jedes Modell auf einem Metamodell basiert. Jedes Metamodell definiert genau eine Modellierungssprache. Letztendlich basieren alle Metamodelle auf einem (übergeordneten) Metametamodell. In MDA bietet die Meta-Object Facility (MOF) Standardabbildungen, beispielsweise auf XML (XML Meta-Data Interchange (XMI), auf Java (Java Metadata Interface

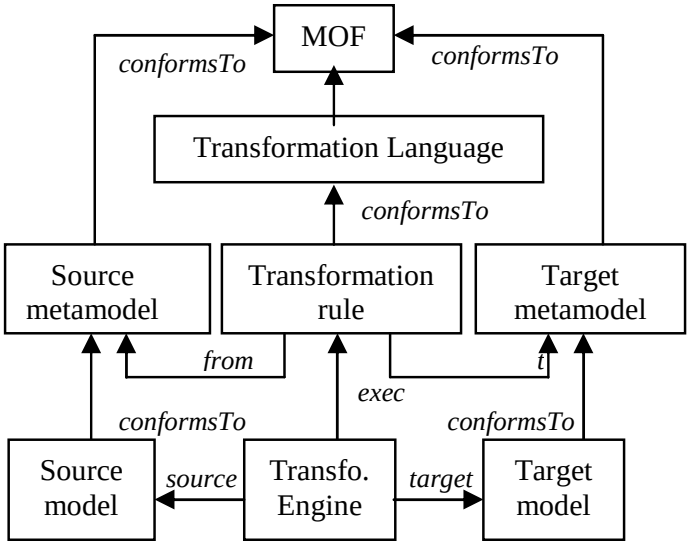


Abbildung 6.2.: Grundlegendes Transformationsmodell der MDA [HJL 05]

(JMI)) oder auf WebServices (Web Services Description Language (WSDL)), an [SH 10]. Abbildung 6.2 stellt das zugrundeliegende Transformationsmodell der MDA dar. Es ist ein allgemeines Modell, durch das Transformationen von mehreren Ausgangsmodellen (Source models) und/oder zu mehreren Zielmodellen möglich sind. MOF ist ein etabliertes Metametamodell, was die Basis für viele andere Metamodelle fungiert. Auf der einer Seite befindet sich ein PIM, das die Funktionalitäten, die Struktur und das Verhalten eines Systems beschreibt und auf der anderen Seite, näher zur Implementierung, findet man das PSM, das die erste Bindung des PIMs an eine bestimmte Plattform realisiert [HJL 05]. Das PSM ist zwar nicht die Implementierung, aber beinhaltet genug Informationen, um Programmcode, Schnittstellendefinitionen, Konfigurationsdateien usw. zu generieren. Die Abbildung des PIMs auf das PSM bestimmt die äquivalenten Elemente zwischen den zwei Metamodellen [BHLJ 04]. Für jedes PIM gibt es oder kann es mehrere PSMs geben; die Auswahl der Zielplattform bestimmt dasjenige PSM, auf das das PIM abgebildet wird (siehe Abbildung 6.3).

Zwei oder mehrere Elemente unterschiedlicher Metamodelle sind äquivalent, falls sie kompatibel sind und sich gegenseitig nicht widersprechen [Lope 05]. Die grundlegende Regel ist demnach, dass ein Model in ein anderes nicht transformiert werden kann, wenn das Metamodell des ersten auf das Metamodell des letzteren nicht abgebildet werden kann [LHdSB 06]. Das Metamodell wird abgebildet, indem die äquivalenten oder gleichartigen Elemente identifiziert werden, ebenso wie der semantische Abstand (*semantic distance*, wie in [BBI$^+$ 04, Gora 05] beschrieben), über den minimiert wird. Das Suchen solcher äquivalenten Elemente wird als Schemaabbildung (*schema matching*) bezeichnet [LHA 06]. Lopes et al. untersuchen die Problematik der Schemaabbildung gründlich in [LHA 06] und geben

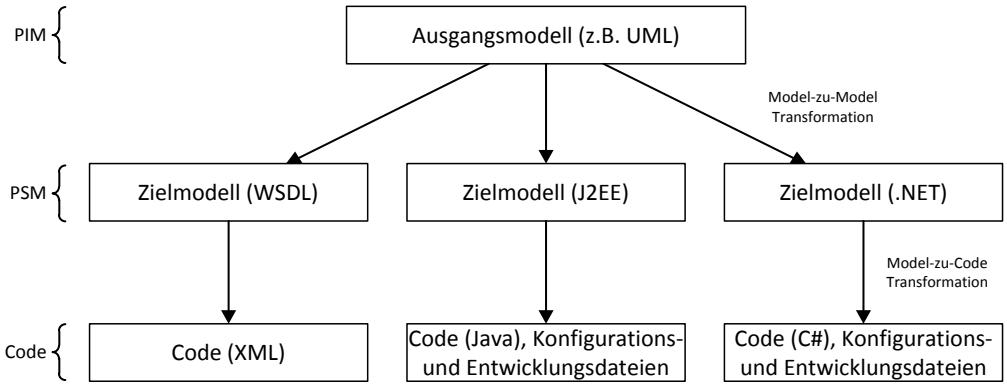


Abbildung 6.3.: Mögliche MDA-Transformationen

klare Methodiken und Implementierungen diesbezüglich an. An dieser Stelle muss man den Unterschied zwischen Abbildung (*mapping*) und Transformation (*transformation*) explizit erläutern. Die Abbildung steht für Übereinstimmung zwischen den Elementen zweier Metamodelle. Auf der anderen Seite wird anhand einer Transformationsdefinition von einem Ausgangselement (*source*) zu einem Zielelement (*target*) transformiert. Die zwei Begriffe sind eng miteinander verbunden, so wird ein Transformationsmodell von einem Abbildungsschema generiert [LHBJ 06].

Eine abstrakte Repräsentation der Abbildung zwischen zwei Metamodellen ist in Abbildung 6.4 grafisch dargestellt. Nach Lopez [Lope 05] werden auf der linken Seite die Elemente des Ausgangsmetamodells (A1/A2) dargestellt, die über ein bestimmtes Abbildungselement (A2B oder Aa2Bb) in ein Element des Zielmetamodells (B1/B2) umgewandelt werden. Es wird hier zwischen mehreren Abbildungskategorien (eins-zu-eins, eins-zu-mehrere und mehrere-zu-eins) unterschieden. Diese Klassifikation basiert auf dem Konzept der gleichartigen Struktur und Semantik zwischen den Elementen des Metamodells. Es wird von einer gleichartigen Struktur und Semantik gesprochen, falls durch das/die Zielelement(e) dieselbe Information wie durch das / die Ausgangselement(e) repräsentiert werden kann. Im Fall der eins-zu-eins Abbildung weist das Ausgangselement eine gleichartige Struktur und Semantik wie das Zielelement auf. Wenn sich ein Element des Ausgangsmodells auf eine Menge gleichartiger Elemente des Zielmodells abbilden lässt, dann liegt eine eins-zu-mehrere Abbildung zugrunde. Bei der mehrere-zu-eins Abbildung ist die Gleichartigkeit einer Menge an Elementen aus dem Ausgangsmetamodell auf ein einziges Element des Zielmetamodells zurückzuführen [LHBJ 05]. Diese Abbildungsschemata sind in der Atlas Transformation Language (ATL) – eine Transformationssprache benutzt im Kontext der MDA – implementiert.

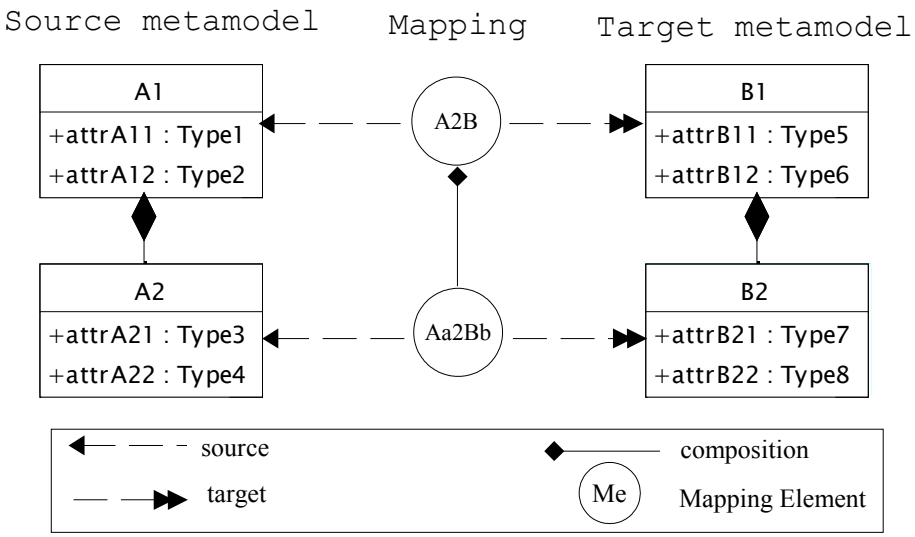


Abbildung 6.4.: Die Repräsentation der Abbildungen (*mappings*) [Lope 05]

6.2. Formalisierung der Abbildungsspezifikation

Die Abbildungsspezifikation ist weitgehend [LHBJ 05] entnommen.
Gegeben seien $M_1(s)/M_a$, $M_2(s)/M_b$, und $C_{M_a \to M_b}/M_c$, mit:

$M_1(s)/M_a$: ein Model (Ausgangsmodell) des zu beschreibenden Systems s generiert mit dem Metamodell M_a

$M_2(s)/M_b$: ein Model (Zielmodell) des zu beschreibenden Systems s generiert mit dem Metamodell M_b

$C_{M_a \to M_b}/M_c$: die Abbildung zwischen M_a und M_b generiert durch Anwendung des Metamodells M_c

Damit kann eine Transformation des Ausgangsmodells $M_1(s)$ auf das Zielmodell $M_2(s)$ formal definiert werden als:

$Transf(M_1(s)/M_a, C_{M_a \to M_b}/M_c) \to M_2(s)/M_b$

Im Allgemeinen basieren alle Metamodelle M_a, M_b und M_c auf demselben Metametamodell (beispielsweise MOF, vgl. Abbildung 6.2). Diese Tatsache vereinfacht die Abbildungsspezifikation erheblich und ermöglicht eine Transformation (z.B. in ATL). Hiermit wird die Abbildung (*mapping*) zwischen den zwei Metamodellen M_a und M_b als

$C_{M_a \to M_b} \supseteq \{M_a \cap M_b\}$

definiert. Der binäre Operator $\cap$ liefert die Elemente von M_a und M_b, die gleichartige Struktur und Semantik haben. Die Metamodelle M_a, M_b und $C_{M_a \to M_b}$ können dann als Mengen definiert werden:

$M_a = \{a_1, a_2, a_3, ..., a_m\}$, $M_b = \{b_1, b_2, b_3, ..., b_n\}$ und $C_{M_a \to M_b} = \{c_1, c_2, c_3, ..., c_p\}$,

mit $c_i = \{a_k, b_j\}$ und $i = \{i \in \mathbb{N} \mid 1 \leq i \leq p\}$, $k = \{k \in \mathbb{N} \mid 1 \leq k \leq m\}$, $j = \{j \in \mathbb{N} \mid 1 \leq j \leq n\}$

Auf diese Art und Weise können alle Modelle (Modelle, Metamodelle und Metametamodelle) als Mengen repräsentiert werden. Dennoch sind diese sehr komplex und heterogen, da deren Elemente Klassen, Attribute, Beziehungen, Aufzählungen und Datentypen sind. Aus diesem Grund werden die Elemente des Metamodells in zwei Kategorien aufgeteilt: Basiselemente (*basic elements*) und Beziehungen (*relationships*). Basiselemente sind Klassen, Attribute, Aufzählungen und Datentypen. Beziehungen sind Verbindungen zwischen den Klassen. Zwei Regeln sind von einer gültigen Abbildung $C_{M_a \to M_b}$ einzuhalten:

Erhalt der Basiselemente: Jedes Element aus M_a muss eine der folgenden Regeln erfüllen:

- Es entspricht einem gleichen (=) Basiselement in M_b
- Es entspricht einer Menge von Basiselementen in M_b, die gleichartig sind ($\cong$), also die miteinander verwandt sind, wobei nicht klar definiert sein muss, in welcher Weise.
- Es ist in einer Teilmenge von Basiselemente aus M_a, die untereinander gleichartig sind, und einem Element aus M_b entsprechen.

Erhalt der Beziehungen: Jede Beziehung aus M_a muss eine der folgenden Regeln erfüllen:

- Sie hat eine korrespondierende Beziehung in M_b.
- Sie korrespondiert mit einer gleichartigen Menge an Beziehungen in M_b.
- Sie kann implizit in M_b (z. B. durch Aggregation von Attribute oder Zusammenführen von Klassen) enthalten sein, d. h. wenn in M_b ein Basiselement oder eine Beziehung auf zwei oder mehrere Basiselemente oder Beziehungen in M_a zurückzuführen sind.

Wenn diese beiden Anforderungen von der Abbildungen nicht erfüllt sind, dann ist das Abbildungsschema unvollständig und die Transformationsdefinition ebenso. Folglich enthält ein anhand einer solchen Transformation generiertes Zielmodell nicht alle Informationen des Ausgangsmodells.

Da in dieser Arbeit die ioFMA auf eine Web Services Plattform (bzw. auf WSDL) transformiert werden soll, ist der erste Schritt nach dem oben beschriebenen Prinzip, eine geeignete Transformation des UML-Metamodells in das WSDL-Metamodell zu definieren.

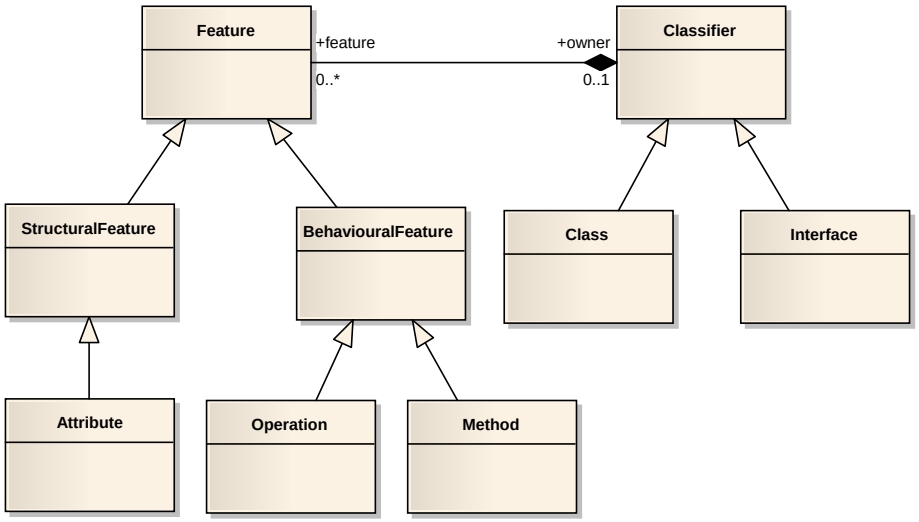


Abbildung 6.5.: Das UML-Metamodell (Fragment)

6.3. Transformation von UML auf WSDL

Wie schon im Abschnitt 5.3 in Rahmen der Motivation für die Plattformauswahl erwähnt, ist Web Services Description Language (WSDL) eine Spezifikationssprache für Web Services. Diese besitzt eine abstrakte Definition, basierend auf einer XML-Grammatik, die die Syntax und Semantik, die für den Dienstaufruf benötigt wird, beschreibt. Bisher wurden die zwei Versionen WSDL 1.1 [W3C 01] und WSDL 2.0 [W3C 07] veröffentlicht, allerdings ist die letztere noch nicht so weit verbreitet in den Web Services-Implementierungen wie die erste; deswegen ist hier unter WSDL immer WSDL 1.1 gemeint. Web Services sind Applikationen, die anhand von XML-Artefakten Schnittstellen, Datenbindungen (*bindings*) und Aufrufe (*invocations*) definieren, beschreiben und auffinden (*discover*).

Wie in Abbildung 6.2 dargestellt, werden bei einer Transformation eines PIM (UML) in ein PSM (Web Services) zwei Metamodelle (Ausgangspunkt und Ziel) benötigt. Das UML-Metamodell ist der Ausgangspunkt und das WSDL-Metamodell ist das Ziel.

Abbildung 6.5 zeigt ein Fragment des UML-Metamodells mit den folgenden (für diese Arbeit wichtigen) Klassen:

- `Class`: Eine Instanz von `Class` beschreibt die Menge der Objekte mit denselben Attributen, Operationen und Beziehungen.
- `Interface`: Ein `Interface` spezifiziert das externe Verhalten einer Klasse und enthält in abstrakter Form Signaturen und Beschreibungen von Operationen.
- `Attribute`: Repräsentiert eine Eigenschaft oder einen Zustand

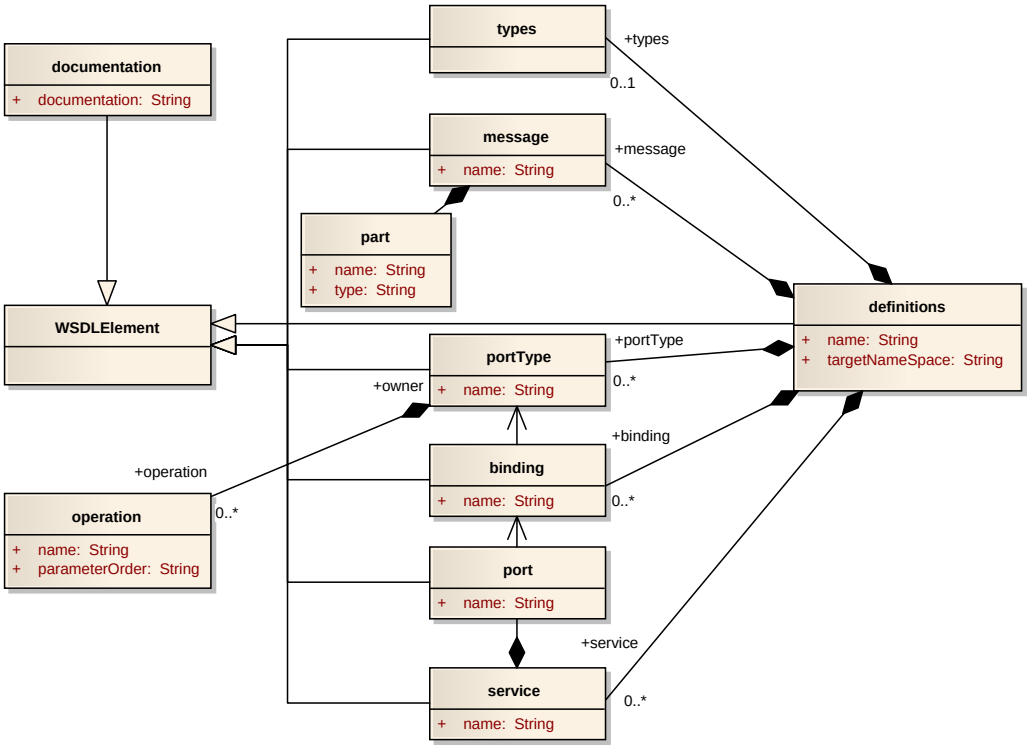


Abbildung 6.6.: Das WSDL-Metamodell (vereinfacht)

- `Operation`: Ist eine Dienstleistung, die von einem Objekt aufgerufen werden kann. Sie wird durch ihre Signatur (Operationsname, Parameter, Rückgabewert) beschrieben.

- `Method`: Ist die Implementierung einer Operation. Sie besteht aus einer Folge von Anweisungen.

Im Folgenden wird als Ziel der Transformation das WSDL-Metamodell (siehe Abbildung 6.6) betrachtet. Die Syntax bzw. Notationen im Abbildung 6.6 ist von Lopez in [Lope 05] beschrieben und in dieser Arbeit eingeführt. Das Metamodell der WSDL hat als Wurzel das `definitions`-Element, das aus den folgenden Komponenten besteht [W3C 01]:

- `types`: Definiert Datentypen, benutzt zur Beschreibung der ausgetauschten Nachrichten

- `message`: Abstrakte Definition der übertragenden Daten

- `portType`: Ein Satz abstrakter Operationen; jede davon verweist auf eingehende und ausgehende Nachrichten (*input* und *output messages*)

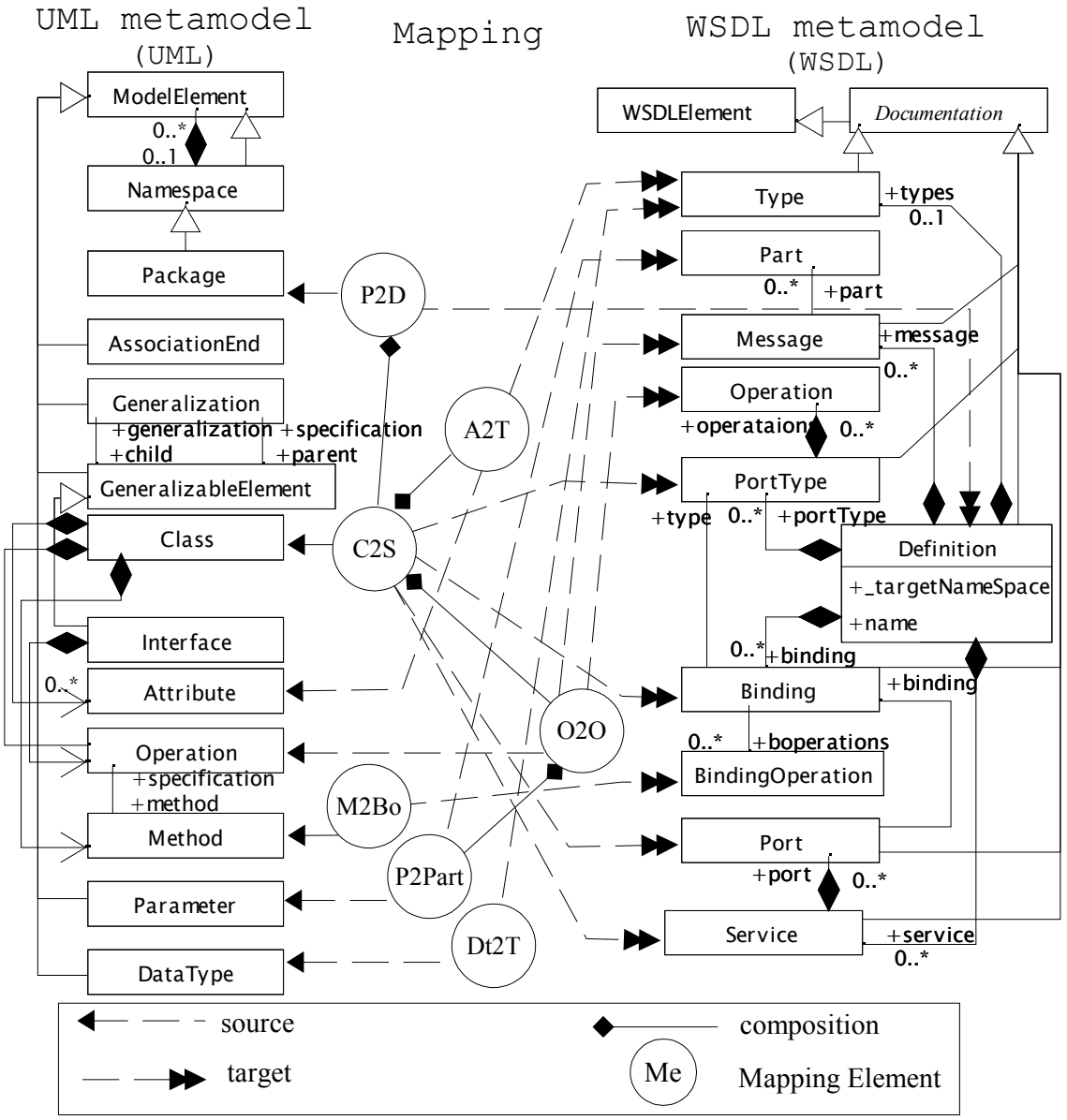


Abbildung 6.7.: Die Abbildungsspezifikation von UML auf WSDL [Lope 05]

- `binding`: Spezifiziert das konkrete Protokoll und die Datentypspezifikationen für die Operationen (`operation`) und Nachrichten (`message`) definiert von einem bestimmten `portType`

- `port`: Spezifiziert eine Adresse zur Bindung, und definiert somit einen einzelnen Kommunikationsendpunkt

- `Service`: Aggregiert unterschiedliche verwandte Ports.

Abbildung 6.7 stellt die Abbildung (`Mapping`) des UML-Metamodells auf dem WSDL-Metamodell dar. Die benutzte Notation wurde in Abbildung 6.4 eingeführt und ist [Lope 05] entnommen.

- Das Element `P2D` stellt die Abbildung zwischen dem Element `Package` des UML-Metamodells und dem Element `definitions` des WSDL-Metamodells.

- Zwischen dem Element `Class` und den Elementen `Service`, `port`, `binding` und `portType` befindet sich das Element `C2S`.

- Das Abbildungselement `O2O` realisiert die Verbindung zwischen dem Element `Operation` und den Elementen `operation` und `message`.

- `P2Part` bildet das Element `Parameter` aus UML auf das Element `part` aus WSDL ab.

- Die Abbildung `Dt2T` wird zwischen dem Element `DataType` aus UML und dem Element `types` realisiert.

Wenn man die Abbildung `P2D` näher betrachtet (vgl. Abbildung 6.8), wird deutlich, dass das Element `P2D` ein komplexes zusammengesetztes Element ist. Seine Teile `n2n` und `n2t` deuten auf die Existenz von zwei Abbildungen für das gleiche Element hin, dies kann auf eine „Inkonsistenz" des Ausgangs- und Zielelements zurückgeführt werden. Das Ausgangselement `Package` des UML-Metamodells beinhaltet nur das Attribut `name`, das auf zwei Attribute (`name` und `targetNameSpace`) des Zielelements `definition` abzubilden ist. Deswegen wird die komplexe Abbildung `P2D` mit den zwei Abbildungen `n2n` (`name-zu-name`) und `n2t` (`name-zu-targetNameSpace`) definiert. `n2n` wird sehr einfach definiert, indem das `name`-Attribut (UML) direkt in `name` überführt wird. Die Abbildung `n2t` (von `name` zum `targetNameSpace`) ist eine komplexere Abbildung. Es wird hier die in UML integrierte Sprache Object Constraint Language (OCL) die zur textuellen Spezifikation benutzt wird. Somit wird anhand des OCL-Ausdruckes `'urn://'+ @.name+ '.wsdl'` die Transformation realisiert. Der Platzhalter `@` wird zum Referenzieren des Ausgangselements benutzt.

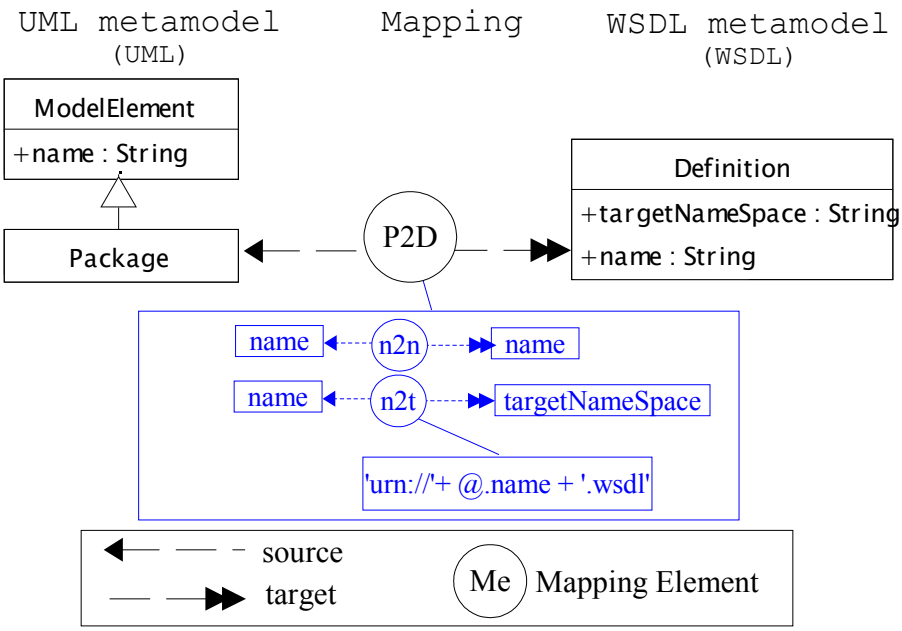


Abbildung 6.8.: Abbildung `P2D` in Detail [Lope 05]

6.4. Transformation der ioFMA

Konform zu der obengenannten Metamodell-zu-Metamodell Transformation wird nun das ioFMA-PIM (auf Basis der UML) in der PSM (auf Basis von WSDL) übergeleitet. Da die Abbildbarkeit des UML-Metamodells auf das WSDL-Metamodell definiert wurde, kann man nun die benötigten Transformationsregeln aufstellen.

6.4.1. Pakete

Die Bezeichner der Pakete der ioFMA werden anhand des OCL-Ausduckes `'urn://'+ @.name+ '.wsdl'` in Namespaces, wie in Tabelle 6.1 dargestellt, transformiert.

6.4.2. Einfache Klassen

Die UML-Klassen der ioFMA werden als Teil des WSDL-Element `types` zu Typen eines XML-Schema. Beispielsweise wird die Klasse `ioFMARootEntity` zu einem Element des XML-Schemas IOFMATopLevel.xsd vom Typ `complexType`. Die Attribute dieser Klasse `description`, `entityName` und `objectid` werden zu einer Sequenz von XML-Elementen innerhalb des komplexen Typs. Diese Klasse kann dann zusammengefasst folgendermaßen transformiert werden:

ioFMA-Package	WSDL-Namespace
IOFMATopLevel	urn://IOFMATopLevel.wsdl
IOFManagement	urn://IOFManagement.wsdl
Service	urn://Service.wsdl
Resource	urn://Resource.wsdl
Fault	urn://Fault.wsdl
IOFMARole	urn://IOFMARole.wsdl
IOFMASpecification	urn://IOFMASpecification.wsdl

Tabelle 6.1.: Transformation der ioFMA-Packages in WSDL-Namespaces

```
...
<xs:complexType name="ioFMARootEntity">
   <xs:sequence>
      <xs:element name="description" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="entityname" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="objectid" type="xs:string" minOccurs="1" maxOccurs="1"/>
   </xs:sequence>
</xs:complexType>
```

Die UML-Operationen werden zu `operation`-WSDL-Elemente transformiert. Für deren unterschiedlichen Typen gibt es unterschiedliche Transformationsregeln:

6.4.3. Getter-Operationen

Eine getter-Operation im UML-Modell wird zu einer WSDL-Operation vom Typ `Request-Response` mit dem Namen der ursprünglichen Operation. Diese hat zwei Teile: `input` für die eingehende Nachricht und `output` für die entsprechende Antwort. Jeder der Bestandteile beinhaltet ein Attribut `name` und ein Attribut `message` mit dem entsprechenden Wert: Name der `get`-Operation + 'Request'/'Response'. Die `message`-Elemente sind vorher in der WSDL-Datei definiert. Beispielsweise wird die in ioFMA definierte Operation *getEntityName()* wie folgt transformiert.

```
...
<wsdl:message name="getEntityNameRequest">
    <wsdl:part name="term" type="xs:string"/>
  </wsdl:message>
  <wsdl:message name="getEntityNameResponse">
    <wsdl:part name="value" type="xs:string"/>
</wsdl:message>
...
    <wsdl:operation name="getEntityName">
      <wsdl:input name="Request1" message="tns:getEntityNameRequest"/>
      <wsdl:output name="Response1" message="tns:getEntityNameResponse"/>
    </wsdl:operation>
...
```

6.4.4. Assoziationen

Die Assoziationen zwischen den unterschiedlichen UML-Klassen und deren Operationen werden zu `Solicit-Response` WSDL-Operationen transformiert. Der Unterschied ist in diesem Fall, dass erst eine Nachricht gesendet und danach auf eine Antwort gewartet wird. Folgendes ist ein Beispiel für die Transformation der Assoziation `affects` zwischen den Klassen `Service` und `ServiceFailure` zur WSDL-Operation mit zwei Teilen: `output` und `input`. Die `message` im `output`-Teil stammt ursprünglich von der Klasse `Service`, die im `input` Teil aus der Klasse `ServiceFailure`.

```
...
<wsdl:message name="getFailureRequest">
   <wsdl:part name="term" type="Service.serviceID"/>
</wsdl:message>
  ...
<wsdl:message name="setFailureRequest">
   <wsdl:part name="term" type="ServiceFailure"/>
</wsdl:message>
...
  <wsdl:operation name="afects">
    <wsdl:output name="Solicit1" message="getFailureRequest"/>
    <wsdl:input name="Request1" message="setFailureRequest"/>
  </wsdl:operation>
```

6.4.5. Setter-Operationen

Eine setter-Operation in UML wird zu einer WSDL-Operation vom Typ `One-Way`, mit gleichem Namen. Sie beinhaltet nur ein `input`-Element mit den Attributen `name` (z. B. OneWay1) und `message`. Ein Beispiel für die Transformation der Operation *setFailure* der Klasse `ServiceFailure` ist hier dargestellt:

```
...
<wsdl:message name="setFailureRequest">
   <wsdl:part name="term" type="ServiceFailure"/>
</wsdl:message>
...
<wsdl:operation name="setFailure">
      <wsdl:input name="OneWay1" message="setFailureRequest"/>
</wsdl:operation>
```

6.4.6. Benachrichtigungen

Analog werden auch Benachrichtigungsoperationen transformiert. Diese sind ebenfalls Einweg-Operationen, aber nur mit einem Teil `output`. Eine `message`-Definition (Name der Operation + 'Response') geht voraus. Nachfolgend ist die Transformation der Operation *inform* des Packages `IOFMARole` in einer WSDL-Operation vom Typ `Notification` dargestellt:

```
...
<wsdl:message name="informResponse">
   <wsdl:part name="value" type="FaultInformation"/>
</wsdl:message>
...
   <wsdl:operation name="inform">
          <wsdl:output name="Notification2" message="informResponse"/>
   </wsdl:operation>
```

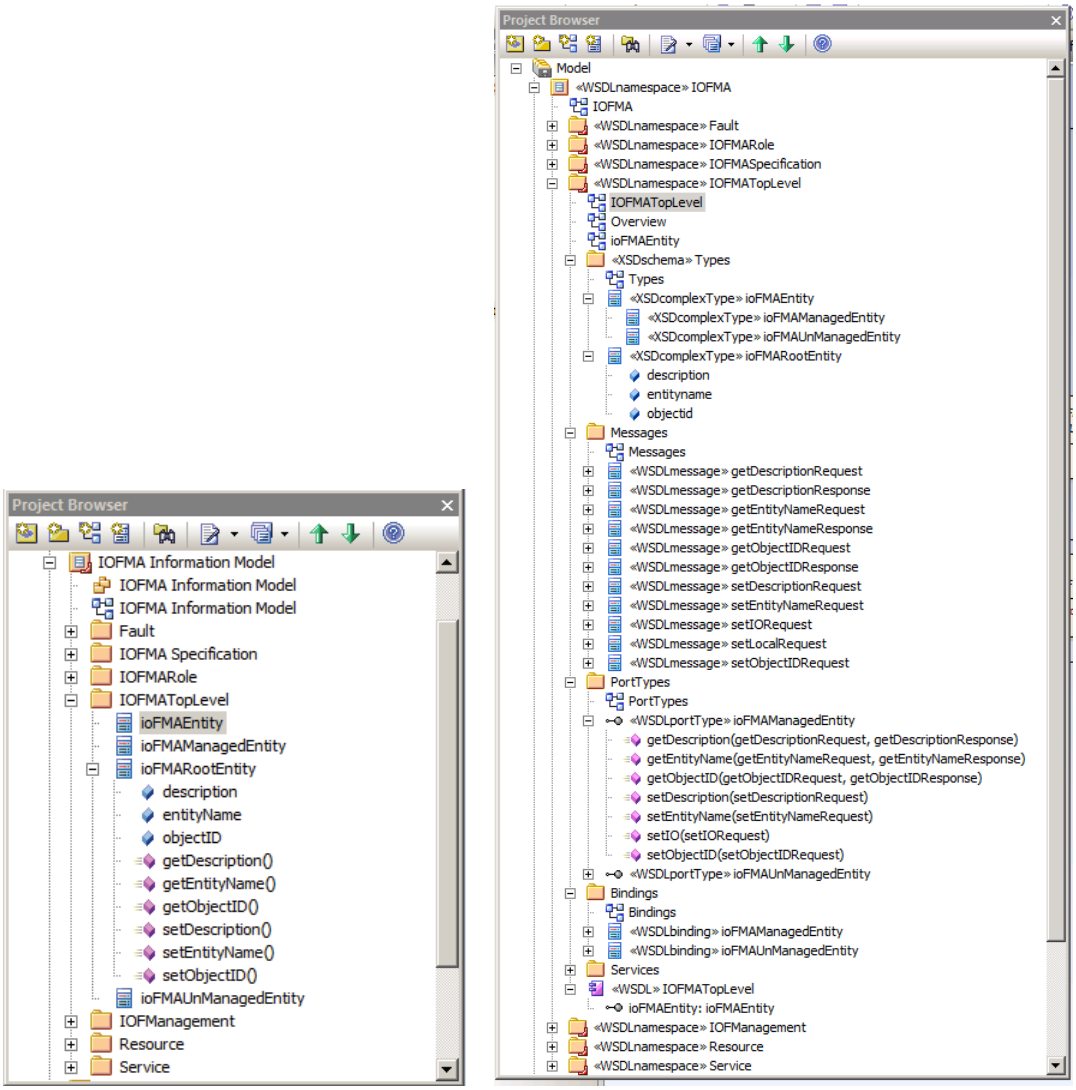
6.4.7. Klassen mit Operationen

Eine UML-Klasse, die Operationen bzw. Klassenmethoden besitzt, wird zu einem `portType`-WSDL-Element mit demselben Namen wie die ursprüngliche Klasse transformiert. Das `portType`-Element beinhaltet `operation`-Elemente, die oben beschrieben wurden. Ein Beispiel für eine solche Transformation der Klasse `ServiceFailure` in das `portType`-Element `ServiceFailure` ist hier dargestellt:

```
...
  <wsdl:portType name="ServiceFailure">
    <wsdl:operation name="inducesFault">
      <wsdl:output name="Notification1" message="setFailureRequest"/>
    </wsdl:operation>
    <wsdl:operation name="setFailure">
      <wsdl:input name="OneWay1" message="setFailureRequest"/>
    </wsdl:operation>
    <wsdl:operation name="afects">
      <wsdl:output name="Solicit1" message="getFailureRequest"/>
      <wsdl:input name="Request1" message="setFailureRequest"/>
    </wsdl:operation>
  </wsdl:portType>
```

Ein besserer Überblick über die Transformation wird in der Gegenüberstellung mit Abbildung 6.9 verschafft. In Abbildung 6.9(a) wird die Projektansicht des `IOFMATopLevel`-Packages der ioFMA-PIM dargestellt. Gegenüber befindet sich die Abbildung 6.9(b) die der `IOFMATopLevel`-Namespace des PSM in Projektansicht (mit allen WSDL-Elementen).

In dieser Art und Weise wird das gesamte ioFMA-PIM entsprechend der obengenannten Transformationsregeln in dass ioFMA-PSM umgewandelt. Der Vollständigkeit halber sind am Ende dieser Arbeit beispielhaft die WSDL-Modelle für drei der Namensräume hinzugefügt: `IOFMATopLevel` (Anhang I), `Fault` (Anhang II) und `Service` (Anhang III). An dieser Stelle sind nur Bespiele für den Allgemeinfall angegeben; die hierarchischen bzw. heterarchischen PSM-Modelle können aber leicht durch Erweiterung und durch Anwenden der Transformationsregeln auf das entsprechende PIM-Modell realisiert werden.



(a) IOFMATopLevel-Package (PIM) (b) IOFMATopLevel-Namespace (PSM)

Abbildung 6.9.: Transformation des `IOFMATopLevel`-Packages nach WSDL

6.5. Zusammenfassung

Dieses Kapitel befasste sich mit der Definition eines geeigneten PDM, das die Transformation vom ioFMA-PIM in das ioFMA-PSM ermöglicht. Dies beinhaltet die Abbildung und Transformation von Metamodellen und darauf beruhend die Erstellung von Transformationsregeln für die ioFMA. Das in dieser Arbeit benutzte Transformationsbeispiel basiert auf

der Web-Services Plattform, wobei das in UML entworfene ioFMA-PIM in ein WSDL-Modell transformiert wurde. Diese Vorgehensweise ist insofern generisch, als bei einer anderen Plattform (z.B. .NET oder J2EE) analog vorgegangen werden kann. Ende des Kapitel 5 wurde in Tabelle 5.11 (siehe Seite 181) die Erfüllung der im Kapitel 3 gestellten Anforderungen nach der Realisierung der ioFMA-PIMs zusammengefasst. Von den noch nicht erfüllten Anforderungen sind nach Realisierung des PSM-ioFMA die IM-01 (gemeinsames Datenformat) vollständig und die IM-02 (Konvertierungsmethoden) teilweise erfüllt. Die KM-02 (Interdomänen-Kommunikation) wird durch die Definition eines gemeinsames Datenformats unterstützt.

Das Ergebnis dieses Kapitels ist ein PSM, das aber immer noch ein Modell ist. Um zu einer Realisierung zu gelangen, sind noch einige wichtige Punkte zu beachten:

- Die ioFMA muss in bereits bestehende Systemen integrierbar sein.
- Die zugrunde liegenden Techniken der ioFMA sollten an aktuelle, weit verbreitete Realisierungstechniken angepasst werden.
- Die Anbindung und Anpassung der ioFMA an eine bestehende Infrastruktur (Netz) muss realisiert werden.
- Ein Client/Server „Integrationssystem" muss bereitgestellt werden.
- Nicht zuletzt wird ein Oberflächenbaustein benötigt, um die Nutzerfreundlichkeit des Fehlermanagementsystems zu steigern.

Um die Tragfähigkeit der Modelle und der Vorgehensweise zu demonstrieren, wird in Kapitel 7 basierend auf der perfSONAR-Architektur ein Teil der ioFMA im interorganisationalen Bereich der E2E-Links des LHCOPN umgesetzt. Nach dieser prototypischen Einführung werden die bis zu diesem Zeitpunkt noch nicht erfüllten Anforderungen noch einmal bewertet und aktualisiert.

7 Deployment und operativer Einsatz

Inhalt des Kapitels

In Kapitel 3 wurden die Anforderungen an interorganisationales Fehlermanagement und an eine geeignete Managementarchitektur durch die Darstellung von Szenarien und darauf aufbauenden Anwendungsfällen aufgestellt. Diese dienen dann als Grundlage für den Entwurf der ioFMA (Kapitel 5). Das Organisationsmodell beschäftigt sich mit der Abgrenzung von Managementdomänen (`IOFMADomain`) und den darin erhaltenen Rollen (`IOFMARole`). Das Funktionsmodell umfasst die Beschreibung der Funktionsbereiche und der darunterliegenden Managementfunktionen. Das Informationsmodell fasst die Informationen, die als Basis den beiden anderen Modellen zur Verfügung gestellt werden, zusammen. Das Informationsmodell der ioFMA stellt unter anderem die Managementobjekte (*managed objects*) bereit. Die wichtigsten Klassen in diesem Zusammenhang sind `ioService`, `ioResource` und `ioFault`; in allen Fällen handelt es sich um interorganisationale Objekte. Das Kommunikationsmodell ergänzt die ioFMA mit Zusatzinformationen bezüglich Kommunikationsmechanismen zum Austausch von Managementinformationen.

Mit den in Kapitel 5 entworfenen Teilmodellen wurde ein (plattformunabhängiges) Rahmenwerk ioFMA zum interorganisationalen Einsatz für dass Fehlermanagement realisiert. Dem MDA-Entwicklungsansatz folgend wurde in Kapitel 6 anhand von Transformationsvorschriften die plattformunabhängige ioFMA in ein plattformspezifisches Modell überführt. Dieses wurde für eine Web-Services-Plattform demonstriert, indem ein WSDL-Modell realisiert wurde. Diese Transformationen wurden so definiert, dass sie nicht nur für Web Services anwendbar sind, sondern für jede Art von Plattform, auch für spezielle Konzepte wie z. B. perfSONAR. Diese Portabilität der Teilmodelle auf unterschiedliche Plattformen kann Dank des MDA-Ansatzes realisiert werden. Dieser sieht die Wiederverwendbarkeit der Teilmodelle vor. Das Deployment für das auf perfSONAR basierende LHCOPN-Management in diesem Kapitel repräsentiert den Implementierungsschritt für ein heterarchisches Szenario im MDA-Vorgehen. Die Implementierung eines hierarchischen Szenarios ist in dieser Arbeit nicht vorgesehen. Die hier entwickelte Methodik bzw. PIM/PSM-Modell ist durch ihre inhärente Flexibilität auch auf weitere Fälle leicht anwendbar.

7.1. Deployment der ioFMA für das LHCOPN

Um die ioFMA in bestehenden Systemen einzusetzen, ist es notwendig, eine bestimmte Vorgehensweise zu definieren. Im folgenden wird diese Vorgehensweise für das Deployment der ioFMA im bereits bestehenden LHCOPN-Management [UlSc 06] und dessen zugrunde liegenden perfSONAR-System [per 08] definiert. Die Infrastruktur des LHCOPN [BMM 05] besteht seit längerer Zeit und wurde im Laufe des Géant2 Projektes [GÉ 07] etabliert. Dieses interorganisationale, weltweit verteilte Netz basiert grundlegend auf dem in Géant2 entwickelten perfSONAR-System. Für dieses gegebene Netz bzw. System, wird im folgenden der Einsatz der ioFMA demonstriert. Dieses Deployment hat in der Praxis prototypisch in Form der „LHCOPN-Wetterkarte" stattgefunden [MSFY 11] und wird hier schrittweise beschrieben. Die Realisierung dieser Lösung in einer bereits etablierten Infrastruktur ist ein Beispiel des ioFMA-Deployments in der Praxis.

Wie schon im Abschnitt 3.1.3 definiert, ist das LHCOPN ein sternförmiges, weltweites Netz mit CERN alsTier-0 in der Mitte. Jedes der elf Tier-1-Zentren ist mit einem E2E-Link an dieses angebunden [BMM 05, LHC 09]. Die Überwachung dieser E2E-Links wird mit Hilfe des E2EMon-Werkzeugs [HaYa 08] (beschrieben in Abschnitt 4.1.2) realisiert.

Die Performance focused Service Oriented Network monitoring ARchitecture (perfSONAR) als eine Komponentenarchitektur für ein föderiertes Multi-Domain Monitoring wurde im europäischen Netzverbund Géant [Geant11] entwickelt. Diese wird für die QoS- und SLA-Überwachung im Géant-Netz benutzt. perfSONAR hat auch das Ziel der Unterstützung domänenübergreifender Fehlersuche. Sowohl die dreischichtige perfSONAR-Architektur (Measurement Point Layer, Service Layer und User Interface Layer) als auch die Metriken und

Messverfahren zum Performance Monitoring [per 08], die perfSONAR benutzt, wurden in Abschnitt 4.1.1 beschrieben.

Wenn man aus einem integrierten Management-Ansatz heraus den Einsatz der ioFMA im hauptsächlich auf perfSONAR und E2EMon basierenden komplexen System des LHCOPN betrachtet, ist dafür eine Managementplattform nach [HAN 99] am besten geeignet (vgl. Abschnitt 2.1.1.1). Somit werden, ausgehend von der ioFMA, ein Informationsbaustein, ein Kommunikationsbaustein, Basis- und Managementanwendungen sowie eine Benutzeroberfläche benötigt. Diese Komponenten und der darauf aufbauende Entwurf werden in diesem Abschnitt ausführlich beschrieben.

7.1.1. Entwurf des Informationsbausteines

Das in Abschnitt 5.2.3 definierte Informationsmodell fungiert als Basis für den Informationsbaustein. Zusätzlich zum plattformunabhängigen Modell wurde auch eine mögliche Web Services Implementierung besprochen. Durch das nach WSDL transformierte Informationsmodell wurden auch Grundlagen für die Kommunikation mit unterschiedlichen Datenformaten, die Web-Services-orientiert sind, gelegt.

Zuerst wird die Infrastruktur des LHCOPN, der Aufbau dessen Zentren, die unterschiedlichen geschichteten Topologien, die darin realisierten Netzmessungen und die aktuelle technische Implementierung, beschrieben. Dies dient zur Abbildung der in ioFMA beschriebenen Managementobjekte auf die realen Infrastrukturelemente. Die Beschreibung der Komponenten des Infrastruktur wird in Hinblick auf die Visualisierung immer realisiert, weil alle hier beschriebenen Bestandteile später visuell dargestellt werden müssen. Nach der Beschreibung der Infrastruktur und der damit bezogenen Informationen, wird die obengenannte Abbildung realisiert.

In Rahmen von perfSONAR wird für die Abholung und Bearbeitung von Kennzahlen das Network Measurement Working Group (NMWG)-Schema des Open Grid Forum (OGF) [NMWG] eingesetzt. NMWG entstand aus der Notwendigkeit, Netzleistung (*network performance*) zu messen und zu vergleichen und basiert auf früheren Messtechniken, die für Grid-Applikationen genutzt wurden. NMWG kommt dem IPPM Arbeitskreis nahe. Beide konzentrieren sich auf Best Practices im Bereich der Netzmessungen und auf die Etablierung valider Messprotokolle. Das NMWG-Schema, ist ein globales System zur Kategorisierung aller zum damaligen Zeitpunkt im Einsatz befindlichen Messungen unter Berücksichtigung der Anforderungen von Grid-Applikationen.

Das NMWG-Schema wird für die Beschreibung der Topologie [NMTop] und von Messungen [NMMetr] benutzt. Es ist bereits für alle IPPM-Kennzahlen [RFC 3393], aber auch zur Statusermittlung von E2E-Links (siehe Anhang IV) geeignet. Es lässt sich sehr leicht an WSDL anbinden, weshalb die Integration in das ioFMA-PSM selbstverständlich ist.

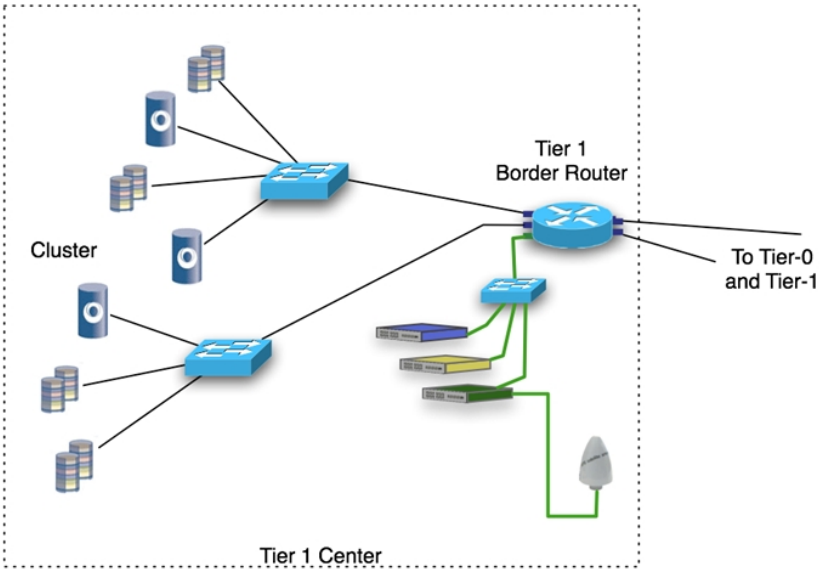


Abbildung 7.1.: Konfiguration eines Tier-1 -Routers [MSHT 10]

Natürlich werden für das Deployment in einer bestimmten Umgebung nicht immer alle Teile der Architektur benötigt. Damit ist gemeint, dass manche Klassen detaillierter implementiert oder eingeführt werden als andere. Um eine Übersicht über die benötigten Informationen zu geben, wird sowohl der Aufbau des LHCOPN und dessen Management als auch der darunterliegenden perfSONAR-Architektur unter der Lupe genommen.

Die in perfSONAR benutzten Metriken wurden schon ausführlich im Abschnitt 4.1.1.1 dargestellt. Hier wird nur deren Zusammenhang mit dem LHCOPN und deren Zusammenspiel mit der Topologie erläutert. Es werden hier die zwölf Zentren des LHCOPN betrachtet: ein Tier-0 und elf Tier-1 Zentren. In jedem dieser Zentren befindet sich ein Router mit einer bestimmten festgelegten Konfiguration bezüglich der Dienste, die darauf laufen, und der Messungen, die durchgeführt werden. Abbildung 7.1 zeigt den Aufbau eines Tier-1 Routers und die Messungen, die darauf realisiert werden. Jeder dieser genannten Router und die Verbindungen untereinander sind, im Sinne des in Abschnitt 5.2.3 entworfenen Informationsmodells, interorganisationale Ressourcen (`ioResource`), bezeichnet als Knoten (*Node*) und Links [MSFY 11].

An den Routern der jeweiligen Tier-1 Zentren befinden sich drei Server: Der erste beinhaltet eine Instanz des RRD MA (vgl. Abschnitt 4.1.1.2), die für die Abholung von Auslastungsdaten (*utilization*) und Kennzahlen zu Paketverluste an Routerinterfaces (*interface errors* und *output drops*) dieses Routers zuständig ist. Der zweite Server ist eine HADES-Box. Diese misst OWD, IPDV, OWPL und Traceroute zwischen den HADES-Boxen aller anderer zwölf Zentren. HADES ist für eine genaue Zeitsynchronisation bei der Messung mit einer GPS-

Antenne verbunden. Zusätzlich wird auf der HADES-Box ein BWCTL-MP gehostet, der für die Messung des Durchsatzes (*throughput*) zum und von Tier-0 Zentrum im acht-Stunden-Takt benutzt wird. Der dritte Server ist ein Telnet/SSH MP (vgl. Abschnitt 4.1.1.2), der zur Abholung der Konfigurationsdaten des Routers dient. Dieser ist hier nur der Vollständigkeit halber erwähnt; es führt keine eigenen Messungen durch.

Zusätzlich zu diesen Messungen auf höheren Protokoll-Schichten werden auch auf der IP-Ebene Messungen durchgeführt, indem Monitoringdaten der E2E-Links gesammelt werden. Jeder E2E-Link, der im Tier-0 oder in einem der Tier-1 Zentren (*backup links*) beginnt und in einem anderen Tier-1 Zentrum endet, durchquert mehrere administrative Domänen. Der Status dieser Links wird auf Grund der Daten der jeweiligen Managementsysteme -Network Management System (NMS)- der beteiligten Domänen berechnet.

Die Messungen sowie deren Interpretation und Visualisierung bilden die Grundlage für das Fehlermanagement. Um die korrekte Ursache eines möglichen Fehlers ermitteln zu können und dessen Behebung weiter zu verfolgen, ist die Durchführung von Messungen auf verschiedenen Protokoll-Schichten notwendig. Im Fall des LHCOPN sind die IP- und die optische Schicht relevant. Aus diesem Grund wurde in der Implementierung zwischen den unterschiedlichen Schichten (oder Topologien) E2E-Link-, HADES-, BWCTL und Routertopologien differenziert [MSFY 11].

E2E-Link-Topologie

Die E2E-Link-Topologie gibt eine Übersicht über die E2E-Links des LHCOPN. Für jeden Link, der in der Topologiedarstellung angezeigt werden soll, wird eine von zwei Abstraktionen eingeführt. Erstens kann ein dargestellter E2E-Link entweder einen E2E-Link selbst oder eine Abstraktion eines E2E-Links und seiner *Optical Protection* repräsentieren. Unter *Optical Protection* versteht man, dass im Fall eines ausgefallenen E2E-Links dieser automatisch entdeckt und nahezu zeitgleich ein anderer existierender E2E-Link geschaltet wird. Zweitens ist mit Abstraktion der Status des abstrakten Links, der aus dem Status aller zugehörigen E2E-Links berechnet wird, gemeint. Der Status der jeweiligen E2E-Links wird wie schon oben erwähnt aus den Daten der beteiligten NMS berechnet (siehe 4.1.2).

Die E2E-Link-Topologie beinhaltet somit abstrakte Knoten und abstrakte Links. Die abstrakten Knoten repräsentieren die Tier-0 und Tier-1 Zentren des LHCOPN und sind entsprechend benannt (CH-CERN, DE-KIT, NL-T1, UK-T1-RAL, NDGF, TW-ASGC, CA-TRIUMF, US-T1-BNL, US-FNAL-CMS, FR-CCIN2P3, ES-PIC, IT-INFN-CNAF). Diese abstrahieren den Standort, an dem die Messungen stattfinden, um den Zugriff auf andere Topologien zu erleichtern. Die abstrakten Links sind bidirektionale Links zwischen den abstrakten Knoten.

HADES-Topologie

Wie bereits beschrieben, befindet sich am Tier-0-Zentrum und an jedem der Tier-1-Zentren eine HADES-Box, die QoS-Messungen bereitstellt. Die HADES-Topologie besteht aus HADES-Knoten, zwischen denen sehr einfach Ende-zu-Ende Messungen durchgeführt werden können; diese Messungen werden repräsentiert durch HADES-Links (zwischen den Knoten).

Die HADES-Links sind unidirektionale Verbindungen zwischen den HADES-Knoten. Dadurch, dass diese Paare abstrakten Knoten entsprechen, werden diese auch durch geordnete Paare abstrakter Standorte (siehe E2E-Topologie) identifiziert.

Die HADES-Kennzahlen sind die IPPM-Kennzahlen OWD [RFC 2679], IPDV [RFC 3393], OWPL [RFC 2680] und der *hop count*. Dadurch, dass es sich hier um unidirektionale Links handelt, gibt es pro Paar abstrakter Knoten zwei HADES-Links. Die Kennzahlen werden im fünf-Minuten-Takt vom HADES-MA (vgl. Abschnitt 4.1.1.2) abgeholt.

BWCTL-Topologie

Die BWCTL-Topologie wird ähnlich wie die HADES-Topologie realisiert. BWCTL kontrolliert die verfügbare Bandbreite von jedem Endpunkt zu anderen Punkten, um Durchsatz-Probleme zu identifizieren. Im LHCOPN sind die BWCTL-Knoten auf einer separaten Interface-Karte der HADES-Box aktiv. Jeder BWCTL-Endpunkt entspricht einem abstrakten Knoten. Zwar ist dies eine 1:1 Abbildung, aber die IDs der BWCTL-Endpunkte entsprechen nicht exakt denen der abstrakten Knoten; es werden vielmehr IP-Adressen der BWCTL Endpunkte auf Standordnamen abgebildet.

Eine BWCTL-Durchsatzmessung (*throughput*) mit den drei Werten Minimum, Durchschnitt und Maximum wird im 45-Minuten-Takt von SQL MAs (vgl. Abschnitt 4.1.1.2) abgeholt. Allerdings wird pro BWCTL-Link der Wert nur alle acht Stunden neu gemessen. Analog zu den HADES-Links sind auch die BWCTL-Links unidirektional. Also werden auch hier zwei Links pro Paar abstrakter Knoten betrachtet.

Routertopologie

Die Routertopologie besteht entsprechend der IP-Interfaces der Router auch aus abstrakten Knoten und den dazwischenliegenden IP-Links. Ein VLAN in LHCOPN-Terminologie ist ein IP-Link. Ein abstrakter Link entspricht einem einzigen Paar von IP-Interfaces. Das heißt, wenn ein abstrakter Link aus einem oder mehreren E2E-Links besteht, werden bei einem IP-Link alle beteiligten Links miteinbezogen.

Nachdem grundlegende Informationen über die Topologie des LHCOPN als auch der darin realisierten Messungen beschrieben wurden, wird nun auf die Datenspeicherung eingegangen. Die Klassen der ioFMA müssen auf die realen Infrastrukturelemente und die darin realisierten Messungen abbildbar sein. Tabelle 7.1 führt einige dieser abgebildeten Klassen auf (diese werden dann weiter ausführlich erklärt).

Datenspeicherung

Die oben beschriebenen Topologie- und Kennzahleninformationen werden nach der Abholung (siehe Abschnitt 7.1.2) in einer Datenbasis gespeichert. Für das Deployment wurde hier eine PostgreSQL-Datenbank ausgewählt. Alternativ könnte eine XML-basierte Datenbank genutzt werden, wie beispielsweise eXist DB, aber diese ist bei solchen großen Datenmengen wie im Fall des LHCOPN nicht sehr leistungsfähig. Aufgrund der praktischen Erfahrungn im Géant- bzw. perfSONAR kann gesagt werden, dass die PostgreSQL-Datenbank bei der gegebene Datenmenge deutlich besser skaliert.

Aus historischen Gründen, die mit den ersten Implementierungen des Customer Network Management (CNM)-Systems zusammenhängen, werden die Topologiedaten von den Kennzahldaten getrennt gespeichert. Wir werden aus diesem Grund auch hier zwei separate

PIM/PSM-Info	Systemgröße	Implementierung
		Tabellen der perfSONAR DB
IOFMADomain	Domäne	domain
ioResource	Netz	network
	Knoten	network_node
	Router-Interface	network_interface
ResourceInformation		Attributen der Tabellen
		bereits implementiert in perfSONAR
ioService	E2E	E2E MP
	HADES	HADES MA
	BWCTL	BWCTL MP
	IP Routing	RRD MA
		Tabellen der csm-data DB
ServiceInformation	E2E Admin-Status	adminstate, adminstatecurrent
	E2E oper. Status	operstate, operstateecurrent
	HADES OWD	ippm_default_max(min, avg)_owd, ippm_default_max(min, avg)_owd_current
	HADES IPDV	ippm_default_max(min, avg)_ipdv, ippm_default_max(min, avg)_ipdv_current
	HADES OWPL	ippm_default_loss, ippm_default_loss_current
	HADES Hopcount	hopcount, hopcount_current
	BWCTL	bwctl_max5(min5, avg5), bwctl_max5(min5, avg5)_current
	Durchsatz	throughput, throughput_current
	Input Errors	perfsonar_ifc_errors, perfsonar_ifc_errors_current
	Output Drops	perfsonar_ifc_discards, perfsonar_ifc_discards_current

Tabelle 7.1.: Abbildung einiger ioFMA-Managementobjekte auf reale perfSONAR Infrastrukturelemente

Datenbanken nutzen: perfSONAR speichert die Topologiedaten, csm-data ist für die Speicherung der Kennzahlen zuständig.

Die perfSONAR Datenbank beinhaltet mehrere Tabellen, von denen folgende für das Deployment der ioFMA für das LHCOPN wichtig sind:

- domain: Speichert eine bestimmte Domäne mit eindeutiger ID (*domain_id*), Name (*domain_name*), Beschreibung (*domain_description*), einer externen ID (*domain_external_id*) und Land (*country_iso3166_code*). Entspricht im ioFMA-PIM der Klasse

`IOFMADomain`.

- network: Speichert Informationen über ein bestimmtes Netz mit eindeutiger ID (*object_id*), Netzname (*network_name*), Netztyp (*network_type*), Beschreibung (*description*), Gültigkeitsintervall (*valid_from und valid_to*), Gültigkeitsflag (*valid*), letztes Erscheinungsdatum (*time_last_seen*). Entspricht im ioFMA-PIM der Klasse `ioResource`. Daraus ergeben sich auch die Knoten, deren Interfaces und die Abbildung der Interfaces pro Netz, die daraus folgen.
- network_node: Dient zur Speicherung von Informationen über die Netzknoten mit den Attributen *valid_from*, *valid_to*, *time_last_seen*, *object_id*, *valid*, *name* (wie bei der Tabelle network) als auch mit den spezifischen Attributen IP-Adresse (*ip_management_address),* DNS IP-Adresse (*dns_name1*) und Standort (*location_id*)
- network_interface: Bildet die Interfaceinformation in der Datenbank ab. Beinhaltet die gemeinsamen Attribute der Tabelle network, aber auch spezifische Attribute, von denen die wichtigsten selbsterklärende Namen haben: default_ip_interface_address, default_ip_network_su if_alias, if_description, parent_interface_object_id, layer.
- network_to_interface Realisiert die Netz-zu-Interface-Abbildung in der Datenbank. Folgende Attribute werden verwendet: *network_object_id* (Netz-ID), *network_interface_object_id* (Interface-ID), *association_link_id* (ID des assoziierten Links) und die Zeitattribute (*valid_from*, *valid_to und time_last_seen*)

Eine ausführliche Darstellung der Schemata dieser Tabellen ist im Anhang VIII dargestellt.

Die csm-data Datenbank besteht aus gleichartigen Tabellen (mit demselben Tabellenformat) für jede Kennzahl. Sie beinhalten die Attribute id, das die ID der Netzkomponente (Knoten oder Link), für die dieser Wert gemessen wurde, identifiziert, *timestamp* mit dem Zeitstempel der Messung und *forwardvalue* für die tatsächliche Messung. Pro Kennzahl wird eine Tabelle für die Speicherung der letzten Messung und eine für die historischen Kennzahlen (alle Kennzahlen was älter als die letzte Messung) angelegt. Für die HADES- und BWCTL-Kennzahlen gibt es separate Tabellen für Maximum-, Minimum- und Durchschnittswerte, sowie für den aktuellen (_current) und historischen Wert.

Folgende Messungen werden in nachstehenden Tabellen gespeichert:

- Für E2E-Messungen stehen die Tabellen adminstate bzw. adminstatecurrent für den Admin-Status und operstate bzw. operstatecurrent für den operationalen Status zur Verfügung.
- Die HADES-Kennzahlen:
  - Minimal- Maximal- und Durchschnittswert der Verzögerung (OWD) sind in den Tabellen ippm_default_max_owd, ippm_default_min_owd, ippm_default_avg_owd, ippm_default_max_owd_current, ippm_default_min_owd_current und ippm_default_avg_owd_current gespeichert.

  - Minimal- Maximal- und Durchschnittswert des Jitter (IPDV) werden in den Tabellen ippm_default_max_ipdv, ippm_default_min_ipdv, ippm_default_avg_ipdv, ippm_default_max_ipdv_current, ippm_default_min_ipdv_current, ippm_default_avg_ipdv_current gespeichert.
  - Paketverlust (OWPL) ist in den Tabellen ippm_default_loss und ippm_default_loss_current enthalten
  - Hopcount wird in den Tabellen hopcount und hopcount_current abgelegt.
- Für die BWCTL-Messungen stehen die Tabellen bwctl_avg5, bwctl_min5, bwctl_max5, bwctl_avg5_current, bwctl_min5_current, bwctl_max5_current zur Verfügung.
- Für die IP-Messungen werden
  - der Durchsatz in throughput und throughput_current
  - die Input Errors in perfsonar_ifc_errors und perfsonar_ifc_errors_current
  - die Output Drops in perfsonar_ifc_discards und perfsonar_ifc_discards_current

  gespeichert.

Diese Tabellen entsprechen in der ioFMA-PIM der Klasse `ServiceInformation` und die interorganisationalen Dienste, die in Géant erbracht werden, der Klasse `ioService`.

Die Topologie- und Kennzahldaten werden mit Hilfe von Perl-Skripten (Characteristic.pm, CharacteristicStore.pm, CharacteristicData.pm) gespeichert und aktualisiert. Der Einsatz von Skriptsprachen ist laut MDA-Ansatz für die Implementierung sehr geeignet, da sie sehr leicht anpassbar sind, falls sich die Transformationsbeschreibungen ändern. Sie sind schnell ausführbar, da sie keinen Compile-Lauf benötigen – damit ist auch der Zugriff auf die Objekte sehr schnell [GPR 06]. Wegen dieser Eigenschaften werden Perl-Skripte nicht nur für die hier beschriebene Informationsverwaltung eingesetzt, sondern im folgenden auch für anderen Bausteine.

7.1.2. Entwurf des Kommunikationsbausteines

In Anlehnung an das Network Measurement Working Group (NMWG)-Schema kann das Kommunikationsmodell fast ausschließlich auf die in dieser Schema definierten Request-/Response-Operationen abgebildet werden, insbesondere wenn es sich um die Anforderung und Abholung von Daten handelt. Ein Überblick über einige abgebildete Operationen der ioFMA, die innerhalb des Kommunikationsbausteines benötigt werden, wird in Tabelle 7.2 gegeben. Um sowohl die NMWG-Request/Response Operationen, als auch diejenigen, die zur Speicherung der Daten dienen, anzustoßen, werden Perl-Skripte genutzt. Im folgenden werden die hier zusammengefassten Operationen ausführlich beschrieben. Zuerst werden allgemein die Request/Response Operationen der NMWG beschrieben und danach auf den konkreten Fall der ioFMA angewendet.

ioFMA-PIM/PSM-Operationen	LHCOPN-Implementierung
getServInformation()	NMWG-Message SetupDataRequest
setServInformation()	NMWG-Message SetupDataResponse
getStateInformation()	e2emon-topo-and-status-import.pl
getStatisticalInfo()	CharImportDataLogic.pm
getResInformation()	lhcopn-bwctl-mapping-import.pl, lhcopn-maintopo-import.pl, e2emon-topo-and-status-import.pl,
getResource()	e2emon-topo-import.pl, LHCOPN_Main.pm
affects()	lhcopn-fetchstatus-report.pl
aggregateInformation()	IPPMMetricAggregation.pm

Tabelle 7.2.: Ausgewählte Beispiele für Implementierungen der ioFMA-Operationen

Beispielhaft werden hier die folgenden zwei Operationen abgebildet: In Listing 7.1 ist ein einfaches Echo-Request im Format des NMWG-Schemas dargestellt, in Listing 7.2 wird die Antwort als Echo-Request zurückgegeben. Der Request besitzt einen einzigen Teil, und zwar das Metadata-Tag, durch den das Wichtigste bezüglich der angeforderten Information angegeben wird (z.B. die Art der Anfrage, die ID der angefragten Entität, die angeforderte Kennzahl). Die Antwort hingegen ist in zwei Teile gegliedert: einer, der die Metadata-Informationen liefert, und der zweite, der die tatsächlichen Daten (Informationen), die durch den Request angefordert wurden, zurückgibt.

Listing 7.1: Beispiel eines NMWG Echo Request

```
<nmwg:message type="EchoRequest" id="EchoMessage1">
 <nmwg:metadata id="metadata1">
  <nmwg:eventType>http://schemas.perfsonar.net/tools/admin/echo/2.0
  </nmwg:eventType>
 </nmwg:metadata>
 <nmwg:data id="data1" metadataIdRef="metadata1"/>
</nmwg:message>
```

Listing 7.2: Beispiel eines NMWG Echo Response

```
<nmwg:message messageIdRef="EchoMessage1" id="message.11515259"
                                        type="EchoResponse">
 <nmwg:metadata metadataIdRef="metadata1" id="metadata.6012497">
  <nmwg:eventType>success.echo</nmwg:eventType>
 </nmwg:metadata>
 <nmwg:data metadataIdRef="metadata.6012497" id="data.2038174">
  <nmwgr:datum>The echo request has passed.</nmwgr:datum>
 </nmwg:data>
 <nmwg:metadata id="metadata1">
  <nmwg:eventType>http://schemas.perfsonar.net/tools/admin/echo/2.0
  </nmwg:eventType>
 </nmwg:metadata>
</nmwg:message>
```

Diese Request/Response (Anfrage/Antwort) Konstrukte wurden im Kommunikationsmodell für die spezielle Abholung der Daten von den unterschiedlichen Messpunkten des perfSONAR Systems angepasst. Die abstrakte Topologie, assoziiert mit den existierenden E2E-Links, wird von einer Konfigurationsdatei, die vom Service Desk in DANTE zur Verfügung gestellt wird, abgeholt.

Die E2E-Link-Topologie und die Statusdaten der Links werden von der E2EMon Export-Schnittstelle abgeholt. Ein aggregierter Status des abstrakten Links wird sofort bei der Datenabholung berechnet. Wenn ein E2E-Link eine *optical protection* besitzt, muss für den abstrakten E2E-Link ein Status aus denen der jeweiligen E2E-Links bzw. dessen *optical protection* berechnet werden (s. vorherigen Abschnitt) Es wurden in diesem Zusammenhang vier aggregierte Werte für den Status definiert:

- down: wenn mindestens einer der zugehörigen E2E-Links ausgefallen ist
- warning: wenn keiner der E2E-Links ausgefallen ist, aber sich mindestens einer im Status warning befindet.
- unknown: wenn der Status mindestens einer der zugehörigen E2E-Links unbekannt ist; d.h. dieser Link konnte nicht gemessen werden oder die Messungen konnten nicht abgeholt werden (der Zustand der anderen Links ist weder down noch warning).
- up: wenn jeder der zugehörigen E2E-Links ordnungsgemäß funktioniert.

Die HADES-Topologie und die zugehörigen Daten werden von der Metadaten der LHCOPN HADES MA importiert. Die HADES-Knoten werden 1:1 auf die abstrakten Knoten abgebildet. Die BWCTL-Topologie wird analog von der Metadaten der LHCOPN BWCTL SQL MA abgeholt und die Knoten werden auch hier 1:1 auf die abstrakten Knoten abgebildet.

Die mögliche Paare von LHCOPN-IP-Interface-Adressen werden von der Metadaten der LHCOPN RRD MAs abgeholt und dann auf die abstrakten Link abgebildet (vgl. Abschnitt 7.1.1).

Die Datenabholung ist mit einem Pull-Kommunikationsmechanismus realisiert. Die Topologie- und Kennzahldaten werden in fünf-Minuten-Takt von der hier entwickelten Plattform abgeholt.

Die Daten (für gewöhnlich Kennzahldaten, Änderungen der Topologiedaten) werden durch SOAP-Requests abgeholt. Diese werden dann in der WSDL Datei als Operationen-`Bindings` referenziert. Für das LHCOPN werden für die abgefragten Gruppen von Metriken bzw. Topologiedaten unterschiedliche Request-Operationen zusammengeführt. Im Listing 7.3 ist die übergreifende Anfrage, die für die Abholung der HADES-Daten zuständig ist, abgebildet. Die Metadaten zeigen dass es sich um HADES-Daten und ein bestimmtes Zeitintervall handelt. Der Typ der ausgetauschten Nachricht in dieser Kommunikation ist `SetupDataRequest`, um zu kennzeichnen, dass es sich um eine Anfrage handelt.

Listing 7.3: Hades-Request für das LHCOPN

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope ...>
 <soap:Body>
 <nmwg:message ... type="SetupDataRequest">
  <nmwg:metadata id="metadata1">
    <perfsonar:subject id="subject1" />
    <nmwg:eventType>http://ggf.org/ns/nmwg/tools/hades/</nmwg:eventType>
    <nmwg:parameters id="param1">
     <nmwg:parameter value="0x0" name="precedence" />
     <nmwg:parameter value="9" name="groupsize" />
     <nmwg:parameter value="60" name="interval" />
     <nmwg:parameter value="41" name="packetsize" />
    </nmwg:parameters>
  </nmwg:metadata>

  <nmwg:metadata id="metadata1-2">
    <perfsonar:subject metadataIdRef="metadata1" id="subject1-2" />
    <nmwg:eventType>http://ggf.org/ns/nmwg/ops/select/2.0</nmwg:eventType>
    <nmwg:parameters id="param1-2">
     <nmwg:parameter name="startTime">1294666800</nmwg:parameter>
     <nmwg:parameter name="endTime">1294667100</nmwg:parameter>
    </nmwg:parameters>
  </nmwg:metadata>

  <nmwg:data metadataIdRef="metadata1-2" id="data1" />

 </nmwg:message>
 </soap:Body>
</soap:Envelope>
```

Für die HADES-Daten im abgefragten Zeitraum wird anschließend eine Antwort (SOAP-Response) geliefert. Diese beinhaltet, wie auch die Anfrage, erst die Metadaten und zusätzlich zu den abgefragten Kennzahlen auch die Beschreibung der Links mit den Namen und IP-Adressen des Empfängers und des Senders (vgl. Listing 7.4). Für jeden Link gibt es jeweils ein Metadatum für jede Richtung. Im Gegensatz zur Anfrage ist die hier ausgetauschte Nachricht vom Typ `SetupDataResponse`. Die vollständige Request/Response Operation für den Link DE-KIT – CH-CERN und seine Umkehrrichtung ist im Anhang VI zu finden.

Listing 7.4: Teil des Hades-Response für das LHCOPN (Link DE-KIT – CH-CERN)

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope ...>
 <soap:Body>
 <nmwg:message ... type="SetupDataResponse">
 ...
  <nmwg:metadata id="result30">
   <nmwg:parameters id="param1">
     <nmwg:parameter value="CH-CERN-HADES" name="receiver"/>
     <nmwg:parameter value="9" name="groupsize"/>
     <nmwg:parameter value="60" name="interval"/>
     <nmwg:parameter value="15652" name="mid"/>
     <nmwg:parameter value="0x0" name="precedence"/>
     <nmwg:parameter value="128.142.223.231" name="receiver_ip"/>
     <nmwg:parameter value="DE-KIT-HADES" name="sender"/>
     <nmwg:parameter value="192.108.46.124" name="sender_ip"/>
```

```
      <nmwg:parameter value="41" name="packetsize"/>
     </nmwg:parameters>
   </nmwg:metadata>

   <nmwg:data metadataIdRef="metadata1-2" id="data1"/>
  ...
  </nmwg:message>
  </soap:Body>
</soap:Envelope>
```

Nachdem die allgemeine Anfrage/Antwort realisiert wurde, in der alle existierenden Links vorhanden sind, wird für jede dieser Leitungen (für jede der beiden Richtungen) eine gesonderte Abfrage der Kennzahlen, die diese Leitung charakterisieren, erstellt. Die Kennzahldaten sind im Rohformat dargestellt. Diese werden in die PostgreSQL-Datenbank gespeichert. Ein Beispiel einer solcher Request/Response Operation für eine bestimmte Richtung eines Links (DE-KIT zu CH-CERN) ist in Listing 7.5, für beide Richtungen ausführlich in Anhang VII gezeigt.

Listing 7.5: Hades-Request/Response für Kennzahlen des Links DE-KIT – CH-CERN

```
<!--Request-->
...
<soap:Body>
 <nmwg:message ... type="SetupDataRequest">
 ...
   <nmwg:metadata id="metadata1">
    <perfsonar:subject id="subject1">
     <nmwgt:endPointPair>
      <nmwgt:src value="CH-CERN-HADES" type="IFName" />
      <nmwgt:dst value="DE-KIT-HADES" type="IFName" />
     </nmwgt:endPointPair>
    </perfsonar:subject>
   <nmwg:eventType>http://ggf.org/ns/nmwg/tools/hades/</nmwg:eventType>
   ...
   </nmwg:metadata>
 ...
 </nmwg:message>
 </soap:Body>
<!--Response-->
...
 <soap:Body>
  <nmwg:message ... type="SetupDataResponse">
  ...
   <nmwg:metadata id="metadata1">...</nmwg:metadata>
   <nmwg:data metadataIdRef="metadata1-2" id="data1">
    <default:datum
      xmlns="http://ggf.org/ns/nwmg/tools/hades/aggregated"
      med_delay="0.00570082664489746" med_ipdv_jitter="-5.96046447753906e-06"
      min_ipdv_jitter="-0.000265836715698242"
      time="1294666803.20085"
      sync="yes" min_delay="0.00568294525146484"
      max_delay="0.00577592849731445"
      duplicates="0" loss="0"
      max_ipdv_jitter="9.20295715332031e-05"/>
    ...
   </nmwg:data>
   ...
```

```
  </nmwg:message>
</soap:Body>
```

Die Request/Response Operationen entsprechen den getter- und setter-Operationen der ioFMA-PIM bzw. PSM. Die Request-Operationen werden automatisch im fünf-Minuten-Takt mit Hilfe von Perl-Skripten angestoßen. Diese holen regelmäßig die neuesten Topologie- und Kennzahldaten ab (z.B. e2emon-topo-and-status-import.pl, lhcopn-bwctl-mapping-import.pl, lhcopn-maintopo-import.pl, CharImportDataLogic.pm usw.) und aktualisieren die Datenbank entsprechend.

Die Informations- und Kommunikationsbausteine bilden die Grundlage für das Deployment und sind daher auch am stärksten ausgeprägt.

7.1.3. Entwurf von Basisanwendungen

Sobald das Kernsystem realisiert ist (die Infrastruktur der Plattform besteht aus Informations- und Kommunikationsbaustein), können die Basisanwendungen darauf aufbauen. Im Fall eines interoganisationalen Werkzeugs sind die eingesetzten Basisanwendungen stark auf die Informationen der beteiligten Domänen angewiesen. Mögliche nennenswerte Basisanwendungen sind in diesem Kontext folgende:

Die Zustandsüberwachung der externen Ressourcen (E2EMon, RRD MA, HADES MA, BWCTL MP usw.) wird gleichzeitig mit der Abholung der Daten im fünf-Minuten-Takt realisiert. Als Kommunikationsprotokoll kommt SOAP zum Einsatz. Die damit gesammelten Informationen werden dann entsprechend in der Benutzeroberfläche angezeigt.

Das Alarmierungs- und Ereignismanagement sind ebenso wichtige Basisanwendungen, in einem interorganisationalen System aber nicht leicht zu realisieren und einzuführen. Deshalb wurde zunächst ein sehr vergleichsweise einfaches Mail-basiertes Alarmierungssystem realisiert. Es besteht aus zwei Teilen: einem für die Alarmierung der Domänen und einem für die Tool-Admins. Die Idee dahinter ist, dass fehlerhafte oder unstimmige Lieferungen von Daten der Domänen zu Benachrichtigungen führen, die die Domänen-Admins über mögliche „lokale" Probleme, die die Funktionalität des interorganisationalen Tools beeinträchtigen, informiert. Diese Funktionalität wird mit Hilfe einiger Perl-Skrips realisiert (*notify_admins_bymail*, *notify_admins_bymail_bulk*). Der zweite Teil überprüft, ob die Werkzeug-internen Prozesse ordnungsgemäß funktionieren. Dafür wurden Schwellwerte für die Laufdauer der unterschiedlichen Prozesse festgelegt. Bei Überschreitung dieser Werte wird den Tool-Admins eine entsprechende Benachrichtigung gesendet (z.B. *monitor_process_db*, *monitor_longrunning_perl_scripts*, *monitor_longrunning_busy_perl_scripts*).

Das Topologiemanagement, durch das möglichst viele Informationen über die Netze der beteiligten Domänen abgeholt und in der Datenbank der Plattform gespeichert werden, ist

schon bei den Informations- und Kommunikationsbausteinen beschrieben worden. Das Topologiemanagement kann in diesem Fall nicht eigenständig betrachtet werden, da die Informationsartefakte die Topologie selbst und deren beschreibende Informationen beinhalten. Ein wichtiger Teil davon ist die Topologieabstraktion der E2E-Links, die die Bearbeitung der Topologiedaten durchführt, um sie zu visualisieren bzw. mit anderen Informationen (z. B. HADES, BWCTL u. a.) zu koppeln.

7.1.4. Entwurf von Managementanwendungen

Zusätzlich zu den Basisanwendungen sind im Fall des interorganisationalen Fehlermanagement auch grundlegende Managementanwendungen wie z. B. der Transformationsdienst und das Management des Abholstatus (Fetchstatus Management) zu berücksichtigen.

Der Transformationsdienst realisiert Transformationen der Rohdaten (entsprechend der Anforderungen der Kunden, Anwender usw.) in eine geeignete Form. Ein Spezialfall davon ist der Aggregationsdienst. Die Kennzahlenaggregation wird z. B. eingesetzt, wenn viele Daten abgeholt werden, für die (spätere) Anzeige aber nur ein bestimmtes Zeitintervall benötigt wird. Beispielsweise ist dieses bei HADES-Daten der Fall; diese werden minütig aktualisiert/erstellt, aber nur im fünf-Minuten-Takt abgeholt. Daher muss eine Aggregation dieser Daten vorgenommen werden. Das Perl-Skript IPPMMetricAggregation.pm realisiert diese Funktion und ist hiermit eine Implementierung der Operation *aggregate()*. Es werden Aggregationsregeln für jede Kennzahl festgelegt. Für die Kennzahlen, die einen Minimal-, Maximal- und Durchschnittswert aufweisen, wird folgendermaßen vorgegangen: Der berechnete Maximal-/Minimalwert ist das Maximum/Minimum der abgeholten jeweiligen Werte im betreffenden Zeitintervall. Der Durchschnittswert entsteht aus dem Durchschnitt der Durchschnittswerte im Zeitintervall. Bei Paketverlust werden die Werte für das Zeitintervall addiert.

Das Management des Abholstatus (Fetchstatus Management) wurde eingeführt, um die Vollständigkeit der abgeholten Daten zu gewährleisten. Da in einer interorganisationalen Umgebung immer mehrere Domänen beteiligt sind, die regelmäßig Daten und Informationen zur Verfügung stellen, ist es äußerst wichtig, deren Vollständigkeit, Erreichbarkeit, als auch mögliche organisatorische Änderungen innerhalb dieser Domänen zu beobachten. In diesem Sinne wurde eine Managementanwendung konzipiert (getriggert von einem Perl Skript lhcopn-fetchstatus-report.pl), die die Verfügbarkeit der Domänen, die Vollständigkeit der Daten und mögliche aufgetretene Probleme in den Domänen anzeigt. Sie zeigt in einer dynamischen Webseite eine Übersicht über alle vom Werkzeug benötigten Daten pro Domäne, Link und Knoten. Sie beinhaltet jeweils einen Teil für jede Topologie, worin die Verfügbarkeit des Dienstes und die Vollständigkeit der Kennzahlen angezeigt wird. Abbildung 7.2 zeigt einen Ausschnitt dieser Seite (sieh auch Abschnitt B.1). Dies implementiert als die Assoziation/Operation *affects()* zwischen den Klassen `ServiceFailure` und `Service` als auch zwischen `ResourceAlarm` und `Resource`.

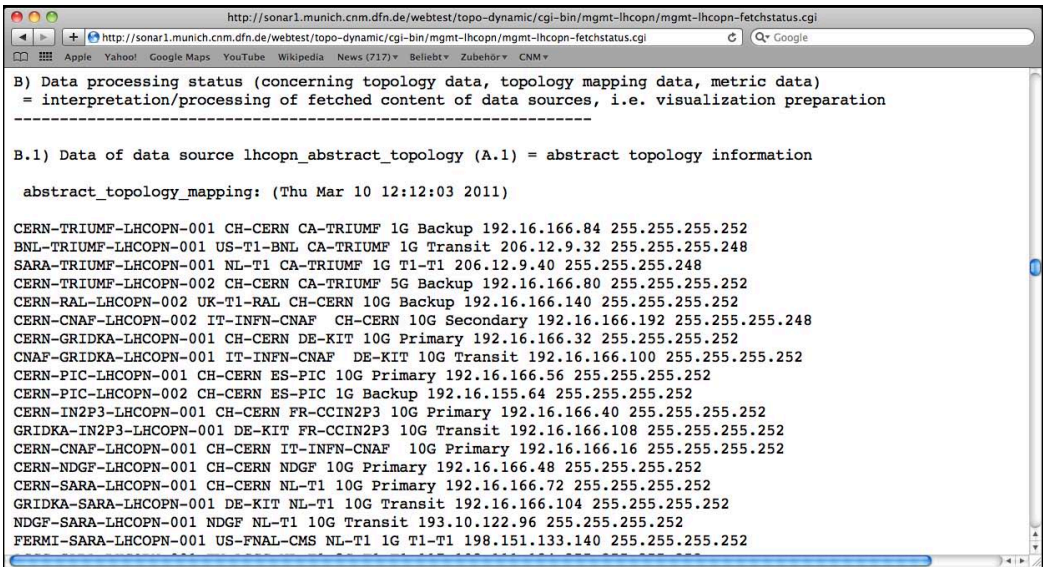


Abbildung 7.2.: Fetchstatus Management Seite für das LHCOPN

7.1.5. Entwurf der Oberflächenbausteine

Die vorher entwickelten Bausteine sind als Server-Komponenten grundlegend für die Plattform und insbesondere für die Oberflächenbausteine, die die Visualisierung der darunter liegenden Daten, Informationen und Funktionen für den Anwender in einer benutzerfreundlichen Art realisieren (Client). Die oben vorgestellte Topologie mit den zugehörigen Kennzahldaten werden gemäß der Kundenanforderungen [LHC 09] in der „LHCOPN-Wetterkarte" abgebildet. Deren Aufbau und Implementierung wurde in ein vorheriges Paper [MSHT 10] beschrieben. Die LHCOPN-Wetterkarte besteht aus drei Komponenten, aufgeteilt in drei Reiter: die Übersichtskarte des LHCOPN, die Kennzahlkarte und die E2E-Link-Karte.

Übersichtskarte

Die Übersichtskarte stellt, wie die Name schon sagt, eine Übersicht über die abstrakte Topologie des LHCOPN bereit. Es werden die abstrakten Knoten (Tier-0 und Tier-1 Zentren) und die dazwischen liegenden abstrakten Links (mit dem Tier-0-Zentrum CH-CERN in der Mitte) abgebildet. In dieser Übersichtskarte werden die abstrakten Links in unterschiedlichen Farben, die den Zustand des abstrakten Linkes, wiedergegeben (vgl. Abschnitt 7.1.2). Es werden die Farben rot, gelb, grün und blau für den entsprechenden Status des Links down, warning, up und unknown eingesetzt. Zusätzlich zu diesen wird ein sechster Status eingeführt, der keine korrespondierende aggregierte Statuskennzahl in der E2E-Link-Topologie hat. Die Farbe magenta deutet auf eine grundlegende Diskrepanz in der Topologieabbildung. Das heißt, dass keiner der E2E-Links (aus der Sicht der abstrakten Topologie) auf die E2E-Links (aus Sicht der E2E-Topologie) abgebildet werden kann. Dieser Status wird unbekannte Topologie genannt und zeigt, dass für diesen Link kein aggregierter Status berechnet werden

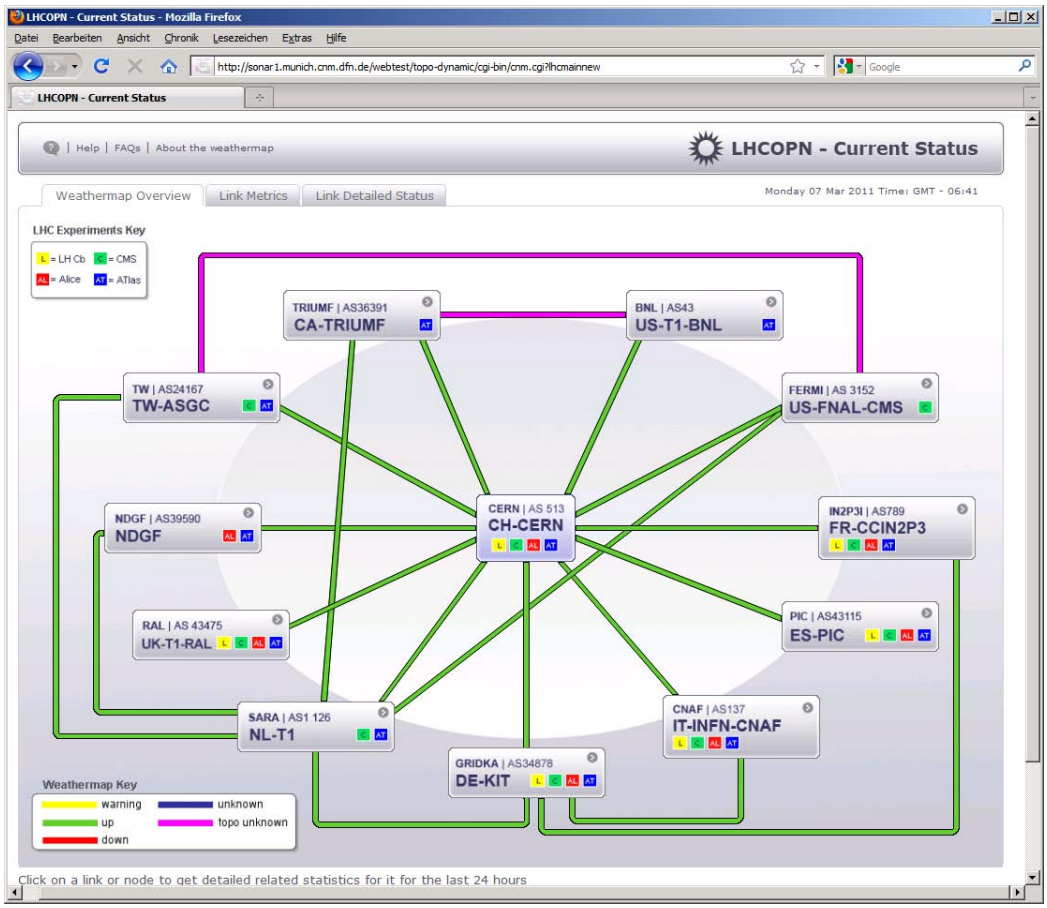


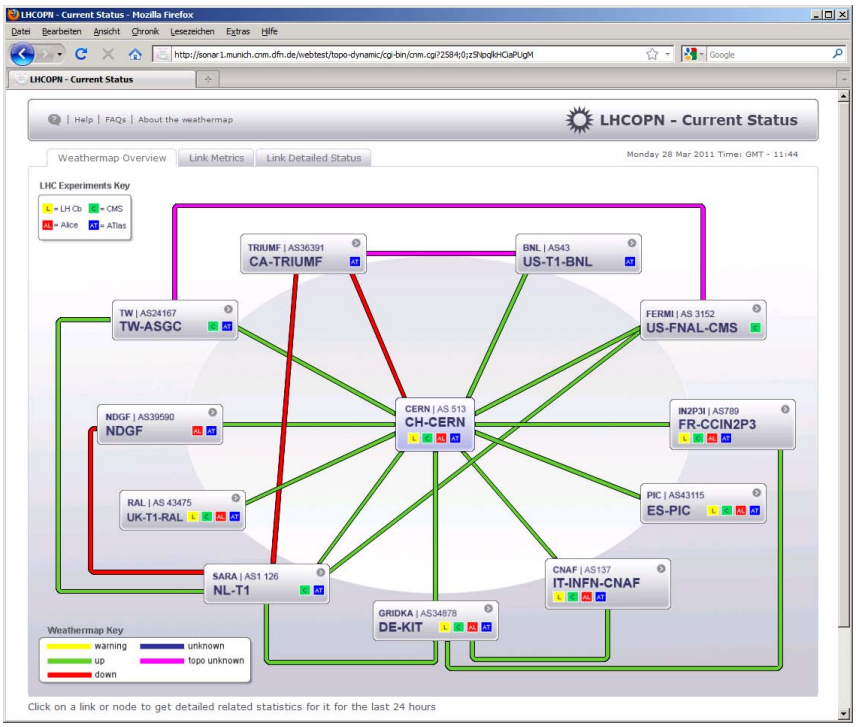
Abbildung 7.3.: Die Übersichtskarte

konnte. Somit kann man anhand der Farbe in der Karte einen Fehler oder ein Fehlverhalten im LHCOPN erkennen, ihm weiter nachgehen, um die Fehlerursache zu entdecken und zu beheben. Abbildung 7.3 zeigt die Übersichtkarte mit deren Links in funktionsfähigen Zustand.

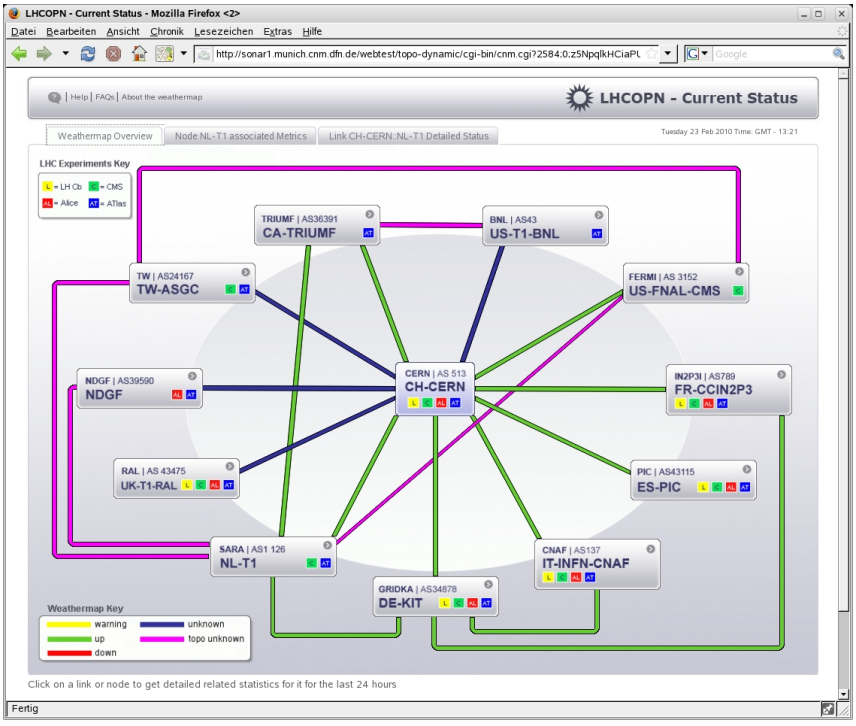
Desweiteren werden in Abbildung 7.4 zwei andere Darstellungen der Übersichtskarte für einen anderen Status der abstrakten Links aufgeführt

Kennzahlenkarte

Die Kennzahlenkarte zeigt statistische Kennzahlgraphen der jeweiligen abstrakten Links. In der Kennzahlkarte werden 24-Stunden-Graphen der Kennzahlen, die in Abschnitt 7.1.1 genannt wurden, angezeigt. Die visualisierten Metriken sind abhängig vom gewählten abstrakten Element (Link oder Knoten). Grundsätzlich gibt es in den Kennzahlgraphen immer zwei Spalten mit Graphen –eine für jede Richtung eines Links. Für alle Graphen gibt es Tooltips, die beim Überfahren mit der Maus die entsprechenden Werte anzeigen oder beim Anklicken die Vergrößerung des Graphen produzieren.



(a)



(b)

Abbildung 7.4.: Übersichtskarte des LHCOPN mit unterschiedlichen Status der abstrakten Links

Abbildung 7.5.: Die linkbezogene Kennzahlkarte

...für abstrakte Links

Wenn in der Übersichtkarte ein bestimmter abstrakter Link ausgewählt wird, werden die zugehörigen Kennzahldaten angezeigt. In diesem Fall wird in der obersten Reihe der aggregierte Status des ausgewählten E2E-Link dargestellt. Die Berechnungsvorschrift, die hinter diesem Wert des Status steht (vgl. Abschnitt 7.1.2) und die dafür in der Visualisierung vorgesehenen Farben (siehe Übersichtskarte) wurden bereits beschrieben. Die zweite und dritte Reihe zeigen RRD MA Kennzahlgraphen (Utilization und Input Errors/Output Drops) für den IP-Link des korrespondierenden abstrakten Links (siehe Abbildung 7.5). Die Graphen werden immer im fünf-Minuten-Takt mit den neu abgeholten Daten aktualisiert.

...für abstrakte Knoten

Wenn ein Tier-1-Zentrum ausgewählt wird, so wird der dazugehörige abstrakte Link zwi-

Abbildung 7.6.: Die knotenbezogene Kennzahlkarte

schen diesem und dem Tier-0 Zentrum ausgewählt. Wenn das Tier-0 ausgewählt wird, werden alle zu den Tier-1-Zentren gehenden abstrakten Links ausgewählt. Für den ausgewählten

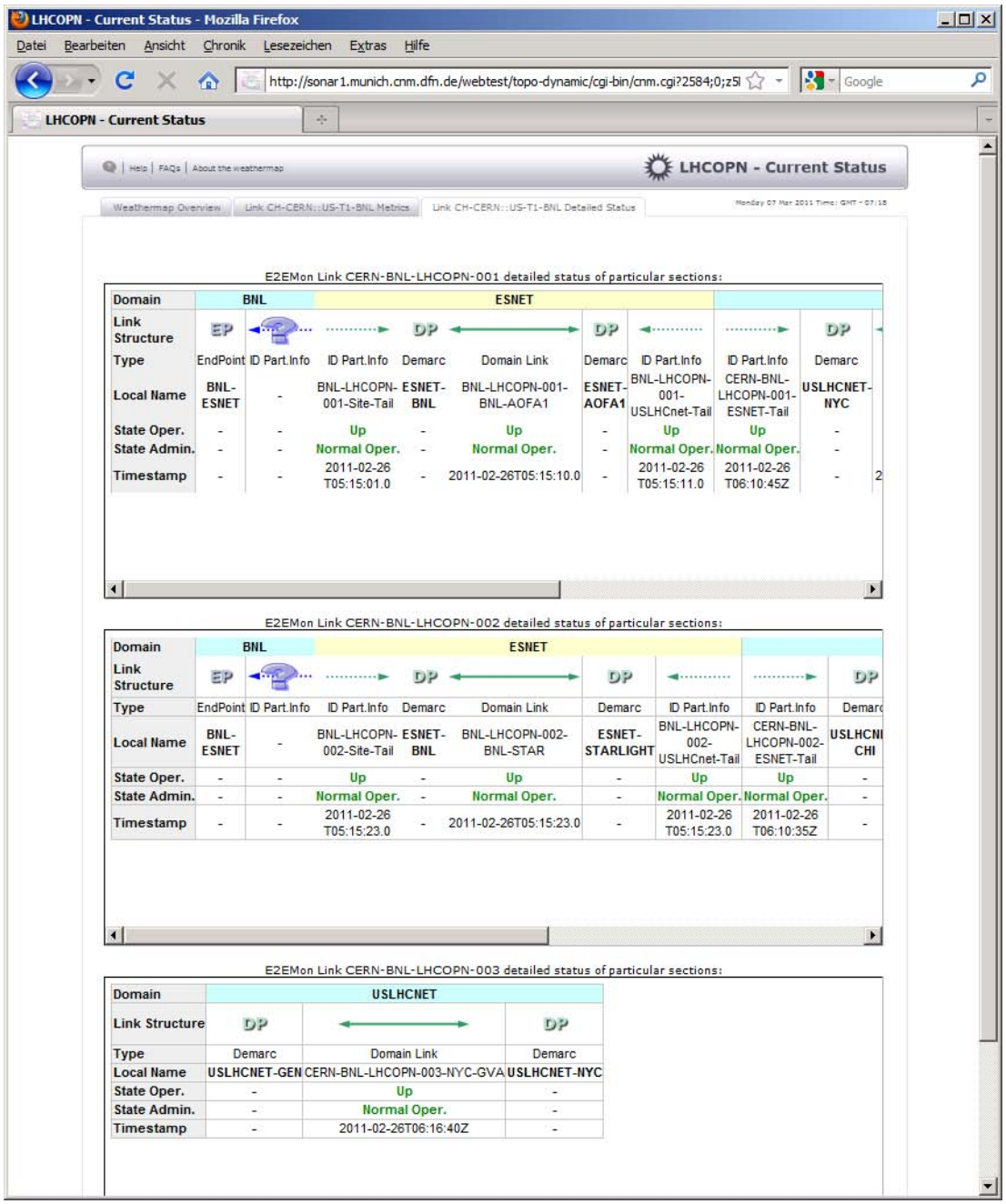


Abbildung 7.7.: Die E2E-Link-Karte

Link werden die HADES- und BWCTL-Metriken angezeigt (siehe Abbildung 7.6). In der ersten Reihe befindet sich der Hop count Kennzahlgraph, der für jede geänderte Route eine andere Spaltenfarbe nutzt. In den Reihen zwei, drei und vier werden die HADES Kenn-

zahlen graphisch dargestellt (OWD, IPDV und Paketverlust). Die Werte dieser Messungen werden im fünf-Minuten-Takt aktualisiert und als Punkte (einer pro 5 Minuten) in den Graphen dargestellt. Für OWD und Jitter werden pro Zeitpunkt jeweils drei Werte für Maximalwert (rot), Durchschnittswert (blau) und Minimalwert (grün) angezeigt. In der letzten Reihe befindet sich der Graph, der die Maximalwerte (rot), Durchschnittswerte (grün) und Minimalwerte (blau) der BWCTL-Thoughput Messungen anzeigt. Obwohl die Datenabholung im fünf-Minuten-Takt stattfindet, werden pro Link nur alle acht Stunden neue Werte geliefert.

E2E-Link-Karte

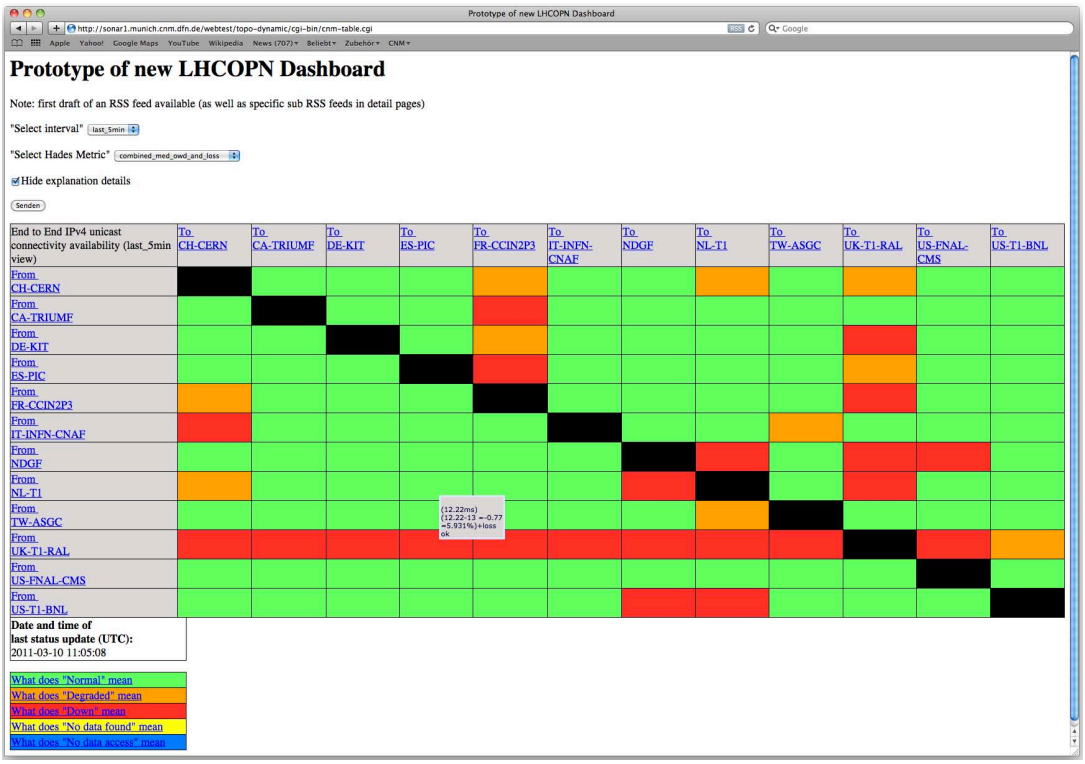
In der Kennzahlenkarte werden die Kennzahlen der unterschiedlichen Netzschichten einer Ende-zu-Ende zwischen Tier-0 und Tier-1-Standorten dargestellt. Die E2E-Link-Karte ergänzt diese Darstellung mit einem direkten Einblick in die E2E-Links, die sich hinter den abstrakten Links verbergen. Dies wird durch Einbau des Statusabschnitts der HTML-Seite des E2EMon-Tools in die E2E-Link-Karte realisiert. Wenn in der Übersichtskarte ein abstrakter Link ausgewählt wurde, wird das E2EMon-Segment aller assoziierten E2E-Links angezeigt. Wenn ein Tier-1-Standort ausgewählt wurde, wird für alle zum abstrakten Link zum Tier-0-Standort gehörende E2E-Links das entsprechende E2EMon-Segment abgebildet. Abbildung 7.7 zeigt ein Beispiel für eine E2E-Link-Karte eines abstrakten Links mit drei zugehörigen E2E-Links.

Zusätzlich zur Karten-Sicht wurde auf Grund von Kundenanfragen eine Matrix geschaffen, in der für eine ausgewählte Kennzahl oder mehrere korrelierte Kennzahlen eine Übersicht über alle n:m LHCOPN-Verbindungen (abstrakte Links) gegeben wird. Abbildung 7.8(a) zeigt das sogenannten Dashboard des LHCOPN. Hier werden die abstrakte Knoten in Feldern oben und an der Seite der Matrix dargestellt. Die Matrixfelder repräsentieren die abstrakten Links mit dem Wert der ausgewählten Kennzahl und dem Zeitintervall (links oben kann man diese Kennzahl auswählen). Die Felder zeigen über deren Farben den Status dieses Links bezüglich der gewählten Kennzahl. Die Schwellwerte für den Status dieser Felder werden auf Grund der Kundenanforderungen gesetzt. Grün zeigt den ,,normal"-Zustand, rot bedeutet dass der Link im Zustand ,,alarm" (oder ausgefallen) ist, orange deutet auf eine Warnung hin, gelb zeigt, dass keine Daten zur Verfügung stehen, und blau weist auf ein Datenzugriffsproblem hin.

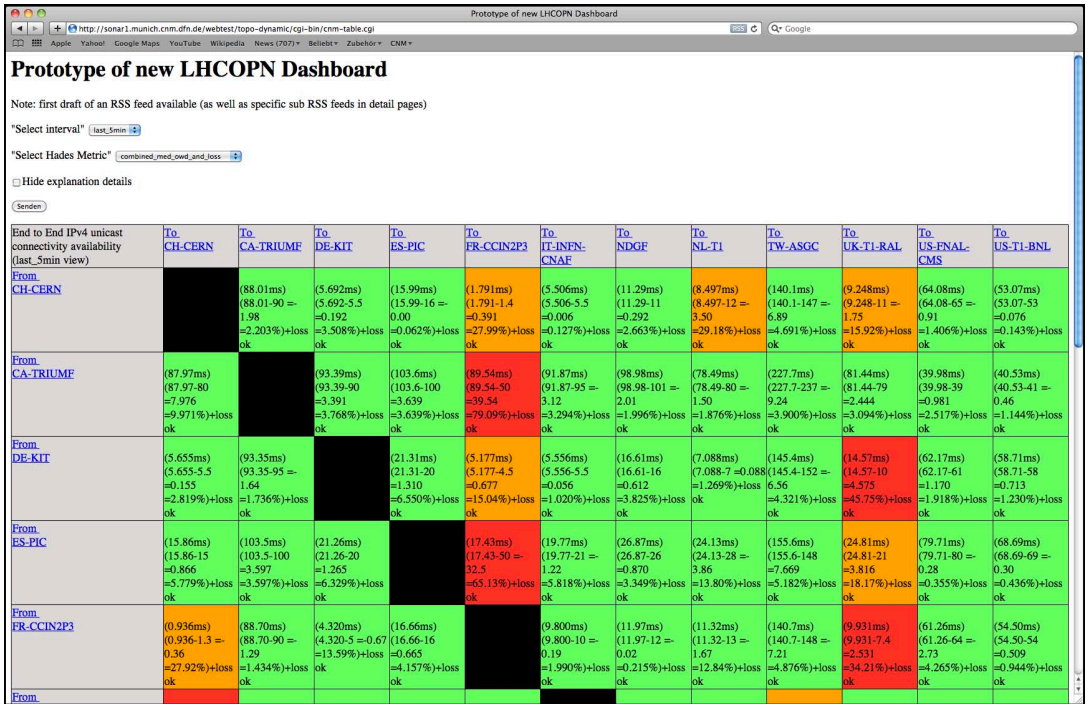
Das LHCOPN-Dashboard kann entweder nur anhand der Farben, aber auch in einer anderen Sicht mit genauen Angaben der Kennzahlwerte, wie in Abbildung 7.9 gezeigt, dargestellt werden. Es können ebenso auch alle von einem Knoten aus- oder eingehenden Links mit ein- oder ausgeblendeten Kennzahlwerten von bzw. zu einem bestimmten abstrakten Knoten angezeigt werden (Abbildung 7.8(b)).

7.1.6. Zusammenfassung der LHCOPN-Wetterkarte

In den vorigen Abschnitten 7.1.1 bis 7.1.5 wurden die unterschiedlichen Bausteine, die zur Realisierung des Werkzeugs beitragen, in ihrer Entstehungsreihenfolge beschrieben. Nun



(a) Ohne ausgeführten Kenzahlwerte



(b) Mit ausgeführten Kenzahlwerte

Abbildung 7.8.: Das LHCOPN Dashboard (Hauptansicht)

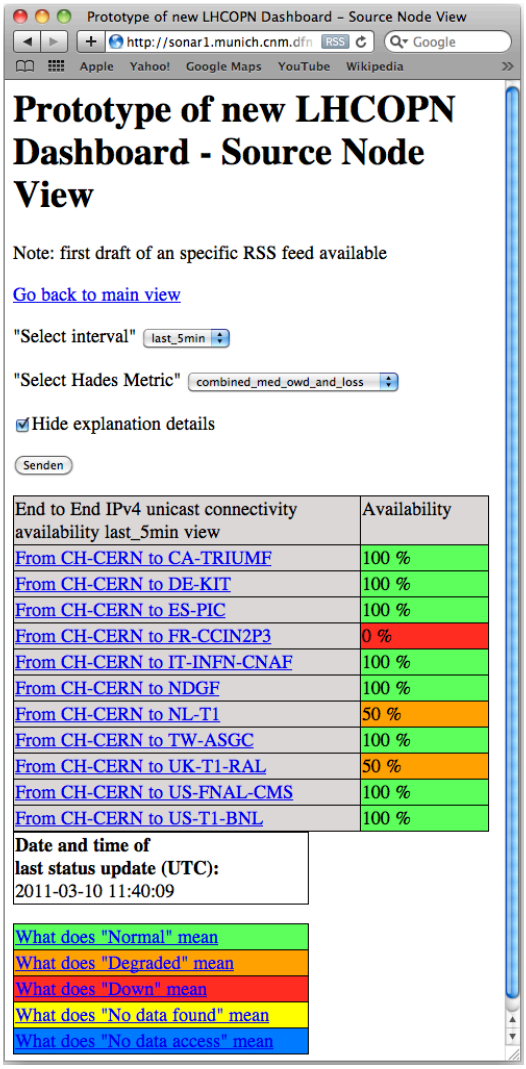


Abbildung 7.9.: Das LHCOPN Dashboard (Detail-Sicht pro Link)

werden diese in Form einer Client-Server Darstellung zusammengefasst. Abbildung 7.10 zeigt die Architektur der LHCOPN-Wetterkarte in einer einfachen Darstellung.

Es können dabei drei größere Komponente identifiziert werden (v.l.n.r.): Client-Seite (Browser und der Client selbst), Server-Seite (Apache-Server und der Server der Anwendung) sowie die Ressourcen.

Die Anwender (in diesem Fall die LHCOPN Betriebsgruppe) rufen im Browser eine Karte ab. Im Client sind schon alle Registerkarten vorbereitet, um mit den entsprechenden aufgerufenen Daten gefüllt werden zu können. Um diese Schablone mit Daten zu füllen, wird ein cgi-skript auf dem Server aufgerufen. Dazwischen ist ein Apache-Server geschaltet, über

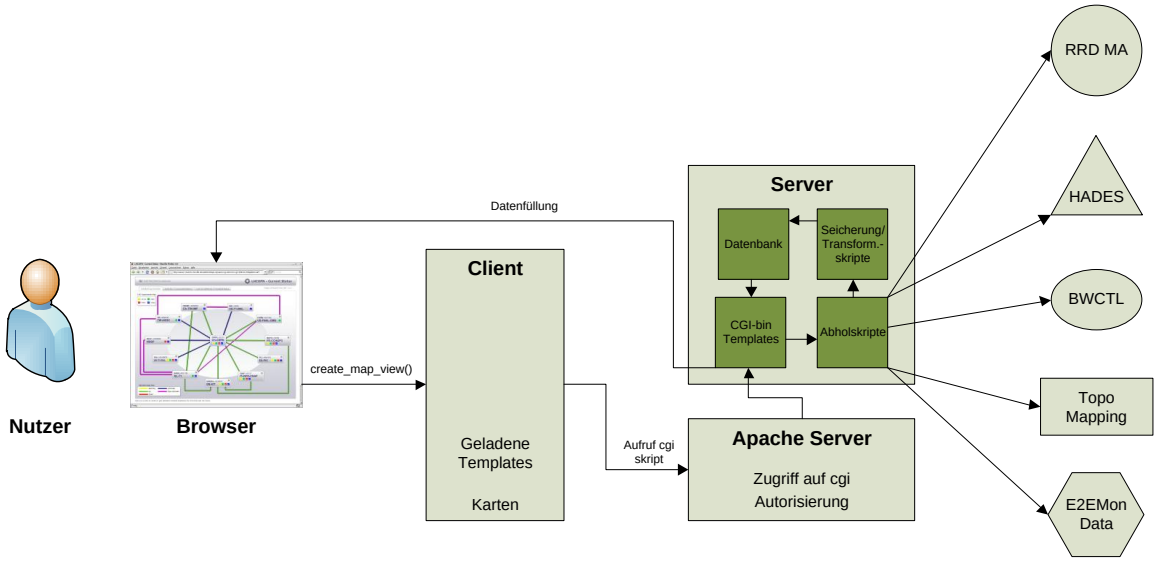


Abbildung 7.10.: Client-Server-Architektur der LHCOPN-Wetterkarte

den auch die Autorisierung läuft. Es wird geprüft, ob aktuelle Kennzahlen vorhanden sind. Die Daten werden aus der Datenbank abgeholt. Über das „Füllskript" werden dynamische HTML-Seiten erstellt und im Browser angezeigt. Server-seitig werden regelmäßig (im fünf-Minuten-Takt) die Messungen von den interorganisationalen Ressourcen abgeholt. Sie werden entsprechend vorbereitet, transformiert und in die PostgreSQL Datenbank gespeichert. So wird gewährleistet, dass immer höchstens fünf Minuten alte Daten zur Verfügung stehen.

Über die unterschiedlichen Visualisierungen (Übersichtskarte, Status von Knoten und von Links, Status von E2E-Links und Dashboard) können die Operateure des LHCOPN Fehler nachvollziehen, die in der eigenen Domäne oder in der der beteiligten Partner auftreten. Einige sehr interessante Fehler-Muster sind im Anhang IX dargestellt. Eine interaktive Fehlerlösung ist in dieser prototypischen Implementierung nicht realisiert worden. Aus diesem Gesichtspunkt ist es also nur ein halbautomatisches Fehlermanagementtool: Ein Fehler wird von der betroffenen Domäne übernommen, analysiert und bearbeitet und erst nach seiner Behebung als gelöst in der Wetterkarte angezeigt. Um das ganze Spektrum an Informationen, Funktionalitäten und organisatorischen Einheiten in einer interorganisationalen Umgebung zu auszurollen, würde ein wesentlich höheres Ausmaß an Ressourcen benötigt.

Es gibt einige wichtige Schritte im Deployment der ioFMA, die immer notwendig sind. Diese werden nach Bedarf mehr oder weniger ausführlich durchgeführt, abhängig von der Umgebung in der die ioFMA eingeführt wird.

7.2. Allgemeine Deployment-Schritte

Wie schon in den vorherigen Abschnitten dieses Kapitels betont wurde hier ein Spezialfall des Deployments realisiert. Dieser, zusammen mit den in [Schif 07, Scha 08] vorgestellten Ansätzen, tragen aber dazu bei, ein bestimmtes Vorgehen beim Deployment der ioFMA festzulegen:

Analyse des bestehenden Systems Da die ioFMA immer in einem bestehenden, mehr oder weniger strukturierten interorganisationalen Umfeld einzuführen ist, wird als erstes ein genauer Einblick in dieses benötigt. Dafür ist eine detaillierte Analyse der benötigten Informationen erforderlich. Die im bestehenden Umfeld benötigten Informationen sind dann als Basis für das interorganisationale Fehlermanagement anzusehen. Desweiteren müssen in der Analysephase die bestehenden Kommunikationsstrukturen betrachtet werden, da diese größtenteils auch in der Implementierung der ioFMA weitergenutzt werden. Ebenso spielen die schon im Einsatz befindlichen Dienste und Managementwerkzeuge (z.B. Netzmonitoring, Trouble Ticket Systeme) eine wichtige Rolle.

Benutzte Technologien Eine zweite, ebenso wichtige und mit der ersten verknüpfte Phase ist die Untersuchung der aktuell im System benutzten Technologien. Mit anderen Worten muss eine Bestandsaufnahme der Plattformen, die genutzt werden (z. B. Web Services, CORBA, JE2E usw.), der Schemata und Implementierungen erstellt werden. Dabei muss bewertet werden, welche dieser Technologien mit Erfolg eingesetzt werden, welche veraltet sind und ersetzt werden müssen usw. Das dient unter der Verwendung der in Kapitel 6 festgelegte Transformationsvorschriften als Basis für die Transformation des ioFMA-PIM auf das ioFMA-PSM.

Aufbau des Informationsbausteines Auf Grundlage der Daten aus den Analysen in den ersten zwei Phasen und des Informationsmodells der ioFMA wird entschieden, welche Klassen des ioFMA-PIM in Bestandteile des PSM transformiert werden können. Insbesondere die Klassen, die die Managementobjekte repräsentieren, sollten entsprechend abgebildet werden. Dieselben Überlegungen müssen dann bezüglich der Informationsdatenbank angestellt werden. Es besteht die Möglichkeit, eine bereits existierende Datenbank zu nehmen und diese entsprechend zu erweitern, oder man wählt ein neues Produkt, das mehr Potential bietet und deshalb die Umstellung des restlichen Systems bzw. seine Anbindung an das bestehende System rechtfertigt. Das Informationsmodell wird dann im bestehenden System eingesetzt, wobei man die Granularität der Implementierung an die gewonnenen Informationen der Analyse ausrichtet.

Aufbau des Kommunikationsbausteines Die existierenden Kommunikationsprotokolle und deren aktuelle Implementierung wurden schon in den ersten zwei Phasen analysiert. Dazu wurde in der dritten Phase der Informationsbaustein definiert. Anschließend muss durch Erweiterung und Anpassung der ioFMA an die bestehende Technologie oder durch Einführung komplett neuartiger Kommunikationsprotokolle, - mechanismen und -implentierungen ein geeigneter Kommunikationsbaustein realisiert wer-

den. Dieser soll die Kommunikation zwischen den im Informationsbaustein definierten Komponenten, wie beispielsweise die Datenabholung von interorganistationalen Ressourcen und Dienste festlegen und koordinieren.

Entwicklung der Basisanwendungen Ausgehend von den Funktionalitäten, die durch die ioFMA unterstützt werden müssen oder von den Kunden angefordert wurden, werden Basisfunktionalitäten realisiert. Die Zustandsüberwachung des Systems, das Ereignis- und Topologiemanagement sind einige der wichtigen Basisanwendungen, die im Deployment der ioFMA repräsentiert werden sollten. Nichtsdestotrotz können nach Bedarf auch weitere Anwendungen realisiert werden oder die Granularität der Entwicklung den Gegebenheiten des existierenden Systems angepasst werden.

Entwicklung von Managementanwendungen Es kann eine Vielfalt von Managementanwendungen implementiert werden. Im Bereich des interorganisationalen Fehlermanagements sind die Überwachung der Datenquellen und der Daten, die daraus stammen, ebenso wie die Bereitstellung von Mechanismen, durch die die Daten entsprechend der Vorgaben des gegebenen Systems umgewandelt werden können, sehr bedeutend. Auch eine mögliche Trouble-Ticket-Administration und Service-Desk-Unterstützung sind dafür relevant. Man befindet sich hier wieder, wie bei den Basisanwendungen auch, vor der Entscheidung, welche Managementanwendungen und mit welcher Granularität diese eingeführt werden müssen, um die Gesamtanforderungen (Kunden, System, verteilte Datenquellen, beteiligte Domänen usw.) zu erfüllen.

Entwicklung der Oberflächenbausteine Nachdem die Grundkomponenten eingesetzt wurden, kann man zu dem für den Anwender wichtigsten Teil übergehen: die graphische Oberfläche. Diese muss die für den Anwender wichtigsten Bausteine abbilden und wiedergeben. Für die ioFMA sind dies die interorganisationalen Ressourcen, Dienste und Fehler. Die Granularität der Darstellung ist Aufgabe der Kundenanforderung und nicht zuletzt der Implementierung. Man kann gegebenenfalls mehrere Sichten einführen, um sicherzustellen, dass alle benötigten Informationsartefakte visualisiert werden. Bei der Realisierung dieses Teiles muss ein ständiger Kontakt zu den Kunden beibehalten werden, um deren Anforderungen und Feedback einzubauen.

Beschreibung des übergreifenden Fehlermanagementsystems Nachdem alle benötigten Komponenten existieren, wird eine übergreifende Beschreibung (Architektur) dieses neu entwickelten Systems realisiert.

7.3. Zusammenfassung

In diesem Kapitel wurde die zuvor entwickelte interorganisationale Fehlermanagementarchitektur (ioFMA) in die LHCOPN-Umgebung eingeführt. In Anlehnung an die Managementwerkzeuge des bereits existierenden perfSONAR-Systems und nach einer entsprechenden Transformation wurden der Reihe nach die unterschiedlichen Bausteine (Informations-, Kom-

munikationsbaustein, Basis- und Managementanwendungen als auch Oberflächenbausteine) für diese spezielle Umgebung erstellt. Es wurden anschließend allgemeine Schritte für das Deployment der ioFMA definiert.

Mit dieser exemplarischen Einführung als letztem Schritt des MDA-Vorgehens kann die Überprüfung der Erfüllung der Anforderungen fortgesetzt werden. Tabelle 5.11 (Seite 181) fasst die Erfüllung der Anforderungen nach der Realisierung der plattformunabhängigen ioFMA zusammen; in Abschnitt 6.5 wurde dies mit den, nach der Realisierung des ioFMA-PSM, erfüllten Anforderungen ergänzt. Das hier realisierte Deployment erfüllt durch seine Implementierung zusätzlich folgende Anforderungen an das Informationsmodell: Konvertierungsmethoden (IM-02), Standard-Metriken (IM-05), Aggregationsfunktionen (IM-06). Für das Funktionsmodell der ioFMA wird eine Visualisierung angeboten (FM-01). Auch die Reporting-bezogenen Anforderungen sind erfüllt (FM-R01 und FM-R02); FM-R03 durch Realisierung historischer Graphen teilweise. Im Bereich des Kommunikationsmodells ist KM-01 (Kommunikationsmechanismen) durch den exklusiven Einsatz von Pull-Mechanismen nur teilweise realisiert, die Kommunikationsprotokolle (KM-03) wurden der perfSONAR bzw. NMWG entliehen, sind also in diesem speziellen Fall erfüllt. Durch die hier realisierten Managementanwendungen ist die Datenintegrität (NF-02) und Datenaktualität (NF-03) gewährleistet. Für Teile des Fehlerlebenszyklus und die Dokumentation (NF-08) wurde eine Automatisierung (NF-06) realisiert.

Nachdem in den vorherigen Kapiteln Anforderungen an die ioFMA gestellt, das ioFMA-PIM entwickelt und anhand geeigneter Transformationsvorschriften in das ioFMA-PSM umgewandelt wurde, realisiert dieses Kapitel die letzte Stufe im MDA-Vorgehen, nämlich die spezifische Realisierung (Implementierung) der ioFMA in einem bestehenden System.

Das ioFMA-PIM kann bei Anwendung der in der vorliegenden Arbeit definierten Methodik in PSM-Modelle auf der Basis anderer Plattform-Technologien umgewandelt werden, was zu weiteren Implementierungen führt. Deshalb ergänzt der Implementierungsschritt das MDA-Vorgehen in natürlicher Weise.

8 Schlusswort

8.1. Zusammenfassung

Die vorliegende Arbeit gibt mit Hilfe des MDA-Ansatzes einen Überblick über das interorganisationale Fehlermanagement vom Prozess (oberste Ebene) bis zum System (unterste Ebene). Ergebnis der Arbeit ist die Konzeption einer vollständige Managementarchitektur zur Unterstützung des interorganisationalen Fehlermanagements.

Interorganisationale Umgebungen sind grundsätzlich von hoher Komplexität und Heterogenität auf der einen Seite, aber auch von Autonomie der beteiligten Organisationen auf der anderen Seite gekennzeichnet. Das macht den Entwurf eines solchen Systems zu einer Herausforderung. Es bestehen klar definierte organisatorische Konstrukte, die im interorganisationalen Umfeld eingesetzt werden. Diese wurden in der vorstehenden Arbeit als interorganisationale Formen der Diensterbringung benannt. Es kann differenziert werden zwischen hierarchischen und heterarchischen Formen der Diensterbringung. Der hierarchische Fall ist eng mit einer Baumstruktur verbunden. Darin herrscht eine vertikale Dienstkomposition vor. Die Heterarchie ist durch vielfältige Kommunikationsbeziehungen zwischen den Organisationen, die Anwesenheit einer einheitlichen Ordnung und eine horizontale Dienstkomposition charakterisiert. Die zwei Grundformen der interorganisationalen Diensterbringung finden sich in der Realität selten in Reinform, sondern in der Regel in gemischter Form oder mit stärkerer hierarchischer bzw. heterachischer Prägung.

Für diese zwei Grundformen der Diensterbringung wurden zwei Szenarien (MWN für Hierarchie und LCG für Heterarchie) im Allgemeinen und in Bezug auf den Fehlermanagementprozess analysiert. Im Fall der Hierarchie lehnt sich dieser an die Incident- und Problem-Management-Prozesse der ITIL und im Fall der Heterarchie an das Referenzmodell für Verkettete Dienste an. Basierend auf den beiden Szenarien bzw. den darin realisierten Fehlermanagementprozessen wurde ein verallgemeinertes Szenario abgeleitet. Anhand dieses Szenarios wurden die Anwendungsfälle und die darin implizierten Akteure abgeleitet und die An-

forderungsanalyse eingeleitet. Die Anwendungsfälle sind, je nach der Funktionalität, die die ioFMA unterstützen soll, in fünf größere Kategorien eingeteilt worden: Fehlerlokalisierung, Fehlerbearbeitung, Monitoring, Reporting sowie Markierung von Fehlalarmen mit insgesamt 13 Anwendungsfällen. Aus den hier beschriebenen Anwendungsfällen wurden pro Kategorie einen Reihe von funktionalen und nicht-funktionalen Anforderungen an die vier Teilmodelle der ioFMA abgeleitet und zusammengefasst. Dieser Teil der Anforderungsanalyse inklusive der Beschreibung der Szenarien und der Anwendungsfälle entspricht in der MDA-Methodik dem berechnungsunabhängigen Modell (Kapitel 3).

Der damit gebildete Anforderungskatalog wurde dem aktuellen Stand der Wissenschaft gegenüber gestellt, eine Analyse der bestehenden Ansätze durchgeführt und evaluiert, welche Teile daraus auf den übergreifenden Fall des interorganisationalen Fehlermanagements anzuwendbar sind (Kapitel 4).

Nachdem die Grundlagen durch die Anforderungsanalyse gesetzt wurden, wurde das plattformunabhängige Modell der Architektur realisiert ioFMA-PIM. Der Entwurf der ioFMA begann mit dem Organisationsmodell, indem als erstes abhängig von den Zuständigkeitsbereichen (Kunden-, Service-Provider- und interorganisationaler Bereich) die Managementdomänen definiert wurden. Diesen Bereichen wurden Rollen zugeordnet. Interaktionskanäle spielen für das interorganisationale Fehlermanagement eine sehr wichtige Rolle, weil darüber Daten, Informationen und Nachrichten ausgetauscht werden. Zusammen mit den Rollen bestimmen sie sowohl die Interdomänen-Kommunikation als auch die Managementfunktionen, nehmen also auf die Weiterentwicklung dieser Architektur (insbes. Funktions- und Kommunikationsmodell) großen Einfluss. Eine ausführliche Darstellung der Rollen und Interaktionskanäle wurde sowohl für den allgemeinen Fall, aber auch für den hierarchischen und heterarchischen Fall realisiert. Daraus konnten die Unterschiede zwischen den Organisationsmodellen der beiden interorganisationalen Formen der Diensterbringung abgeleitet werden.

Das ioFMA-Funktionsmodell mit Funktionsbereichen und Managementfunktionen folgt unmittelbar aus den in der Anforderungsanalyse beschriebenen Anwendungsfällen. Die Funktionsbereiche sind dabei ähnlich aufgebaut wie die Managementbereiche des Organisationsmodells (abhängig vom Einflussbereich des Nutzers, des lokalen Service-Providers sowie interorganisational). Diese definierten Funktionsbereiche nutzen in der Realisierung Managementfunktionen, die den Anwendungsfällen entsprechen. Zu diesen kommen noch zwei weitere übergreifende Funktionen wie Datenänderung oder Erstellung von Fehlermeldungen hinzu. Eine ausführliche Beschreibung jeder dieser Funktionen wurde anhand von Sequenzdiagrammen in UML realisiert. Dabei wurden auch Nachrichten und Funktionen beim Übergang zwischen den Rollen und Domänen eingeführt. Abschließend wurden die Unterschiede zwischen der Realisierung für die hierarchische und für die heterarchische Form der Diensterbringung aufgezeigt. Diese zeichnen sich durch eine deutlich höhere Komplexität im Fall des heterarchischen Modells aus. Bei der Realisierung ist die Komplexität (gering oder hoch) auf die Komplexität des Organisationsmodells mit dessen Rollen und Interaktionskanälen zurückzuführen.

Nachdem sowohl die Rollen, Verantwortlichkeiten und deren Interaktionen als auch die Funktionsbereiche und die von ihnen genutzten Managementfunktionen entworfen wurden, ist auf diesen basierend die Informationsbasis für die ioFMA realisiert worden. Das ioFMA-Informationsmodell wurde aus separaten Domänen entsprechend dem objektorientierten Entwurf zusammengesetzt. Diese sind thematisch orientierte Gruppen von Entitäten, die konzeptionell abgeschlossene Bereiche des interorganisationalen Fehlermanagements repräsentieren. Mit Hilfe der Modelldomänen wurde ein gemeinsames Verständnis der beteiligten Rollen bezüglich der ausgetauschten Managementinformation realisiert. Um die Plattformunabhängigkeit des Modells zu bewahren, wurde UML als Modellierungssprache eingesetzt. Dadurch wird auch die Wiederverwendbarkeit des Modells für jede Plattform gewährleistet.

Die Charakterisierung der Kommunikation zwischen den definierten Rollen im interorganisationalen Umfeld wurde im Kommunikationsmodell realisiert. Dieses gibt einen generischen Überblick über die Kommunikationsmechanismen, Kommunikationsprotokolle und Interdomänen-Kommunikation der im Organisationsmodell definierten Rollen und deren Beziehungen.

Damit wurde eine plattformunabhängige Managmentarchitektur (ioFMA) zur Unterstützung des Fehlermanagements in interorganisationalen Umgebungen realisiert. Diese ist ein allgemeingültiges Rahmenwerk, das abhängig von der Umgebung umgesetzt werden kann. Um die Tragfähigkeit dieses allgemeinen Modells zu demonstrieren, wurde entsprechend dem hier verfolgten MDA-Ansatz mit Hilfe der WSDL Beschreibungssprache eine platformspezifische Transformation des ioFMA-PIM auf ein PSM basierend auf Web-Services durchgeführt. Diese Transformation beruht auf der in [Lope 05] beschriebenen Metamodelltransformation. Basierend darauf wurde ein Satz von Transformations- und Abbildungsregeln vom ioFMA-PIM zum ioFMA-PSM erstellt.

Die hier beschriebene plattformspezifische ioFMA repräsentiert stellvertretend mögliche PSM. Durch die Anwendung der Metamodelltransformation und Etablierung klarer Transformationsregeln kann das ioFMA-PIM auf jede Plattform abgebildet werden. Diese Schritte sind unverzichtbar, um den Informationsverlust zu minimieren und die Semantik der beschriebenen Klassen (mit deren Attribute und Operationen) bei der Transformation beizubehalten.

Der letzte Schritt im MDA-Ansatz ist der Einsatz der ioFMA in einer bestimmten bereits existierenden Infrastruktur. Dies wurde anhand der LHCOPN-Infrastruktur, an deren Basis die im Géant Projekt realisierte perfSONAR Architektur steht, demonstriert. Gemäß des Managementplattformkonzepts wurden die realen Elemente der LHCOPN-Topologie und deren Messungen auf die ioFMA-PIM/PSM Klassen/Elemente abgebildet. Das Ergebnis ist die prototypische Implementierung der LHCOPN-Wetterkarte. Anhand dieses speziellen Vorgehens wurde eine allgemeine Vorgehensweise für das Deployment der ioFMA in jeder beliebiger Umgebung definiert. Das Deployment beinhaltet grundsätzlich die Analyse des bestehenden Zustands eines Systems (Infrastruktur, eingesetzte Techniken, Funktionalitäten usw.), der Bereitstellung eines Informationsbausteines basierend auf dem Informationsmodell und auf

den im Ist-Zustand vorhandenen Komponenten. Zusammen mit dem auf dem Kommunikationsmodell basierenden Kommunikationsbaustein realisieren diese den Kern der eingesetzten Architektur. Es folgt dann die Entwicklung von Basis- und Managementanwendungen, die stark an das Funktionsmodell angelehnt sind. Eine Zusammenfassung des so realisierten Fehlermanagementsystem schließt das Deployment ab.

Zusammenfassend liefert diese Arbeit eine allgemeine Managementarchitektur zur Unterstützung des Fehlermanagements (ioFMA-PIM) im interorganisationalen Umfeld . Unterschiede zwischen den Diensterbingungsformen wurden aufgezeigt. Eine Transformation der allgemeinen Architektur wurde exemplarisch auf die Web-Services Plattform mit dem Ergebnis eines ioFMA-PSM durchgeführt. Dem MDA-Ansatz folgend wurde diese im LHCOPN eingeführt und die entsprechenden allgemeinen Einführungsschritte definiert.

8.2. Ausblick

Auch wenn diese Arbeit alle für das interorganisationale Fehlermanagement wichtigen Punkte anspricht, wurden einige weniger ausführlich als andere betrachtet. Diese können aber auf Grund der hier bereitgestellten allgemeinen Modellierung, der exemplarischen Transformation, der prototypischen Implementierung und insbesondere der darin realisierten Methodik weitergeführt werden.

Ein solcher Punkt könnte die Realisierung mehrerer ioFMA-PSM unter Einsatz weiterer Plattformen und der Vergleich der Effizienz, Skalierbarkeit und Automatisierung der restlichen Plattformen sein. Dabei können nicht nur die PSMs verglichen werden, sondern auch die darunterliegenden Transformationen.

Ein anderer weiterführender Punkt ist die Realisierung ausführlicher Deployments für hierarchische und heterarchische Szenarien. Diese können dann in Bezug auf die Transformation und die gewählten Plattformen gegenübergestellt werden.

Die Anbindung dieses Fehlermanagement-Frameworks an andere interoganisationale Managementplattformen (z. B. HPOpenView, IBM WebSphere, IBM Netcool), aber auch die Erweiterung hinsichtlich eines Konfigurationsmanagements, Änderungsmanagements und Leistungsmanagements ist ein wichtiger Schritt, um die Umsetzbarkeit der ioFMA in bestehenden Umgebungen zu erleichtern.

Weiterhin ist die Koppelung an Trouble-Ticket-Systeme (TTS) ein äußerst interessantes Themengebiet. Vorstellbar ist z.B., dass ein interorganisational eingesetztes TTS an die lokalen, domänenspezifischen TTS angebunden wird, aber auch, dass direkte Anbindungen zwischen den unterschiedlichen TTS der beteiligten Domänen realisiert werden. Dies wäre ein wichtiger Meilenstein auf dem Weg zu einem nahtlosen domänenübergreifenden interorganisationalen Fehlermanagementprozess.

Anhänge

I. WSDL-Modell für das Package IOFMATopLevel

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<wsdl:definitions name="IOFMATopLevel"
  targetNamespace="urn://IOFMATopLevel.wsdl"
  xmlns:tns="urn://IOFMATopLevel.wsdl"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
  xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:types>
    <xs:schema targetNamespace="urn://IOFMATopLevel.wsdl"
      xmlns:xs="http://www.w3.org/2001/XMLSchema"
      xmlns:s1="urn://IOFMATopLevel.wsdl">
      <xs:element name="ioFMAEntity" type="s1:ioFMAEntity"/>
      <xs:complexType name="ioFMAEntity">
        <xs:complexContent>
          <xs:extension base="s1:ioFMARootEntity">
            <xs:sequence/>
          </xs:extension>
        </xs:complexContent>
      </xs:complexType>
      <xs:element name="ioFMAManagedEntity" type="s1:ioFMAManagedEntity"/>
      <xs:complexType name="ioFMAManagedEntity">
        <xs:complexContent>
          <xs:extension base="s1:ioFMAEntity">
            <xs:sequence/>
          </xs:extension>
        </xs:complexContent>
      </xs:complexType>
      <xs:element name="ioFMARootEntity" type="s1:ioFMARootEntity"/>
      <xs:complexType name="ioFMARootEntity">
        <xs:sequence>
          <xs:element name="description" type="xs:string" minOccurs="1" maxOccurs="1"/>
          <xs:element name="entityname" type="xs:string" minOccurs="1" maxOccurs="1"/>
          <xs:element name="objectid" type="xs:string" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
```

```
      </xs:complexType>
      <xs:element name="ioFMAUnManagedEntity" type="s1:ioFMAUnManagedEntity"/>
      <xs:complexType name="ioFMAUnManagedEntity">
        <xs:complexContent>
          <xs:extension base="s1:ioFMAEntity">
            <xs:sequence/>
          </xs:extension>
        </xs:complexContent>
      </xs:complexType>
    </xs:schema>
  </wsdl:types>
  <wsdl:message name="getEntityNameRequest">
    <wsdl:part name="term" type="xs:string"/></wsdl:message>
  <wsdl:message name="getEntityNameResponse">
    <wsdl:part name="value" type="xs:string"/></wsdl:message>
  <wsdl:message name="getObjectIDRequest">
    <wsdl:part name="term" type="xs:string"/></wsdl:message>
  <wsdl:message name="getDescriptionRequest">
    <wsdl:part name="term" type="xs:string"/></wsdl:message>
  <wsdl:message name="getDescriptionResponse">
    <wsdl:part name="value" type="xs:string"/></wsdl:message>
  <wsdl:message name="getObjectIDResponse">
    <wsdl:part name="value" type="xs:ID"/></wsdl:message>
  <wsdl:message name="setDescriptionRequest">
    <wsdl:part name="term" type="xs:string"/> </wsdl:message>
  <wsdl:message name="setEntityNameRequest">
    <wsdl:part name="name" type="xs:string"/> </wsdl:message>
  <wsdl:message name="setObjectIDRequest">
    <wsdl:part name="identifier" type="xs:ID"/></wsdl:message>
  <wsdl:message name="setIORequest">
    <wsdl:part name="IO" type="xs:boolean"/></wsdl:message>
  <wsdl:message name="setLocalRequest">
    <wsdl:part name="IO" type="xs:boolean"/></wsdl:message>
  <wsdl:portType name="ioFMAManagedEntity">
    <wsdl:operation name="getEntityName">
      <wsdl:input name="Request1" message="tns:getEntityNameRequest"/>
      <wsdl:output name="Response1" message="tns:getEntityNameResponse"/>
    </wsdl:operation>
    <wsdl:operation name="getDescription">
      <wsdl:input name="Request2" message="tns:getDescriptionRequest"/>
      <wsdl:output name="Response2" message="tns:getDescriptionResponse"/>
    </wsdl:operation>
    <wsdl:operation name="setDescription">
      <wsdl:input name="OneWay1" message="tns:setDescriptionRequest"/>
    </wsdl:operation>
    <wsdl:operation name="setEntityName">
      <wsdl:input name="OneWay2" message="tns:setEntityNameRequest"/>
    </wsdl:operation>
    <wsdl:operation name="setObjectID">
      <wsdl:input name="OneWay3" message="tns:setObjectIDRequest"/>
    </wsdl:operation>
    <wsdl:operation name="getObjectID">
      <wsdl:input name="Request3" message="tns:getObjectIDRequest"/>
      <wsdl:output name="Response3" message="tns:getObjectIDResponse"/>
    </wsdl:operation>
    <wsdl:operation name="setIO">
      <wsdl:input name="OneWay4" message="tns:setIORequest"/>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:portType name="ioFMAUnManagedEntity">
    <wsdl:operation name="getObjectID">
```

```
      <wsdl:input name="getObjectIDRequest" message="tns:getObjectIDRequest"/>
      <wsdl:output name="getObjectIDResponse" message="tns:getObjectIDResponse"/>
    </wsdl:operation>
    <wsdl:operation name="getDescription">
      <wsdl:input name="Request2" message="tns:getDescriptionRequest"/>
      <wsdl:output name="Response2" message="tns:getDescriptionResponse"/>
    </wsdl:operation>
    <wsdl:operation name="getEntityName">
      <wsdl:input name="getEntityNameRequest" message="tns:getEntityNameRequest"/>
      <wsdl:output name="getEntityNameResponse" message="tns:getEntityNameResponse"/>
    </wsdl:operation>
    <wsdl:operation name="setDescription">
      <wsdl:input name="setDescription" message="tns:setDescriptionRequest"/>
    </wsdl:operation>
    <wsdl:operation name="setEntityName">
      <wsdl:input name="setEntity" message="tns:setEntityNameRequest"/>
    </wsdl:operation>
    <wsdl:operation name="setObjectID">
      <wsdl:input name="OneWay3" message="tns:setObjectIDRequest"/>
    </wsdl:operation>
    <wsdl:operation name="setIO">
      <wsdl:input name="setIO" message="tns:setLocalRequest"/>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="ioFMAManagedEntity" type="tns:ioFMAManagedEntity">
   ...
  </wsdl:binding>
  <wsdl:binding name="ioFMAUnManagedEntity" type="tns:ioFMAUnManagedEntity">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    ...
  </wsdl:binding>
  <wsdl:service name="ioFMAEntity">
    <wsdl:port name="ioFMAManagedEntity" binding="tns:ioFMAManagedEntity">
      <soap:address location="urn://IOFMATopLevel/ioFMAManagedEntity"/>
    </wsdl:port>
    <wsdl:port name="ioFMAUnManagedEntity" binding="tns:ioFMAUnManagedEntity">
      <soap:address location="urn://IOFMATopLevel/ioFMAUnManagedEntity"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

II. WSDL-Modell für das Package `Fault`

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<wsdl:definitions name="Fault"
  targetNamespace="urn://Fault.wsdl"
  xmlns:tns="urn://Fault.wsdl"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
  xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:import namespace="urn://Service.wsdl"/>
  <wsdl:import namespace="urn://IOFMASpecification.wsdl"/>
  <wsdl:import namespace="urn://IOFMARole.wsdl"/>
  <wsdl:types>
    <xs:schema targetNamespace="urn://Fault.wsdl"
      xmlns:xs="http://www.w3.org/2001/XMLSchema"
      xmlns:s1="urn://IOFMATopLevel.wsdl">
      <xs:import namespace="urn://Service.wsdl"/>
      <xs:import namespace="urn://Resource.wsdl"/>
      <xs:import namespace="urn://IOFMATopLevel.wsdl"/>
      <xs:import namespace="urn://IOFMASpecification.wsdl"/>
      <xs:import namespace="urn://IOFManagement"/>
      <xs:element name="Fault" type="Fault"/>
      <xs:complexType name="Fault">
        <xs:complexContent>
          <xs:extension base="s1:ioFMAEntity">
            <xs:sequence>
              <xs:element name="description" type="xs:string" minOccurs="1" maxOccurs="1"/>
              <xs:element name="faultcategory" type="xs:int" minOccurs="1" maxOccurs="1"/>
              <xs:element name="faultid" type="xs:string" minOccurs="1" maxOccurs="1"/>
              <xs:element name="faultresstate" type="xs:string" minOccurs="1"
                maxOccurs="1"/>
              <xs:element name="firstalert" type="xs:string" minOccurs="1" maxOccurs="1"/>
              <xs:element name="io" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
              <xs:element name="originatingsystem" type="xs:string" minOccurs="1"
                maxOccurs="1"/>
              <xs:element name="priority" type="xs:string" minOccurs="1" maxOccurs="1"/>
              <xs:element name="timechanged" type="xs:string" minOccurs="1" maxOccurs="1"/>
              <xs:element name="timeraised" type="xs:string" minOccurs="1" maxOccurs="1"/>
            </xs:sequence>
          </xs:extension>
        </xs:complexContent>
      </xs:complexType>
      <xs:element name="FaultInformation" type="FaultInformation"/>
      <xs:complexType name="FaultInformation">
        <xs:complexContent>
          <xs:extension base="ManagementInformation">
            <xs:sequence>
              <xs:element name="Fault" type="Fault" minOccurs="1" maxOccurs="1"/>
              <xs:element name="FaultSpecification" type="FaultSpecification"
                minOccurs="1" maxOccurs="1"/>
            </xs:sequence>
          </xs:extension>
        </xs:complexContent>
      </xs:complexType>
      <xs:element name="ioFault" type="ioFault"/>
      <xs:complexType name="ioFault">
        <xs:sequence>
          <xs:element name="description" type="xs:string" minOccurs="1" maxOccurs="1"/>
```

```
        <xs:element name="faultcategory" type="xs:int" minOccurs="1" maxOccurs="1"/>
        <xs:element name="faultid" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="faultresstate" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="firstalert" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="io" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
        <xs:element name="originatingsystem" type="xs:string" minOccurs="1"
          maxOccurs="1"/>
        <xs:element name="priority" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="timechanged" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="timeraised" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="serviceid" type="xs:int" minOccurs="1" maxOccurs="1"/>
        <xs:element name="ServiceInformation" type="ServiceInformation" minOccurs="1"
          maxOccurs="1"/>
      </xs:sequence>
    </xs:complexType>
    <xs:element name="localFault" type="localFault"/>
    <xs:complexType name="localFault">
      <xs:sequence>
        <xs:element name="description" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="faultcategory" type="xs:int" minOccurs="1" maxOccurs="1"/>
        <xs:element name="faultid" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="faultresstate" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="firstalert" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="io" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
        <xs:element name="originatingsystem" type="xs:string" minOccurs="1"
          maxOccurs="1"/>
        <xs:element name="priority" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="timechanged" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="timeraised" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="domid" type="xs:int" minOccurs="1" maxOccurs="1"/>
        <xs:element name="ResourceInformation" type="ResourceInformation" minOccurs="1"
          maxOccurs="1"/>
      </xs:sequence>
    </xs:complexType>
  </xs:schema>
</wsdl:types>
<wsdl:message name="setCategoryRequest">
  <wsdl:part name="value" type="Fault.faultCategory"/></wsdl:message>
<wsdl:message name="setDescriptionRequest">
  <wsdl:part name="value" type="Fault.description"/></wsdl:message>
<wsdl:message name="setOriginatingSystem">
  <wsdl:part name="value" type="Fault.originatingSystem"/></wsdl:message>
<wsdl:message name="setPriorityRequest">
  <wsdl:part name="value" type="Fault.priority"/></wsdl:message>
<wsdl:message name="setTimeRaised">
  <wsdl:part name="value" type="Fault.timeRaised"/></wsdl:message>
<wsdl:message name="getDomainsAffectedRequest">
  <wsdl:part name="term" type="Fault.faultID"/></wsdl:message>
<wsdl:message name="getDomainsAffectedResponse">
  <wsdl:part name="value" type="IOFMADomain"/></wsdl:message>
<wsdl:message name="getFaultCategoryRequest">
  <wsdl:part name="term" type="Fault.faultID"/></wsdl:message>
<wsdl:message name="getFaultCategoryResponse">
 <wsdl:part name="value" type="Fault.faultCategory"/></wsdl:message>
<wsdl:message name="getFaultIDRequest">
  <wsdl:part name="term" type="Fault.faultID"/></wsdl:message>
<wsdl:message name="getFaultIDResponse">
  <wsdl:part name="value" type="Fault.faultId"/></wsdl:message>
<wsdl:message name="getFaultPriorityRequest">
  <wsdl:part name="term" type="Fault.faultID"/></wsdl:message>
<wsdl:message name="getFaultPriorityResponse">
```

```
    <wsdl:part name="value" type="Fault.priority"/></wsdl:message>
  <wsdl:message name="getFaultResStateRequest">
    <wsdl:part name="term" type="Fault.faultID"/></wsdl:message>
  <wsdl:message name="getFaultResStateResponse">
    <wsdl:part name="value" type="Fault.fautResState"/></wsdl:message>
  <wsdl:message name="setFaultResStateRequest">
    <wsdl:part name="value" type="Fault.fautResState"/></wsdl:message>
  <wsdl:message name="getOriginSystemRequest">
    <wsdl:part name="term" type="Fault.faultID"/></wsdl:message>
  <wsdl:message name="getOriginSystemResponse">
    <wsdl:part name="value" type="Fault.originatingSystem"/></wsdl:message>
  <wsdl:message name="getTimeRaisedRequest">
    <wsdl:part name="term" type="Fault.faultID"/></wsdl:message>
  <wsdl:message name="getTimeRaisedResponse">
    <wsdl:part name="value" type="Fault.timeRaised"/></wsdl:message>
  <wsdl:message name="setTimeChangedRequest">
    <wsdl:part name="value" type="Fault.timeChanged"/></wsdl:message>
  <wsdl:message name="setProcessingTerminal">
    <wsdl:part name="value" type="IOFMARole.roleID"/></wsdl:message>
  <wsdl:message name="setFaultSpecificationRequest">
    <wsdl:part name="value" type="FaultSpecification"/></wsdl:message>
  <wsdl:message name="setServicesAffectedRequest">
    <wsdl:part name="value" type="Service"/></wsdl:message>
  <wsdl:message name="setDomainsImpliedRequest">
    <wsdl:part name="value" type="IOFMADomain"/></wsdl:message>
  <wsdl:message name="getServicesAffectedRequest">
    <wsdl:part name="term" type="Fault.faultID"/></wsdl:message>
  <wsdl:message name="getServicesAffectedResponse">
    <wsdl:part name="value" type="Service"/></wsdl:message>
  <wsdl:portType name="Fault">
    <wsdl:operation name="setCategory">
      <wsdl:input name="OneWay1" message="tns:setCategoryRequest"/>
    </wsdl:operation>
    <wsdl:operation name="setDescription">
      <wsdl:input name="OneWay2" message="tns:setDescriptionRequest"/>
    </wsdl:operation>
    <wsdl:operation name="setOriginatingSystem">
      <wsdl:input name="OneWay3" message="tns:setOriginatingSystem"/>
    </wsdl:operation>
    <wsdl:operation name="setPriority">
      <wsdl:input name="OneWay4" message="tns:setPriorityRequest"/>
    </wsdl:operation>
    <wsdl:operation name="setTimeRaised">
      <wsdl:input name="OneWay5" message="tns:setTimeRaised"/>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:portType name="ioFault">
    <wsdl:operation name="setPriority">
      <wsdl:input name="OneWay4" message="tns:setPriorityRequest"/>
    </wsdl:operation>
    <wsdl:operation name="setCategory">
      <wsdl:input name="OneWay1" message="tns:setCategoryRequest"/>
    </wsdl:operation>
    <wsdl:operation name="setDescription">
      <wsdl:input name="OneWay2" message="tns:setDescriptionRequest"/>
    </wsdl:operation>
    <wsdl:operation name="setOriginatingSystem">
      <wsdl:input name="OneWay3" message="tns:setOriginatingSystem"/>
    </wsdl:operation>
    <wsdl:operation name="setTimeRaised">
      <wsdl:input name="OneWay5" message="tns:setTimeRaised"/>
```

```
    </wsdl:operation>
    <wsdl:operation name="setDomainsImlied">
      <wsdl:input name="OneWay6" message="tns:setDomainsImpliedRequest"/>
    </wsdl:operation>
    <wsdl:operation name="setServicesAffected">
      <wsdl:input name="OneWay7" message="tns:setServicesAffectedRequest"/>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:portType name="FaultInformation">
    <wsdl:operation name="getDomainsAffected">
      <wsdl:input name="Request1" message="tns:getDomainsAffectedRequest"/>
      <wsdl:output name="Response1" message="tns:getDomainsAffectedResponse"/>
    </wsdl:operation>
    <wsdl:operation name="getFaultCategory">
      <wsdl:input name="Request2" message="tns:getFaultCategoryRequest"/>
      <wsdl:output name="Response2" message="tns:getFaultCategoryResponse"/>
    </wsdl:operation>
    <wsdl:operation name="getFaultID">
      <wsdl:input name="Request3" message="tns:getFaultIDRequest"/>
      <wsdl:output name="Response3" message="tns:getFaultIDResponse"/>
    </wsdl:operation>
    <wsdl:operation name="getFaultPriority">
      <wsdl:input name="Request4" message="tns:getFaultPriorityRequest"/>
      <wsdl:output name="Response4" message="tns:getFaultPriorityResponse"/>
    </wsdl:operation>
    <wsdl:operation name="getFaultResState">
      <wsdl:input name="Request5" message="tns:getFaultResStateRequest"/>
      <wsdl:output name="Response5" message="tns:getFaultResStateResponse"/>
    </wsdl:operation>
    <wsdl:operation name="getOriginSystem">
      <wsdl:input name="Request6" message="tns:getOriginSystemRequest"/>
      <wsdl:output name="Response6" message="tns:getOriginSystemResponse"/>
    </wsdl:operation>
    <wsdl:operation name="getServicesAffected">
      <wsdl:input name="Request7" message="tns:getServicesAffectedRequest"/>
      <wsdl:output name="Response7" message="tns:getServicesAffectedResponse"/>
    </wsdl:operation>
    <wsdl:operation name="getTimeRaised">
      <wsdl:input name="Request8" message="tns:getTimeRaisedRequest"/>
      <wsdl:output name="Response8" message="tns:getTimeRaisedResponse"/>
    </wsdl:operation>
    <wsdl:operation name="setFaultResState">
      <wsdl:input name="OneWay1" message="tns:setFaultResStateRequest"/>
    </wsdl:operation>
    <wsdl:operation name="setProcessingTerminal">
      <wsdl:input name="OneWay2" message="tns:setProcessingTerminal"/>
    </wsdl:operation>
    <wsdl:operation name="setTimeChanged">
      <wsdl:input name="OneWay3" message="tns:setTimeChangedRequest"/>
    </wsdl:operation>
    <wsdl:operation name="compare">
      <wsdl:input name="OneWay4" message="tns:setFaultSpecificationRequest"/>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:portType name="describe">
    <wsdl:operation name="descDomainsAffected">
      <wsdl:output name="Solicit1" message="tns:getDomainsAffectedRequest"/>
      <wsdl:input name="Request1" message="tns:setDomainsImpliedRequest"/>
    </wsdl:operation>
    <wsdl:operation name="descrServicesAffected">
      <wsdl:output name="Solicit2" message="tns:getServicesAffectedRequest"/>
```

```
      <wsdl:input name="Request3" message="tns:setServicesAffectedRequest"/>
    </wsdl:operation>
    <wsdl:operation name="descrCategory">
      <wsdl:output name="Solicit3" message="tns:getFaultCategoryRequest"/>
      <wsdl:input name="Request4" message="tns:setCategoryRequest"/>
    </wsdl:operation>
    <wsdl:operation name="descrOriginSystem">
      <wsdl:output name="Solicit4" message="tns:getOriginSystemRequest"/>
      <wsdl:input name="Request5" message="tns:setOriginatingSystem"/>
    </wsdl:operation>
    <wsdl:operation name="descrPriority">
      <wsdl:output name="Solicit5" message="tns:getFaultResStateRequest"/>
      <wsdl:input name="Request6" message="tns:setPriorityRequest"/>
    </wsdl:operation>
    <wsdl:operation name="descrFaultResState">
      <wsdl:output name="Notification1" message="tns:setFaultResStateRequest"/>
    </wsdl:operation>
    <wsdl:operation name="descrTimeChanged">
      <wsdl:output name="Notification2" message="tns:setTimeChangedRequest"/>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:portType name="localFault">
    <wsdl:operation name="setOriginatingSystem">
      <wsdl:input name="OneWay3" message="tns:setOriginatingSystem"/>
    </wsdl:operation>
    <wsdl:operation name="setCategory">
      <wsdl:input name="OneWay1" message="tns:setCategoryRequest"/>
    </wsdl:operation>
    <wsdl:operation name="setDescription">
      <wsdl:input name="OneWay2" message="tns:setDescriptionRequest"/>
    </wsdl:operation>
    <wsdl:operation name="setPriority">
      <wsdl:input name="OneWay4" message="tns:setPriorityRequest"/>
    </wsdl:operation>
    <wsdl:operation name="setTimeRaised">
      <wsdl:input name="OneWay5" message="tns:setTimeRaised"/>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="ioFault" type="tns:ioFault">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    ...
  </wsdl:binding>
  <wsdl:binding name="Fault" type="tns:Fault">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    ...
  </wsdl:binding>
  <wsdl:binding name="localFault" type="tns:localFault">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    ...
  </wsdl:binding>
  <wsdl:binding name="FaultInformation" type="tns:FaultInformation">
    ...
  </wsdl:binding>
  <wsdl:binding name="FaultDescribe" type="tns:describe">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    ...
  </wsdl:binding>
  <wsdl:service name="FaultInformation">
    <wsdl:port name="FaultInformation" binding="tns:FaultInformation">
      <soap:address location="urn://Fault/FaultInformation"/>
    </wsdl:port>
```

```
    <wsdl:port name="FaultDescribe" binding="tns:FaultDescribe">
      <soap:address location="urn://Fault/FaultDescribe"/>
    </wsdl:port>
  </wsdl:service>
  <wsdl:service name="Fault">
    <wsdl:port name="Fault" binding="tns:Fault">
      <soap:address location="urn://Fault"/>
    </wsdl:port>
    <wsdl:port name="ioFault" binding="tns:ioFault">
      <soap:address location="urn://Fault/ioFault"/>
    </wsdl:port>
    <wsdl:port name="localFault" binding="tns:localFault">
      <soap:address location="urn://Fault/localFault"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

III. WSDL-Modell für das Package `Service`

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<wsdl:definitions name="IOFMAService"
  targetNamespace="urn://Service.wsdl"
  xmlns:tns="urn:/Service.wsdl"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
  xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:import namespace="urn://IOFMASpecification.wsdl"/>
  <wsdl:import namespace="urn://Resource.wsdl"/>
  <wsdl:types>
    <xs:schema targetNamespace="urn://Service.wsdl"
      xmlns:xs="http://www.w3.org/2001/XMLSchema"
      xmlns:s1="urn://IOFMATopLevel.wsdl">
      <xs:import namespace=""/>
      <xs:import namespace="urn://Resource.wsdl"/>
      <xs:import namespace="urn://IOFMATopLevel.wsdl"/>
      <xs:import namespace="urn://Fault.wsdl"/>
      <xs:import namespace="urn://IOFManagement"/>
      <xs:element name="ioService" type="ioService"/>
      <xs:complexType name="ioService">
        <xs:sequence>
          <xs:element name="io" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
          <xs:element name="servdescription" type="xs:string" minOccurs="1" maxOccurs="1"/>
          <xs:element name="serviceid" type="xs:int" minOccurs="1" maxOccurs="1"/>
          <xs:element name="Resource" type="Resource" minOccurs="1" maxOccurs="unbounded"/>
          <xs:element name="domainsimplied" type="IOFMADomain" minOccurs="1"
            maxOccurs="unbounded"/>
          <xs:element name="iodelivery" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
      </xs:complexType>
      <xs:element name="localService" type="localService"/>
      <xs:complexType name="localService">
        <xs:sequence>
          <xs:element name="io" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
          <xs:element name="servdescription" type="xs:string" minOccurs="1" maxOccurs="1"/>
          <xs:element name="serviceid" type="xs:int" minOccurs="1" maxOccurs="1"/>
          <xs:element name="Resource" type="Resource" minOccurs="1" maxOccurs="unbounded"/>
          <xs:element name="domainid" type="xs:int" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
      </xs:complexType>
      <xs:element name="Service" type="Service"/>
      <xs:complexType name="Service">
        <xs:complexContent>
          <xs:extension base="s1:ioFMAEntity">
            <xs:sequence>
              <xs:element name="io" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
              <xs:element name="servdescription" type="xs:string" minOccurs="1"
                maxOccurs="1"/>
              <xs:element name="serviceid" type="xs:int" minOccurs="1" maxOccurs="1"/>
              <xs:element name="Resource" type="Resource" minOccurs="1"
                maxOccurs="unbounded"/>
            </xs:sequence>
          </xs:extension>
        </xs:complexContent>
      </xs:complexType>
      <xs:element name="ServiceFailure" type="ServiceFailure"/>
```

```
    <xs:complexType name="ServiceFailure">
      <xs:sequence>
        <xs:element name="servfailureid" type="xs:int" minOccurs="1" maxOccurs="1"/>
        <xs:element name="serviceaffected" type="xs:int" minOccurs="1" maxOccurs="unbounded"/>
        <xs:element name="Service" type="Service" minOccurs="1" maxOccurs="1"/>
      </xs:sequence>
    </xs:complexType>
    <xs:element name="ServiceInformation" type="ServiceInformation"/>
    <xs:complexType name="ServiceInformation">
      <xs:complexContent>
        <xs:extension base="ManagementInformation">
          <xs:sequence>
            <xs:element name="metricname" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="metricvalue" type="xs:long" minOccurs="1" maxOccurs="1"/>
            <xs:element name="localFault" type="localFault" minOccurs="1" maxOccurs="1"/>
            <xs:element name="Service" type="Service" minOccurs="0"
              maxOccurs="unbounded"/>
            <xs:element name="ServiceFailure" type="ServiceFailure" minOccurs="1"
              maxOccurs="1"/>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:element name="ServStateInformation" type="ServStateInformation"/>
    <xs:complexType name="ServStateInformation">
      <xs:complexContent>
        <xs:extension base="ServiceInformation">
          <xs:sequence>
            <xs:element name="status" type="xs:string" minOccurs="1" maxOccurs="1"/>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:element name="ServStatisticalInfo" type="ServStatisticalInfo"/>
    <xs:complexType name="ServStatisticalInfo">
      <xs:complexContent>
        <xs:extension base="ServiceInformation">
          <xs:sequence/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:schema>
</wsdl:types>
<wsdl:message name="getFailureRequest">
  <wsdl:part name="term" type="Service.serviceID"/></wsdl:message>
<wsdl:message name="getFailureResponse">
  <wsdl:part name="value" type="ServiceFailure.servFailureID"/></wsdl:message>
<wsdl:message name="getResourceRequest">
  <wsdl:part name="term" type="Service.serviceID"/></wsdl:message>
<wsdl:message name="getResourceResponse">
  <wsdl:part name="value" type="Resource"/></wsdl:message>
<wsdl:message name="getServInformationRequest">
  <wsdl:part name="term" type="Service.serviceID"/></wsdl:message>
<wsdl:message name="getServInformationResponse">
  <wsdl:part name="value" type="ServiceInformation"/></wsdl:message>
<wsdl:message name="setFailureRequest">
  <wsdl:part name="term" type="ServiceFailure"/></wsdl:message>
<wsdl:message name="getStateInformationRequest">
  <wsdl:part name="term" type="Service"/>
  <wsdl:part name="term2" type="ServStateInformation"/></wsdl:message>
<wsdl:message name="getStateInformationResponse">
```

```
    <wsdl:part name="value" type="ServStateInformation"/></wsdl:message>
  <wsdl:message name="getStatisticalInfoRequest">
    <wsdl:part name="term" type="Service"/>
    <wsdl:part name="term2" type="ServStatisticalInfo"/></wsdl:message>
  <wsdl:message name="getStatisticalInfoResponse">
    <wsdl:part name="value" type="ServStatisticalInfo"/></wsdl:message>
  <wsdl:message name="setServInformationRequest">
    <wsdl:part name="value" type="ServStatisticalInfo"/>
    <wsdl:part name="value" type="ServStateInformation"/></wsdl:message>
  <wsdl:message name="setStatisticalInfoRequest">
    <wsdl:part name="value" type="ServStatisticalInfo"/></wsdl:message>
  <wsdl:message name="getServSpecificationRequest">
    <wsdl:part name="term" type="Service.serviceID"/></wsdl:message>
  <wsdl:message name="getServSpecificationResponse">
    <wsdl:part name="value" type="ServSpecification.specID"/></wsdl:message>
  <wsdl:message name="getLocalServiceRequest">
    <wsdl:part name="term" type="ioService"/></wsdl:message>
  <wsdl:message name="setIoServiceResponse">
    <wsdl:part name="value" type="localService"/></wsdl:message>
  <wsdl:portType name="ioService">
    <wsdl:operation name="getFailure">
      <wsdl:input name="Request1" message="getFailureRequest"/>
      <wsdl:output name="Response1" message="getFailureResponse"/>
    </wsdl:operation>
    <wsdl:operation name="getResource">
      <wsdl:input name="Request2" message="getResourceRequest"/>
      <wsdl:output name="Response2" message="getResourceResponse"/>
    </wsdl:operation>
    <wsdl:operation name="getServInformation">
      <wsdl:input name="Request3" message="getServInformationRequest"/>
      <wsdl:output name="Response3" message="getServInformationResponse"/>
    </wsdl:operation>
    <wsdl:operation name="getServSpecification">
      <wsdl:input name="Request4" message="getServSpecificationRequest"/>
      <wsdl:output name="Response4" message="getServSpecificationResponse"/>
    </wsdl:operation>
    <wsdl:operation name="dependsOn">
      <wsdl:output name="Solicit1" message="getLocalServiceRequest"/>
      <wsdl:input name="Request1" message="setIoServiceResponse"/>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:portType name="localService">
    <wsdl:operation name="getServInformation">
      <wsdl:input name="Request3" message="getServInformationRequest"/>
      <wsdl:output name="Response3" message="getServInformationResponse"/>
    </wsdl:operation>
    <wsdl:operation name="getFailure">
      <wsdl:input name="Request1" message="getFailureRequest"/>
      <wsdl:output name="Response1" message="getFailureResponse"/>
    </wsdl:operation>
    <wsdl:operation name="getResource">
      <wsdl:input name="Request2" message="getResourceRequest"/>
      <wsdl:output name="Response2" message="getResourceResponse"/>
    </wsdl:operation>
    <wsdl:operation name="getServSpecification">
      <wsdl:input name="Request4" message="getServSpecificationRequest"/>
      <wsdl:output name="Response4" message="getServSpecificationResponse"/>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:portType name="ServiceInformation">
    <wsdl:operation name="getStateInformation">
```

```
      <wsdl:input name="Request1" message="getStateInformationRequest"/>
      <wsdl:output name="Response1" message="getStateInformationResponse"/>
    </wsdl:operation>
    <wsdl:operation name="getStatisticalInfo">
      <wsdl:input name="Request2" message="getStatisticalInfoRequest"/>
      <wsdl:output name="Response2" message="getStatisticalInfoResponse"/>
    </wsdl:operation>
    <wsdl:operation name="setServInformation">
      <wsdl:input name="OneWay1" message="setServInformationRequest"/>
    </wsdl:operation>
    <wsdl:operation name="caracterizes">
      <wsdl:output name="Solicit1" message="getServInformationRequest"/>
      <wsdl:input name="Request1" message="setServInformationRequest"/>
    </wsdl:operation>
    <wsdl:operation name="enables">
      <wsdl:output name="Solicit2" message="setServInformationRequest"/>
      <wsdl:input name="Request3" message="setFailureRequest"/>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:portType name="ServStateInformation">
    <wsdl:operation name="getStateInformation">
      <wsdl:input name="Request1" message="getStateInformationRequest"/>
      <wsdl:output name="Response1" message="getStateInformationResponse"/>
    </wsdl:operation>
    <wsdl:operation name="setStateInformation">
      <wsdl:input name="OneWay1" message="setServInformationRequest"/>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:portType name="ServStatisticalInfo">
    <wsdl:operation name="getStatisticalInfo">
      <wsdl:input name="Request2" message="getStatisticalInfoRequest"/>
      <wsdl:output name="Response2" message="getStatisticalInfoResponse"/>
    </wsdl:operation>
    <wsdl:operation name="setStatisticalInfo">
      <wsdl:input name="OneWay1" message="setStatisticalInfoRequest"/>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:portType name="ServiceFailure">
    <wsdl:operation name="inducesFault">
      <wsdl:output name="Notification1" message="setFailureRequest"/>
    </wsdl:operation>
    <wsdl:operation name="setFailure">
      <wsdl:input name="OneWay1" message="setFailureRequest"/>
    </wsdl:operation>
    <wsdl:operation name="afects">
      <wsdl:output name="Solicit1" message="getFailureRequest"/>
      <wsdl:input name="Request1" message="setFailureRequest"/>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="localService" type="localService">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    ...
  </wsdl:binding>
  <wsdl:binding name="ioService" type="ioService">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    ...
  </wsdl:binding>
  <wsdl:binding name="ServiceInformation" type="ServiceInformation">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
   ...
  </wsdl:binding>
```

```
  <wsdl:binding name="ServStateInformation" type="ServStateInformation">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    ...
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:binding name="ServStatisticalInfo" type="ServStatisticalInfo">
    ...
  </wsdl:binding>
  <wsdl:binding name="ServiceFailure" type="ServiceFailure">
   ...
  </wsdl:binding>
  <wsdl:service name="Service">
    <wsdl:port name="ioService" binding="ioService">
      <soap:address location="urn://Service/ioService"/>
    </wsdl:port>
    <wsdl:port name="localService" binding="localService">
      <soap:address location="urn://Service/localService"/>
    </wsdl:port>
  </wsdl:service>
  <wsdl:service name="ServiceInformation">
    <wsdl:port name="ServiceInformation" binding="ServiceInformation">
      <soap:address location="urn://Service/ServiceInformation"/>
    </wsdl:port>
    <wsdl:port name="ServStateInformation" binding="ServStateInformation">
      <soap:address location="urn://Service/ServStateInformation"/>
    </wsdl:port>
    <wsdl:port name="ServStatisticalInfo" binding="ServStatisticalInfo">
      <soap:address location="urn://Service/ServStatisticalInfo"/>
    </wsdl:port>
  </wsdl:service>
  <wsdl:service name="ServiceFailure">
    <wsdl:port name="ServiceFailure" binding="ServiceFailure">
      <soap:address location="urn://Service/ServiceFailure"/>
    </wsdl:port>
    <wsdl:port name="ServiceInformation" binding="ServiceInformation">
      <soap:address location="urn://Service/ServiceInformation"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

IV. NMWG-Schema für die E2E-Link Statusinformation (Ausschnitt)

```
<nmwg:store xmlns:nmwg="http://ggf.org/ns/nmwg/base/2.0/"
            id="store1" type="E2E_Link_status_information" >
  <nmwg:parameters id="storeId">
    <nmwg:parameter name="DomainName">internet2.edu</nmwg:parameter>
  </nmwg:parameters>

  <nmwg:metadata id="md1">
    <nmwg:subject id="sub-DFN-LRZ">
      <nmwgtopo3:node xmlns:nmwgtopo3="http://ggf.org/ns/nmwg/topology/base/3.0/"
            id="DFN-LRZ" role="somerole" nodeIdRef="somenode">

        <!-- <nmwgtopo3:role>some role</nmwgtopo3:role> -->

        <nmwgtopo3:type>TopologyPoint</nmwgtopo3:type>
        <nmwgtopo3:name type="logical">DFN-LRZ</nmwgtopo3:name>
        <nmwgtopo3:country>Germany</nmwgtopo3:country>
        <nmwgtopo3:city>Munich</nmwgtopo3:city>
        <nmwgtopo3:institution>Leibniz Rechenzentrum</nmwgtopo3:institution>
        <nmwgtopo3:latitude>1.11</nmwgtopo3:latitude>
        <nmwgtopo3:longitude>1.11</nmwgtopo3:longitude>
      </nmwgtopo3:node>
    </nmwg:subject>
  </nmwg:metadata>

  <nmwg:metadata id="md2">
    <nmwg:subject id="sub-DFN-MUE">
      <nmwgtopo3:node xmlns:nmwgtopo3="http://ggf.org/ns/nmwg/topology/base/3.0/"
            id="DFN-MUE" role="somerole" nodeIdRef="somenode">

        <!-- <nmwgtopo3:role>some role</nmwgtopo3:role> -->

        <nmwgtopo3:type>TopologyPoint</nmwgtopo3:type>
        <nmwgtopo3:name type="logical">DFN-MUE</nmwgtopo3:name>
        <nmwgtopo3:country>Germany</nmwgtopo3:country>
        <nmwgtopo3:city>Muenster</nmwgtopo3:city>
        <nmwgtopo3:institution>DFN-Verein</nmwgtopo3:institution>
        <nmwgtopo3:latitude>1.11</nmwgtopo3:latitude>
        <nmwgtopo3:longitude>1.11</nmwgtopo3:longitude>
      </nmwgtopo3:node>
    </nmwg:subject>
  </nmwg:metadata>

  <nmwg:metadata id="md-link-LRZ-SARA-DEISA-001">
    <nmwg:subject id="sub1">
      <nmtl2:link xmlns:nmtl2="http://ggf.org/ns/nmwg/topology/l2/3.0/">
        <nmtl2:name type="logical">DFN-link-1234</nmtl2:name>
        <nmtl2:globalName type="logical">LRZ-SARA-DEISA-001</nmtl2:globalName>
        <nmtl2:type>NREN_Link</nmtl2:type>

        <nmwgtopo3:node xmlns:nmwgtopo3="http://ggf.org/ns/nmwg/topology/base/3.0/"
              nodeIdRef="DFN-LRZ">
          <nmwgtopo3:role>EndPoint</nmwgtopo3:role>
        </nmwgtopo3:node>

        <nmwgtopo3:node xmlns:nmwgtopo3="http://ggf.org/ns/nmwg/topology/base/3.0/"
              nodeIdRef="DFN-MUE">
```

```
          <nmwgtopo3:role>DemarcPoint</nmwgtopo3:role>
        </nmwgtopo3:node>

      </nmtl2:link>
    </nmwg:subject>
  </nmwg:metadata>

  <nmwg:metadata id="md-link-LRZ-SARA-DEISA-002">
    <nmwg:subject id="sub1">
      <nmtl2:link xmlns:nmtl2="http://ggf.org/ns/nmwg/topology/l2/3.0/">
        <nmtl2:name type="logical">DFN-Surfnet-Link-5678</nmtl2:name>
        <nmtl2:globalName type="logical">LRZ-SARA-DEISA-002</nmtl2:globalName>
        <nmtl2:type>ID_Link</nmtl2:type>

        <nmwgtopo3:node xmlns:nmwgtopo3="http://ggf.org/ns/nmwg/topology/base/3.0/"
             nodeIdRef="DFN-MUE">
          <nmwgtopo3:role>DemarcPoint</nmwgtopo3:role>
        </nmwgtopo3:node>

        <nmwgtopo3:node xmlns:nmwgtopo3="http://ggf.org/ns/nmwg/topology/base/3.0/"
               nodeIdRef="SURFnet-MUE">
          <nmwgtopo3:role>DemarcPoint</nmwgtopo3:role>
        </nmwgtopo3:node>

      </nmtl2:link>
    </nmwg:subject>
  </nmwg:metadata>

  <nmwg:data id="d1" metadataIdRef="md-link-LRZ-SARA-DEISA-001">
    <ifevt:datum xmlns:ifevt="http://ggf.org/ns/nmwg/event/status/base/2.0/"
                            timeType="ISO"
                            timeValue="2006-04-20T17:20:00.0+1:00">
      <ifevt:stateOper>Up</ifevt:stateOper>
      <ifevt:stateAdmin>NormalOperation</ifevt:stateAdmin>
    </ifevt:datum>
  </nmwg:data>

  <nmwg:data id="d2" metadataIdRef="md-link-LRZ-SARA-DEISA-002">
    <ifevt:datum xmlns:ifevt="http://ggf.org/ns/nmwg/event/status/base/2.0/"
           timeType="ISO" timeValue="2006-04-20T17:20:00.0+1:00">
      <ifevt:stateOper>Down</ifevt:stateOper>
      <ifevt:stateAdmin>Maintenance</ifevt:stateAdmin>
    </ifevt:datum>
  </nmwg:data>

</nmwg:store>
```

V. Integration der NMWG-Schema in WSDL [SZG 03]

```
<?xml version="1.0"?>
<definitions name="NetworkMeasurement"
   targetNamespace="*NM_NS*/service" xmlns:tns="*NM_NS*/service"
   xmlns:defs="*NM_NS*/definitions" xmlns="http://schemas.xmlsoap.org/wsdl/"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">

<types>
<xsd:schema targetNamespace="*NM_NS*">
   <xsd:include schemaLocation="*REQUEST_SCHEMA:request.xsd*" />
   <xsd:include schemaLocation="*REPORT_SCHEMA:response.xsd*" />
</xsd:schema>
</types>

<message name="NMWG-Request">
    <part name="body" element="nm:request"/>
</message>
<message name="NMWG-Response">
    <part name="body" element="nm:response"/>
</message>

<portType name="NMWG-PortType">
   <operation name="NMWG-Get">
     <input message="tns:NMWG-Request" />
     <output message="tns:NMWG-Response" />
   </operation>
</portType>

<binding name="NMWG-Binding" type="defs:NMWG-PortType">
  <soap:binding style="document"
     transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="NMWG-Get">
    <soap:operation soapAction="*HTTP_OPER:http://localhost/GetNetworkMeasurement*"/>
      <input>
        <soap:body use="literal"/>
      </input>
      <output>
        <soap:body use="literal"/>
      </output>
  </operation>
</binding>

<service name="NMWG-Service">
  <documentation>
     Grid NM-WG Network Measurement service
  </documentation>
  <port name="NMWG-Port" binding="tns:NMWG-Binding">
     <soap:address location="*SVC_URLhttp://localhost:8000*"/>
  </port>
</service>
</definitions>
```

VI. Request/Response der HADES-Topologie (Ausschnitt)

```
<!--Request/Response der HADES-Topologie für LHCOPN-Link
    DE-KIT -- CH-CERN und CH-CERN -- DE-KIT -->
<!-- Request -->

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
 <soap:Body>
  <nmwg:message xmlns="http://ggf.org/ns/nmwg/tools/org/perfsonar/1.0/"
     xmlns:nmwg="http://ggf.org/ns/nmwg/base/2.0/"
     xmlns:nmwgt="http://ggf.org/ns/nmwg/topology/2.0/"
     xmlns:perfsonar="http://ggf.org/ns/nmwg/tools/org/perfsonar/1.0/"
     xmlns:nmtm="http://ggf.org/ns/nmwg/time/2.0/"
     type="SetupDataRequest">
    <nmwg:metadata id="metadata1">
       <perfsonar:subject id="subject1" />
       <nmwg:eventType>http://ggf.org/ns/nmwg/tools/hades/</nmwg:eventType>
       <nmwg:parameters id="param1">
         <nmwg:parameter value="0x0" name="precedence" />
         <nmwg:parameter value="9" name="groupsize" />
         <nmwg:parameter value="60" name="interval" />
         <nmwg:parameter value="41" name="packetsize" />
       </nmwg:parameters>
    </nmwg:metadata>

    <nmwg:metadata id="metadata1-2">
       <perfsonar:subject metadataIdRef="metadata1" id="subject1-2" />
       <nmwg:eventType>http://ggf.org/ns/nmwg/ops/select/2.0</nmwg:eventType>
       <nmwg:parameters id="param1-2">
         <nmwg:parameter name="startTime">1294666800</nmwg:parameter>
         <nmwg:parameter name="endTime">1294667100</nmwg:parameter>
       </nmwg:parameters>
    </nmwg:metadata>
    <nmwg:data metadataIdRef="metadata1-2" id="data1" />
  </nmwg:message>
 </soap:Body>
</soap:Envelope>

<!-- Response -->

<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <soap:Body>
   <nmwg:message xmlns="http://ggf.org/ns/nmwg/tools/org/perfsonar/1.0/"
      xmlns:nmwg="http://ggf.org/ns/nmwg/base/2.0/"
      xmlns:nmwgt="http://ggf.org/ns/nmwg/topology/2.0/"
      xmlns:perfsonar="http://ggf.org/ns/nmwg/tools/org/perfsonar/1.0/"
      xmlns:nmtm="http://ggf.org/ns/nmwg/time/2.0/"
      xmlns:nmwgr="http://ggf.org/ns/nmwg/result/2.0/"
```

```
      type="SetupDataResponse">
     <nmwg:metadata id="metadata1">
        <subject id="subject1"/>
        <nmwg:eventType>http://ggf.org/ns/nmwg/tools/hades/</nmwg:eventType>
        <nmwg:parameters id="param1">
          <nmwg:parameter value="0x0" name="precedence"/>
          <nmwg:parameter value="9" name="groupsize"/>
          <nmwg:parameter value="60" name="interval"/>
          <nmwg:parameter value="41" name="packetsize"/>
        </nmwg:parameters>
      </nmwg:metadata>

      <nmwg:metadata id="metadata1-2">
        <subject metadataIdRef="metadata1" id="subject1-2"/>
        <nmwg:eventType>http://ggf.org/ns/nmwg/ops/select/2.0</nmwg:eventType>
        <nmwg:parameters id="param1-2">
          <nmwg:parameter name="startTime">1294666800</nmwg:parameter>
          <nmwg:parameter name="endTime">1294667100</nmwg:parameter>
        </nmwg:parameters>
      </nmwg:metadata>

      <nmwg:data metadataIdRef="metadata1-2" id="data1"/>
      ....
      <nmwg:metadata id="result30">
        <nmwg:parameters id="param1">
          <nmwg:parameter value="CH-CERN-HADES" name="receiver"/>
          <nmwg:parameter value="9" name="groupsize"/>
          <nmwg:parameter value="60" name="interval"/>
          <nmwg:parameter value="15652" name="mid"/>
          <nmwg:parameter value="0x0" name="precedence"/>
          <nmwg:parameter value="128.142.223.231" name="receiver_ip"/>
          <nmwg:parameter value="DE-KIT-HADES" name="sender"/>
          <nmwg:parameter value="192.108.46.124" name="sender_ip"/>
          <nmwg:parameter value="41" name="packetsize"/>
        </nmwg:parameters>
      </nmwg:metadata>
      ...
      <nmwg:metadata id="result35">
        <nmwg:parameters id="param1">
          <nmwg:parameter value="DE-KIT-HADES" name="receiver"/>
          <nmwg:parameter value="9" name="groupsize"/>
          <nmwg:parameter value="60" name="interval"/>
          <nmwg:parameter value="15657" name="mid"/>
          <nmwg:parameter value="0x0" name="precedence"/>
          <nmwg:parameter value="192.108.46.124" name="receiver_ip"/>
          <nmwg:parameter value="CH-CERN-HADES" name="sender"/>
          <nmwg:parameter value="128.142.223.231" name="sender_ip"/>
          <nmwg:parameter value="41" name="packetsize"/>
        </nmwg:parameters>
     </nmwg:metadata>
     ...
     <nmwg:data metadataIdRef="return_metadata1-2" id="data_return_metadata1-2">
       <nmwgr:datum>More than one measurement result found!</nmwgr:datum>
     </nmwg:data></nmwg:message></soap:Body>
</soap:Envelope>'
```

VII. Request/Response der HADES-Kennzahlen (Ausschnitt)

```
<!-- Request/Response der HADES-Kennzahlen für LHCOPN-Link
      DE-KIT -- CH-CERN und CH-CERN -- DE-KIT  -->

<!-- Request -->

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <nmwg:message xmlns="http://ggf.org/ns/nmwg/tools/org/perfsonar/1.0/"
      xmlns:nmwg="http://ggf.org/ns/nmwg/base/2.0/"
      xmlns:nmwgt="http://ggf.org/ns/nmwg/topology/2.0/"
      xmlns:perfsonar="http://ggf.org/ns/nmwg/tools/org/perfsonar/1.0/"
      xmlns:nmtm="http://ggf.org/ns/nmwg/time/2.0/"
      type="SetupDataRequest">

      <nmwg:metadata id="metadata1">
        <perfsonar:subject id="subject1">
          <nmwgt:endPointPair>
            <nmwgt:src value="CH-CERN-HADES" type="IFName" />
            <nmwgt:dst value="DE-KIT-HADES" type="IFName" />
          </nmwgt:endPointPair>
        </perfsonar:subject>
      <nmwg:eventType>http://ggf.org/ns/nmwg/tools/hades/</nmwg:eventType>
      <nmwg:parameters id="param1">
        <nmwg:parameter value="0x0" name="precedence" />
        <nmwg:parameter value="9" name="groupsize" />
        <nmwg:parameter value="60" name="interval" />
        <nmwg:parameter value="41" name="packetsize" />
      </nmwg:parameters>
      </nmwg:metadata>

      <nmwg:metadata id="metadata1-2">
        <perfsonar:subject metadataIdRef="metadata1" id="subject1-2" />
        <nmwg:eventType>http://ggf.org/ns/nmwg/ops/select/2.0</nmwg:eventType>
        <nmwg:parameters id="param1-2">
          <nmwg:parameter name="startTime">1294666800</nmwg:parameter>
          <nmwg:parameter name="endTime">1294667100</nmwg:parameter>
        </nmwg:parameters>
      </nmwg:metadata>

      <nmwg:data metadataIdRef="metadata1-2" id="data1" />
      ....
      <nmwg:metadata id="metadata99">
        <perfsonar:subject id="subject99">
          <nmwgt:endPointPair>
            <nmwgt:src value="DE-KIT-HADES" type="IFName" />
            <nmwgt:dst value="CH-CERN-HADES" type="IFName" />
          </nmwgt:endPointPair>
        </perfsonar:subject>
        <nmwg:eventType>http://ggf.org/ns/nmwg/tools/hades/</nmwg:eventType>
        <nmwg:parameters id="param99">
```

```
          <nmwg:parameter value="0x0" name="precedence" />
          <nmwg:parameter value="9" name="groupsize" />
          <nmwg:parameter value="60" name="interval" />
          <nmwg:parameter value="41" name="packetsize" />
        </nmwg:parameters>
      </nmwg:metadata>

      <nmwg:metadata id="metadata99-2">
        <perfsonar:subject metadataIdRef="metadata99" id="subject99-2" />
        <nmwg:eventType>http://ggf.org/ns/nmwg/ops/select/2.0</nmwg:eventType>
        <nmwg:parameters id="param99-2">
          <nmwg:parameter name="startTime">1294666800</nmwg:parameter>
              <nmwg:parameter name="endTime">1294667100</nmwg:parameter>
        </nmwg:parameters>
      </nmwg:metadata>

      <nmwg:data metadataIdRef="metadata99-2" id="data99" />
    .....
    </nmwg:message>
  </soap:Body>
</soap:Envelope>

<!-- Response -->

<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <soap:Body>
    <nmwg:message xmlns="http://ggf.org/ns/nmwg/tools/org/perfsonar/1.0/"
      xmlns:nmwg="http://ggf.org/ns/nmwg/base/2.0/"
      xmlns:nmwgt="http://ggf.org/ns/nmwg/topology/2.0/"
      xmlns:perfsonar="http://ggf.org/ns/nmwg/tools/org/perfsonar/1.0/"
      xmlns:nmtm="http://ggf.org/ns/nmwg/time/2.0/"
      xmlns:default="http://ggf.org/ns/nwmg/tools/hades/aggregated"
      type="SetupDataResponse">
      <nmwg:metadata id="metadata1">
        <subject id="subject1">
          <nmwgt:endPointPair>
            <nmwgt:src value="CH-CERN-HADES" type="IFName"/>
            <nmwgt:dst value="DE-KIT-HADES" type="IFName"/>
          </nmwgt:endPointPair>
        </subject>
        <nmwg:eventType>http://ggf.org/ns/nmwg/tools/hades/</nmwg:eventType>
        <nmwg:parameters id="param1">
          <nmwg:parameter value="0x0" name="precedence"/>
          <nmwg:parameter value="9" name="groupsize"/>
          <nmwg:parameter value="60" name="interval"/>
          <nmwg:parameter value="41" name="packetsize"/>
          <nmwg:parameter value="DE-KIT-HADES" name="receiver"/>
          <nmwg:parameter value="15657" name="mid"/>
          <nmwg:parameter value="192.108.46.124" name="receiver_ip"/>
          <nmwg:parameter value="CH-CERN-HADES" name="sender"/>
          <nmwg:parameter value="128.142.223.231" name="sender_ip"/>
        </nmwg:parameters>
      </nmwg:metadata>

      <nmwg:metadata id="metadata1-2">
        <subject metadataIdRef="metadata1" id="subject1-2"/>
        <nmwg:eventType>http://ggf.org/ns/nmwg/ops/select/2.0</nmwg:eventType>
```

```
    <nmwg:parameters id="param1-2">
      <nmwg:parameter name="startTime">1294666800</nmwg:parameter>
      <nmwg:parameter name="endTime">1294667100</nmwg:parameter>
    </nmwg:parameters>
  </nmwg:metadata>

  <nmwg:data metadataIdRef="metadata1-2" id="data1">
    <default:datum
       xmlns="http://ggf.org/ns/nwmg/tools/hades/aggregated"
       med_delay="0.00570082664489746" med_ipdv_jitter="-5.96046447753906e-06"
       min_ipdv_jitter="-0.000265836715698242"
       time="1294666803.20085"
       sync="yes" min_delay="0.00568294525146484"
       max_delay="0.00577592849731445"
       duplicates="0" loss="0"
       max_ipdv_jitter="9.20295715332031e-05"/>

    <default:datum
       xmlns="http://ggf.org/ns/nwmg/tools/hades/aggregated"
       med_delay="0.00571799278259277"
       med_ipdv_jitter="1.09672546386719e-05"
       min_ipdv_jitter="-0.00107479095458984"
       time="1294666863.20041"
       sync="yes" min_delay="0.00568199157714844"
       max_delay="0.00678586959838867"
       duplicates="0" loss="0"
       max_ipdv_jitter="0.00110387802124023"/>

     <default:datum
        xmlns="http://ggf.org/ns/nwmg/tools/hades/aggregated"
        med_delay="0.00570893287658691"
        med_ipdv_jitter="-7.15255737304688e-06"
        min_ipdv_jitter="-0.000436067581176758"
        time="1294666923.20098"
        sync="yes" min_delay="0.00569295883178711"
        max_delay="0.00594401359558105"
        duplicates="0" loss="0"
        max_ipdv_jitter="0.000251054763793945"/>
     <default:datum
        xmlns="http://ggf.org/ns/nwmg/tools/hades/aggregated"
        med_delay="0.00568079948425293"
        med_ipdv_jitter="-2.86102294921875e-06"
        min_ipdv_jitter="-0.000577926635742188"
        time="1294666983.20055"
        sync="yes" min_delay="0.00566792488098145"
        max_delay="0.00624799728393555"
        duplicates="0" loss="0"
        max_ipdv_jitter="0.000514030456542969"/>
     <default:datum
        xmlns="http://ggf.org/ns/nwmg/tools/hades/aggregated"
        med_delay="0.00600099563598633"
        med_ipdv_jitter="0.000292062759399414"
        min_ipdv_jitter="-0.00100302696228027"
        time="1294667043.20111"
        sync="yes" min_delay="0.00566601753234863"
        max_delay="0.00669693946838379"
        duplicates="0" loss="0"
        max_ipdv_jitter="0.00101995468139648"/>
  </nmwg:data>
...
  <nmwg:metadata id="metadata99">
```

```
  <subject id="subject99">
    <nmwgt:endPointPair>
      <nmwgt:src value="DE-KIT-HADES" type="IFName"/>
      <nmwgt:dst value="CH-CERN-HADES" type="IFName"/>
    </nmwgt:endPointPair>
  </subject>
  <nmwg:eventType>http://ggf.org/ns/nmwg/tools/hades/</nmwg:eventType>
  <nmwg:parameters id="param99">
    <nmwg:parameter value="0x0" name="precedence"/>
    <nmwg:parameter value="9" name="groupsize"/>
    <nmwg:parameter value="60" name="interval"/>
    <nmwg:parameter value="41" name="packetsize"/>
    <nmwg:parameter value="CH-CERN-HADES" name="receiver"/>
    <nmwg:parameter value="15652" name="mid"/>
    <nmwg:parameter value="128.142.223.231" name="receiver_ip"/>
    <nmwg:parameter value="DE-KIT-HADES" name="sender"/>
    <nmwg:parameter value="192.108.46.124" name="sender_ip"/>
  </nmwg:parameters>
</nmwg:metadata>

<nmwg:metadata id="metadata99-2">
  <subject metadataIdRef="metadata99" id="subject99-2"/>
  <nmwg:eventType>http://ggf.org/ns/nmwg/ops/select/2.0</nmwg:eventType>
  <nmwg:parameters id="param99-2">
    <nmwg:parameter name="startTime">1294666800</nmwg:parameter>
    <nmwg:parameter name="endTime">1294667100</nmwg:parameter>
  </nmwg:parameters>
</nmwg:metadata>

<nmwg:data metadataIdRef="metadata99-2" id="data99">
  <default:datum
     xmlns="http://ggf.org/ns/nwmg/tools/hades/aggregated"
     med_delay="0.00562095642089844"
     med_ipdv_jitter="3.09944152832031e-06"
     min_ipdv_jitter="-2.09808349609375e-05"
     time="1294666801.69151"
     sync="yes" min_delay="0.00561094284057617"
     max_delay="0.00564289093017578"
     duplicates="0" loss="0"
     max_ipdv_jitter="2.19345092773438e-05"/>
  <default:datum
     xmlns="http://ggf.org/ns/nwmg/tools/hades/aggregated"
     med_delay="0.0056300163269043"
     med_ipdv_jitter="-9.5367431640625e-07"
     min_ipdv_jitter="-7.31945037841797e-05"
     time="1294666861.69128"
     sync="yes" min_delay="0.00562000274658203"
     max_delay="0.00569415092468262"
     duplicates="0" loss="0"
     max_ipdv_jitter="7.31945037841797e-05"/>
  <default:datum
     xmlns="http://ggf.org/ns/nwmg/tools/hades/aggregated"
     med_delay="0.00560879707336426"
     med_ipdv_jitter="-9.5367431640625e-07"
     min_ipdv_jitter="-3.31401824951172e-05"
     time="1294666921.69106"
     sync="yes" min_delay="0.00559496879577637"
     max_delay="0.0056297779083252"
     duplicates="0" loss="0"
     max_ipdv_jitter="2.00271606445312e-05"/>
  <default:datum
```

```
            xmlns="http://ggf.org/ns/nwmg/tools/hades/aggregated"
            med_delay="0.00563502311706543"
            med_ipdv_jitter="1.00135803222656e-05"
            min_ipdv_jitter="-4.02927398681641e-05"
            time="1294666981.69083"
            sync="yes" min_delay="0.00559687614440918"
            max_delay="0.0056459903717041"
            duplicates="0" loss="0"
            max_ipdv_jitter="4.22000885009766e-05"/>
         <default:datum
            xmlns="http://ggf.org/ns/nwmg/tools/hades/aggregated"
            med_delay="0.00561094284057617"
            med_ipdv_jitter="0"
            min_ipdv_jitter="-2.59876251220703e-05"
            time="1294667041.6916"
            sync="yes" min_delay="0.00559806823730469"
            max_delay="0.00562095642089844"
            duplicates="0" loss="0"
            max_ipdv_jitter="1.00135803222656e-05"/>
      </nmwg:data>
      ...
    </nmwg:message>
  </soap:Body>
</soap:Envelope>
```

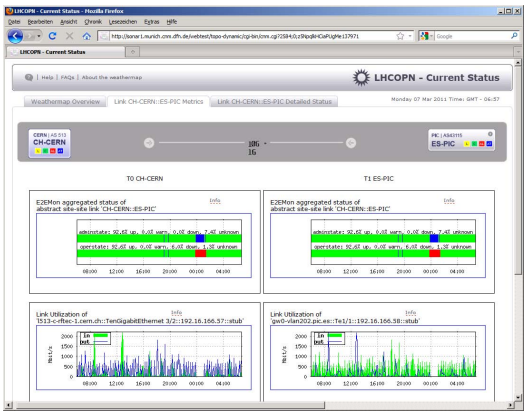
VIII. Die Schema der perfsonar Datenbank

```
perfsonar=# \d network_node
                    Table "public.network_node"
        Column         |               Type                | Modifiers
-----------------------+-----------------------------------+-----------
 valid_from            | timestamp without time zone       |
 valid_to              | timestamp without time zone       |
 time_last_seen        | timestamp without time zone       |
 object_id             | integer                           | not null
 object_version        | integer                           | not null
 display_count         | smallint                          |
 valid                 | boolean                           |
 features              | text                              |
 ip_management_address | text                              |
 network_node_type_id  | integer                           |
 location_id           | integer                           |
 dns_name1             | text                              |
 dns_name2             | text                              |
 usage_type            | information_schema.cardinal_number |
 name                  | text                              |
Indexes:
    "network_node_pkey" PRIMARY KEY, btree (object_id, object_version)
    "fki_node_" btree (object_id) CLUSTER
Foreign-key constraints:
    "network_node_object_id_fkey" FOREIGN KEY (object_id) REFERENCES
    network_node_key(object_id) ON UPDATE RESTRICT ON DELETE RESTRICT
Inherits: monitoring_object,
          network_node_base

perfsonar=# \d network_interface
                      Table "public.network_interface"
             Column              |               Type                | Modifiers
---------------------------------+-----------------------------------+-----------
 valid_from                      | timestamp without time zone       |
 valid_to                        | timestamp without time zone       |
 time_last_seen                  | timestamp without time zone       |
 object_id                       | integer                           | not null
 object_version                  | integer                           | not null
 display_count                   | smallint                          |
 valid                           | boolean                           |
 features                        | text                              |
 if_highspeed                    | bigint                            |
 if_mtu                          | text                              |
 if_type                         | information_schema.cardinal_number |
 if_index                        | integer                           |
 parent_interface_object_id      | integer                           |
 layer                           | integer                           |
 default_ip_interface_address    | text                              |
 default_ip_network_subnet_prefix | text                              |
 if_alias                        | text                              |
 if_description                  | text                              |
 if_speed                        | bigint                            |
 ospf_link_type                  | text                              |
 speed                           | bigint                            |
 name                            | text                              |
Indexes:
    "network_interface_pkey" PRIMARY KEY, btree (object_id, object_version)
    "fki_interface_" btree (object_id) CLUSTER
Foreign-key constraints:
```

```
    "network_interface_object_id_fkey" FOREIGN KEY (object_id) REFERENCES
    network_interface_key(object_id) ON UPDATE RESTRICT ON DELETE RESTRICT
Inherits: monitoring_object,
          network_interface_base

perfsonar=# \d network
                    Table "public.network"
      Column       |            Type             | Modifiers
-------------------+-----------------------------+-----------
 network_type      | text                        |
 bandwidth         | bigint                      |
 description       | text                        |
 network_name      | text                        |
 ospf_network_type | text                        |
 valid_from        | timestamp without time zone |
 valid_to          | timestamp without time zone |
 time_last_seen    | timestamp without time zone |
 object_id         | integer                     | not null
 object_version    | integer                     | not null
 display_count     | smallint                    |
 valid             | boolean                     |
 features          | text                        |
Indexes:
    "network_pkey" PRIMARY KEY, btree (object_id, object_version)
    "fki_connection_" btree (object_id) CLUSTER
Foreign-key constraints:
    "network_connection_object_id_fkey" FOREIGN KEY (object_id) REFERENCES
    network_key(object_id) ON UPDATE RESTRICT ON DELETE RESTRICT
Inherits: network_base,
          monitoring_object

perfsonar-# \d domain
                   Table "public.domain"
        Column         |  Type   |                Modifiers
-----------------------+---------+----------------------------------
 domain_id             | integer | not null default nextval
                                   ('domain_domain_id_seq'::regclass)
 domain_name           | text    |
 domain_description    | text    |
 domain_external_id    | text    |
 domain_networks_name  | text    |
 country_iso3166_code  | text    |
 flags                 | integer |
 features              | text    |
Indexes:
    "domain_pkey" PRIMARY KEY, btree (domain_id) CLUSTER
Referenced by:
  "domain_named_object_domain_id_fkey" IN domain_named_object
   FOREIGN KEY (domain_id) REFERENCES domain(domain_id) ON UPDATE
   RESTRICT ON DELETE RESTRICT
  "ip_network_interface_key_base_domain_id_fkey" IN
   ip_subnetwork_interface_key_base  FOREIGN KEY (domain_id)
   REFERENCES domain(domain_id) ON UPDATE RESTRICT ON DELETE RESTRICT
  "ip_network_key_base_domain_id_fkey" IN ip_subnetwork_key_base
   FOREIGN KEY (domain_id) REFERENCES domain(domain_id) ON UPDATE
   RESTRICT ON DELETE RESTRICT
  "network_connection_key_base_domain_id_fkey" IN network_key_base
   FOREIGN KEY (domain_id) REFERENCES domain(domain_id) ON UPDATE
   RESTRICT ON DELETE RESTRICT
  "network_node_key_base_domain_id_fkey" IN network_node_key_base
   FOREIGN KEY (domain_id) REFERENCES domain(domain_id) ON UPDATE
```

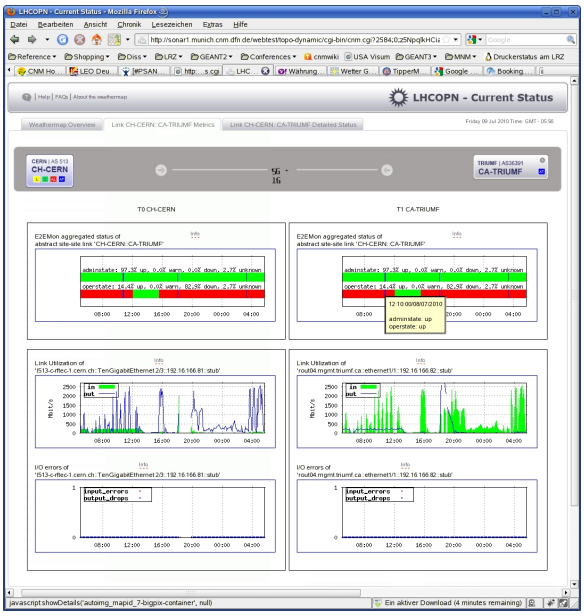
```
   RESTRICT ON DELETE RESTRICT

Table "public.network_to_interface"
           Column            |     Type     |             Modifiers
-----------------------------+--------------+-----------------------------
 valid_from                  | timestamp  |
 valid_to                    | timestamp  |
 time_last_seen              | timestamp  |
 network_object_id           | integer    |
 network_interface_object_id | integer    |
 interface_order             | integer    |
 association_link_id         | integer    | not null default nextval
                                            ('network_to_interface_
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣association_link_id_seq'::regclass)
Indexes:
    "network_to_interface_pkey" PRIMARY KEY, btree (association_link_id)
    "fki_interface2" btree (network_interface_object_id) CLUSTER
    "fki_network_" btree (network_object_id)
Foreign-key constraints:
    "network_to_interface_network_interface_object_id_fkey"
     FOREIGN KEY (network_interface_object_id) REFERENCES
     network_interface_key(object_id) ON UPDATE RESTRICT ON DELETE RESTRICT
    "network_to_interface_network_object_id_fkey" FOREIGN KEY (network_object_id)
     REFERENCES network_key(object_id) ON UPDATE RESTRICT ON DELETE RESTRICT
Inherits: object_with_historyinfo
```

IX. Fehlererkennen in der LHCOPN-Wetterkarte



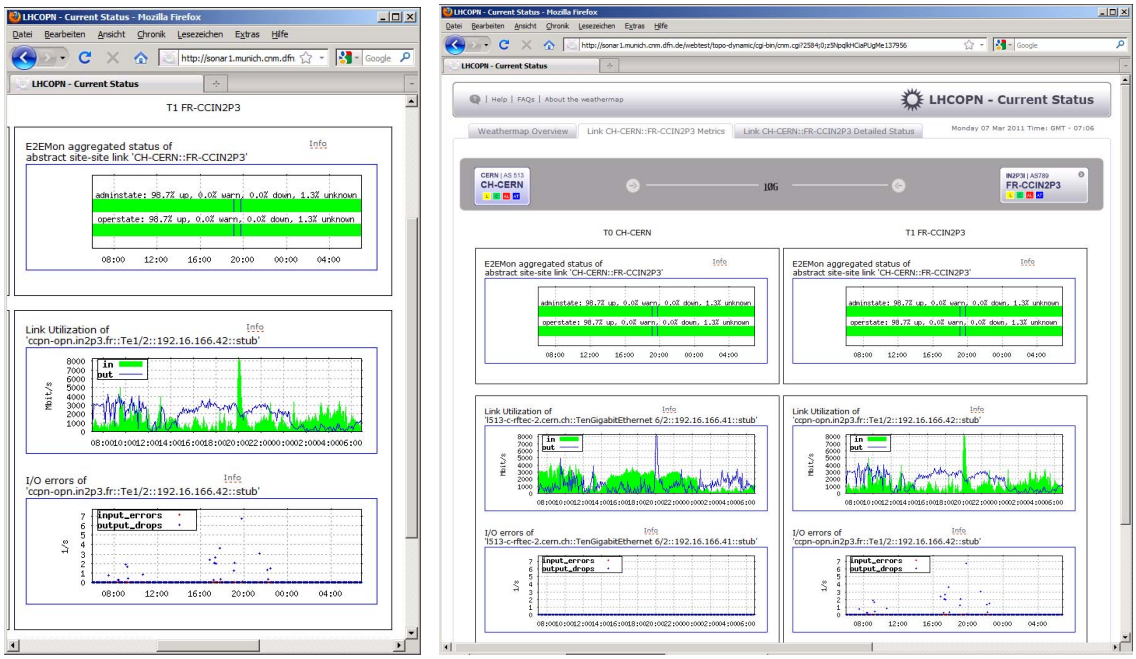
(a)

(b)



(c)

Abbildung 0.1.: Ausfall bzw. fehlenden Daten vom E2E-Mon System korreliert mit fehlenden Messungen der Utilisation



(a) (b)

Abbildung 0.2.: Große Anzahl an Input Errors/Output Drops korreliert mit Fluktuation der Utilisation. Der Ausreißer der Utilisation korrespondiert dem Maximalanzahl an Output Drops

(a) (b)

(c) (d)

Abbildung 0.3.: Gestiegenen Hop count (von 5 auf 20 Hops) zeigt stark geänderte Routen wegen eines Ausfalls der Hauptstrecke. Dies korreliert mit der Abwesentheit bzw. abweichende Werte der Hades IPDV/OWD Messungen in demselben Zeitraum

Abkürzungen

AA	Authentication and Authorization Service
AMM	Architecture Metamodel
API	Application Programming Interface
ARS	Remedy Action Request System
ATL	Atlas Transformation Language
BAdW	Bayerischen Akademie der Wissenschaften
BWCTL	Band Width Controller
CDG	Causal Directed Graph
COBIT	Control Objectives for Information and Related Technology
CERN	Europäisches Labor für Teilchenphysik (früher: *Conseil Européen pour la Recherche Nucléaire*)
CI	Configuration Item
CIM	Computation Independent Model
CMDB	Configuration Management Database
CNM	Customer Network Management
CSM	Customer Service Managememnt
DANTE	Delivering Advanced Networking Technology to Europe
DFRMT	Domain-Fault-Reporting-Management-Team
DFRT	Domain-Fault-Resolution-Team
DFM	Domain-Fault-Manager

DFO	Domain-Fault-Operator
DMS	Domain-Monitoring-System
Dom-SD	Domain Service Desk
Dom-FM	Domain Fault Manager
Dom-TS	Domain Technical Specialist
Dom-MS	Domain Monitoring System
E2E	End-to-End
E2ECU	E2E Coordination Unit
E2EMon	E2E Monitoring System
eTOM	Enhanced Telecom Operations Map
FCAPS	Fault, Configuration, Accounting, Performance, Security Management
GFMT	Global-Fault-Management-Team
GFCM	Global-Fault-Coordinator-Manager
GFP-F	Generic Framing Protocol Encapsulation
GUI	Graphical User Interface
HADES	Hades Active Delay Evaluation System
HADES MA	HADES Measurement Archive
HSWT	Hochschule Weihenstephan-Triesdorf
ISO	International Organization of Standardization
IEC	International Electrotechnical Commission
IETF	Internet Engineering Task Force
ioCMDB	interorganisationale CMDB
ioFMA	interorganisationale Fehlermanagementarchitektur
io-FM	Interorganizational Fault Manager
io-SD	Interorganizational Service Desk
io-FO	Interorganizational Fault Operator
io-MS	Interorganizational Monitoring System
IPDV	IP Packet Delay Variation
IPPM	IP Performance Metrics

ITIL	IT Infrastructure Library
ITPM	IT Process Model
ITS	Incident Ticketing Systeme
ITSM	IT-Service-Management
JMI	Java Metadata Interface
HM	Hochschule München
KPI	Key Performance Indicators
LCG	LHC Compute Grid
LHC	Large Hadron Collider
LHCOPN	LHC Optical Private Network
LMU	Ludwig-Maximilians-Universität
LRZ	Leibniz-Rechenzentrum
LSP	Label Switched Path
MA	Measurement Archive
MDA	Model Driven Architecture
MIB	Managementinformationsbasis
MO	Managementobjekt
MOF	Microsoft Operations Framework
MP	Measurement Point
MPLS	Multiprotocol Label Switching
MWN	Münchner Wissenschaftsnetz
NREN	National Research and Educational Network
NMS	Network Management System
NMWG	Network Measurement Working Group
OCL	Object Constraint Language
OGC	Office of Government Commerce
OGF	Open Grid Forum
OMG	Object Management Group
OPN	Optisches Privates Netz

OSI	Open System Interconnection
OTRS	Open Ticket Request System
OWD	One-way Delay
OWDV	One-way Delay Variation
OWPL	One-way Packet Loss
PDM	Platform Description Model
perfSONAR	Performance focused Service Oriented Network monitoring ARchitecture
pS UI	perfSONAR User Interface
PIM	Platform Independent Model
PSM	Platform Specific Model
QoS	Quality of Service
RRD MA	Round Robin Database Measurement Archive
SAPs	Service Access Points
SID	Shared Information/Data Model
SLA	Service Level Agreement
SLAs	Service Level Agreements
SLM	Service Level Management
SOAP	Simple Object Access Protocol
SP	Service Provider
SQL MA	Structured Query Language Measurement Archive
SYMIAN	SYMulation for Incident ANalysis
TMF	TeleManagementForum
TNOC	Transmission Network Operation Centre
TTS	Trouble Ticket System
TUM	Technischen Universität München
UML	Unified Modeling Language
WSDL	Web Services Description Language
XMI	XML Meta-Data Interchange
XML	Extensible Markup Language

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[AbED 09] Abouelela, Mohamed und Mohamed El-Darieby: *PCE-based hierarchical segment restoration.* In: *IM'09: Proceedings of the 11th IFIP/IEEE international conference on Symposium on Integrated Network Management*, Seiten 164–171, Piscataway, NJ, USA, 2009. IEEE Press.

[ALR 01] Avizienis, A., J.-C. Laprie und B. Randell: *Fundamental Concepts of Dependability.* Technischer Bericht 1145, LAAS-CNRS, April 2001.

[ALRL 04] Avizienis, A., J.-C. Laprie, B. Randell und C. Landwehr: *Basic Concepts and Taxonomy of Dependable and Secure Computing.* Technischer Bericht 47, Institute for Systems Research (ISR), 2004.

[BBI+ 04] Booch, Grady, Alan Brown, Sridhar Iyengar, Jim Rumbaugh und Bran Selic: *An MDA Manifesto.* 2004.

[BBMR 05] Beygelzimer, Alina, Mark Brodie, Sheng Ma und Irina Rish: *Test-Based Diagnosis: Tree and Matrix Representations.* In: *Proc. IM-2005*, Seiten 529–542, Nice, France, May 2005. .

[BCF 94] Bouloutas, A., S. Calo und A. Finkel: *Alarm Correlation and Fault Identification in Communication Networks.* IEEE Transactions on Communications, 42(2):523–533, 1994.

[BGSS 06] Brenner, M., M. Garschhammer, M. Sailer und T. Schaaf: *CMDB - Yet another MIB? On Reusing Management Model Concepts in ITIL Configuration Management.* In: *Proc. DSOM 2006*, Seiten 269–280, Dublin, Ireland, October 2006. .

[BHLJ 04] Bézivin, J., S. Hammoudi, D. Lopes und F. Jouault: *Applying MDA Approach for Web Service Platform.* In: *Proceedings of the 8th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2004)*, April 2004.

[Biar 10] Biardski, Christoph: *Integrierte Speichersystem-Architektur zur Unterstützung hochschulübergreifender IT-Dienste.* In: Bode, A. und R. Borgeest (Herausgeber): *Informationsmanagement in Hochschulen.* Springer-Verlag, Berlin, 2010.

[BMM 05] Bos, Erik-Jan, Edoardo Martelli und Paolo Moroni: *LHC Tier-0 to Tier-1 High-Level Network Architecture.* Technischer Bericht, CERN, 2005.

[BoBo 10] Bode, A. und R. Borgeest (Herausgeber): *Informationsmanagement in Hochschulen.* Springer-Verlag, Berlin, 2010.

[BrDu10] Bruegge, Bernd und Allen H. Dutoit: *Object-Oriented Software Engineering Using UML, Patterns, and Java.* Prentice Hall, 2010.

[Bren 07] Brenner, Michael: *Werkzeugunterstützung für ITIL-orientiertes Dienstmanagement.* Dissertation, Ludwig-Maximilians-Universität München, 2007.

[BRJ 06] Booch, G., J. Rumbaugh und I. Jacobson: *Das UML-Benutzerhandbuch, Aktuell zur Version 2.0.* Addison-Wesley, 2 Auflage, 2006.

[BST 08] Bartolini, Claudio, Cesare Stefanelli und Mauro Tortonesi: *SYMIAN: A Simulation Tool for the Optimization of the IT Incident Management Process.* In: *Proceedings of 19th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM 2008)*, Band 5273 der Reihe *Lecture Notes in Computer Science*, Seiten 83–94, Samos Island, Greece, 2008. Springer.

[BST 09] Bartolini, C., C. Stefanelli und M. Tortonesi: *Business-impact analysis and simulation of critical incidents in IT service management.* In: *IM'09: Proceedings of the 11th IFIP/IEEE international conference on Symposium on Integrated Network Management*, Seiten 9–16, Piscataway, NJ, USA, 2009. IEEE Press.

[BWCTL] *Bandwidth Test Controller (BWCTL).*

[Cand 03] Candea, George: *The basics of Dependability.* Lecture notes for Principles of Dependable Systems, Fall 2003.

[ChSa 01] Chessa, S und P Santi: *Comparison-Based System-Level Fault Diagnosis in Ad Hoc Networks.* In: *Proceedings of the 20th Symposium on Reliable Distributed Systems*, Seiten 257–266, New Orleans, USA, 2001. .

[CIM 10] *Common Information Model (CIM) Version 2.27.0.* Technischer Bericht, Distributed Management Task Force, November 2010.

[COBIT] *Common Objectives for Information and related Technology (COBIT 4.0).* http://www.isaca.org/cobit.

[CQM+ 09] Cheng, Lu, Xue-song Qiu, Luoming Meng, Yan Qiao und Zhi-qing Li: *Probabilistic fault diagnosis for IT services in noisy and dynamic environments.* In: *IM'09: Proceedings of the 11th IFIP/IEEE international conference on Symposium on Integrated Network Management*, Seiten 149–156, Piscataway, NJ, USA, 2009. IEEE Press.

[DBCAF 06] Duarte, Alexandre, Francisco Brasileiro, Walfredo Cirne und Jose Alencar Filho: *Collaborative Fault Diagnosis in Grids through Automated Tests.* In: *AINA '06: Proceedings of the 20th International Conference on Advanced Information Networking and Applications*, Seiten 69–74, Washington, DC, USA, 2006. IEEE Computer Society.

[DEISA] *Distributed European Infrastructure for Supercomputing Applications.* http://www.deisa.eu/.

[Dreo 02] Dreo Rodosek, G.: *A Framework for IT Service Management.* Habilitation, University of Munich (LMU), Department of Computer Science, Munich, Germany, Juni 2002.

[E2EMon] *E2E Monitoring System (E2EMon) Development Homepage.* http://cnmdev.lrz-muenchen.de/e2e/.

[EGEE] *Joint Research Activity 4, Enabling Grids for E-SciencE (EGEE) project.* http://egee-jra4.web.cern.ch/EGEE-JRA4/.

[eTOM] *enhanced Telecom Operations Map (eTOM), The Business Process Framework for the Information and Communications Services Industry.* GB 921 Release 5.0, TeleManagement Forum, April 2005.

[GÉ 07] GÉANT2: *GÉANT2 Homepage.* Elektronische Quelle URL: `http://www.geant2.net`, 2007 (letzter Zugriff: 14.04.2007).

[Geant11] GÉANT: *GéANT Homepage.* `http://www.geant.net/`, 2011.

[GHH+ 01] Garschhammer, M., R. Hauck, H.-G. Hegering, B. Kempter, M. Langer, M. Nerb, I. Radisic, H. Roelle und H. Schmidt: *Towards Generic Service Management Concepts - A Service Model Based Approach.* In: *Proceedings of the 7th IFIP/IEEE International Symposium on Integrated Network Management (IM 2001)*, Seiten 719–732, Seattle, Washington, USA, May 2001. .

[GHH+ 02] Garschhammer, M., R. Hauck, H.-G. Hegering, B. Kempter, I. Radisic, H. Roelle und H. Schmidt: *A Case-Driven Methodology for Applying the MNM Service Model.* In: *Proceedings of the 8th IFIP/IEEE International Network and Operations Management Symposium (NOMS 2002)*, Seiten 697–710, Florence, Italy, April 2002. .

[GHK+ 01] Garschhammer, M., R. Hauck, B. Kempter, I. Radisic, H. Roelle und H. Schmidt: *The MNM Service Model - Refined Views on Generic Service Management.* Journal of Communications and Networks, 3(4), November 2001.

[Gopa 00] Gopal, Rajeev: *Layered model for supporting fault isolation and recovery.* In: *NOMS 2000: IEEE/IFIP Network Operations and Management Symposium 'The Networked Planet: Management Beyond 2000'*, Seiten 729–742, Honolulu, Hawaii, USA, April 2000. .

[Gora 05] Goranson, H: *Semantic Distance and Enterprise Integration.* In: *Knowledge Sharing in the Integrated Enterprise*, Band 183 der Reihe *IFIP International Federation for Information Processing*, Seiten 39–52. Springer Boston, 2005.

[GPL+ 09] Gupta, Rajeev, K. Hima Prasad, Laura Luan, Daniela Rosu und Chris Ward: *Multi-dimensional Knowledge Integration for Efficient Incident Management in a Services Cloud.* In: *SCC '09: Proceedings of the 2009 IEEE International Conference on Services Computing*, Seiten 57–64, Washington, DC, USA, 2009. IEEE Computer Society.

[GPM 08a] Gupta, Rajeev, K. Hima Prasad und Mukesh K. Mohania: *Information integration techniques to automate incident management.* In: *Proceedings of the IEEE/IFIP Network Operations and Management Symposium: Pervasive Management for Ubiquitous Networks and Services (NOMS 2008)*, Seiten 979–982, Salvador Bahia, Brazil, April 2008. .

[GPM 08b] Gupta, Rajeev, K Hima Prasad und Mukesh Mohania: *Automating ITSM Incident Management Process.* In: *ICAC '08: Proceedings of the 2008 International Conference on Autonomic Computing*, Seiten 141–150, Chicago, Juni 2008. .

[GPR 06] Gruhn, Volker, Daniel Pieper und Carsten Röttgers: *MDA – Effektives Software-Engineering mit UML2 und Eclipse.* Springer Verlag, 2006.

[Grid 09] GridKa: *GridKa Homepage.* Elektronische Quelle URL: `http://grid.fzk.de/`, 2009 (letzter Zugriff: 22.12.2009).

[HADES] *HADES.* http://www.win-labor.dfn.de/cgi-bin/hades/selectnew.pl?config=.

[Hamm 09] Hamm, Matthias: *IT Service Management Prozesse verketteter Dienste.* Dissertation, Ludwig–Maximilians–Universität München, Juni 2009.

[HAN 99] Hegering, H.-G., S. Abeck und B. Neumair: *Integrated Management of Networked Systems - Concepts, Architectures and their Operational Application.* Morgan Kaufmann Publishers, 1999.

[Hane 07] Hanemann, A.: *Automated IT Service Fault Management Based on Event Correlation Techniques.* PhD thesis, University of Munich, Department of Computer Science, Munich, Germany, May 2007.

[HaYa 07] Yampolskiy, M. und M. K. Hamm: *Management of Multidomain End–to–End Links. A Federated Approach for the Pan–European Research Network 2.* In: *Moving from Bits to Business Value: Proceedings of the 2007 Integrated Management Symposium*, Seiten 189–198, München, Germany, Mai 2007. IFIP/IEEE.

[HaYa 08] Hamm, Matthias und Mark Yampolskiy: *E2E Link Monitoring: System Design and Documentation.* Technischer Bericht, GN2 Project, 2008.

[HBB+ 05] Hanemann, A., J. Boote, E. Boyd, J. Durand, L. Kudarimoti, R. Lapacz, N. Simar, M. Swany, S. Trocha und J. Zurawski: *PerfSONAR: A Service-Oriented Architecture for Multi-Domain Network Monitoring.* In: *Proceedings of the 3rd International Conference on Service-Oriented Computing (ICSOC 2005)*, Seiten 241–254, Amsterdam, The Netherlands, December 2005. ACM.

[Hedl 05] Hedlund, G.: *Assumptions of Hierarchy and Heterarchy, with Applications to the Management of the Multinational Corporation.* In: Ghoshal, S. und E. Westney (Herausgeber): *Organizational Theory and the Multinational Corporation*, Seiten 198–221. London, 2nd Auflage, 2005.

[Henn 06] Hennicker, Rolf: *Objektorientierte Softwareentwicklung.* Vorlesungsskript WS2005/2006, 2006.

[HJKM 06] Hanemann, A., V. Jeliazkov, O. Kvittem, L. Marta, J. Metzger und I. Velimirovic: *Complementary Visualization of perfSONAR Network Performance Measurements.* In: *Proceedings of the International Conference on Internet Surveillance and Protection (ICISP)*, Band 2006, Cap Esterel, France, August 2006. IARIA/IEEE.

[HJL 05] Hammoudi, Slimane, Jérôme Janvier und Denivaldo Lopes: *Mapping Versus Transformation in MDA: Generating Transformation Definition from Mapping Specification.* In: *Proceedings of the International Workshop on Vocabularies, Ontologies and Rules for The Enterprise (VORTE),*, Enschede, The Netherlands, 2005. .

[HKM+ 08] Hanemann, A., S. Kraft, P. Marcu, J. Reinwand, H. Reiser, D. Schmitz und V. Venus: *perfSONAR: Performance Monitoring in europäischen Forschungsnetzen.* In: *Erstes DFN-Forum Kommunikationstechnologien - Verteilte Systeme im Wissenschaftsbereich.* DFN, GI-Verlag, mai 2008.

[HKMY 10] Hamm, M. K., S. Knittl, P. Marcu und M. Yampolskiy: *Modellierung interorganisationaler IT–Service–Managementprozesse.* PIK — Praxis der Informationsverarbeitung und Kommunikation, Dezember 2010.

[HKT+ 10] Hirota, E, K Kinoshita, H Tode, K Murakami, S Kikuchi, Tsuchiya S, A Sekiguchi und T Katsuyama: *Multilayer Failure Detection Method for the Network Services Based on Distributed Components.* In: *NOMS 2000: 12th IEEE/IFIP Network Operations and Management Symposium 'Towards Management of Future Networks and Services'*, April 2010.

[HLMS 06a] Hanemann, A., A. Liakopoulos, M. Molina und D. M. Swany: *A study on network performance metrics and their composition.* Campus–Wide Information Systems, 2006(23):268–282, September 2006.

[HoFa 07] Hofer, Jürgen und Thomas Fahringer: *Grid Application Fault Diagnosis Using Wrapper Services and Machine Learning.* In: *ICSOC '07: Proceedings of the 5th international conference on Service-Oriented Computing*, Seiten 233–244, Berlin, Heidelberg, 2007. Springer-Verlag.

[HoFa 08] Hofer, Jürgen und Thomas Fahringer: *Grid Application Fault Diagnosis Using Wrapper Services and Machine Learning.* Grid Middleware and Services, (3), 2008.

[HoKn 08] Hommel, W. und S. Knittl: *An Inter–Organizational Configuration Management Database as Key Enabler for Future IT Service Management Processes* . In: *eChallenges 2008*, Band 2008, Stockholm, Schweden, Oktober 2008. .

[HoKn 10] Hommel, W. und S. Knittl: *Aufbau von organisationsübergreifenden Fehlermanagementprozessen im Projekt IntegraTUM.* In: Bode, A. und R. Borgeest (Herausgeber): *Informationsmanagement in Hochschulen.* Springer-Verlag, Berlin, 2010.

[Homm 07] Hommel, W.: *Architektur- und Werkzeugkonzepte für föderiertes Identitäts–Management.* Dissertation, Ludwig–Maximilians–Universität München, Juli 2007.

[IntegraTUM] *IntegraTUM project, Technische Universität München.* http://portal.-mytum.de/iuk/integratum/index_html% else.

[ipsh10] IPSphere: *IPSphere Homepage.* http://www.tmforum.org/ipsphere.

[ISO 20000:1] *ISO/IEC 20000-1:2005 - Information Technology - Service Management - Part 1: Specification.* Technischer Bericht, International Organization for Standardization, Dezember 2005.

[ISO 20000:2] *ISO/IEC 20000-2:2005 - Information Technology - Service Management - Part 2: Code of Practice.* Technischer Bericht, International Organization for Standardization, Dezember 2005.

[KGF 07] Klie, Torsten, Felix Gebhard und Stefan Fischer: *Towards Automatic Composition of Network Management Web Services.* In: *Integrated Network Management, IM 2007. 10th IFIP/IEEE International Symposium on Integrated Network Management*, Seiten 769–772, Munich, Germany, 2007. .

[KHS 01] Keller, A., H. Heger und K. Schopmeyer: *Towards a CIM schema for runtime application management.* In: *Proceedings of the 12th International IFIP/IEEE Workshop on Distributed Systems: Operations and Management (DSOM 2001)*, Nancy, France, Oktober 2001. .

[KiHo 04] Kirmani, E. und C. S. Hood: *Diagnosing Network States through Intelligent Probing.* In: *Proceedings of the 9th IFIP/IEEE International Network Management and Operations Symposium (NOMS 2004)*, Seiten 147–160, Seoul, Korea, April 2004. .

[Knit 10] Knittl, Silvia: *Unterstützung der IT-Service-Management-Prozesse an der Technischen Universität München durch eine Configuration-Management-Database.* In: Bode, A. und R. Borgeest (Herausgeber): *Informationsmanagement in Hochschulen*, Kapitel Berlin. Springer-Verlag, 2010.

[KSW 07] Kapaida, Ravi, Greg Stanley und Mark Walker: *Real World model-based Fault Management.* In: *Proceedings DX-07*, Seiten 306–313, 2007.

[Lang 01] Langer, Michael: *Konzeption und Anwendung einer Customer Service Management Architektur - in German.* Doktorarbeit, March 2001.

[Larm 01] Larman, Craig: *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 2nd Auflage, 2001.

[LeHe 05] Lehner, Franz und Lutz Jürgen Heinricht: *Informationsmanagement: Planung, Überwachung und Steuerung der Informationsinfrastruktur.* Oldenbourg Wissenschaftsverlag, 2005.

[Lewi 95] Lewis, L.: *Managing Computer Networks - A Case-Based Reasoning Approach.* Artech House, Inc., 1995.

[LHA 06] Lopes, Denivaldo, Slimane Hammoudi und Zair Abdelouahab: *Schema Matching in the Context of Model Driven Engineering: From Theory to Practice.* In: *Advances in Systems, Computing Sciences and Software Engineering*, Seiten 219–227. Springer Netherlands, 2006.

[LHBJ 05] Lopes, Denivaldo, Slimane Hammoudi, Jean Bézivin und Frédéric Jouault: *Generating Transformation Definition from Mapping Specification: Application to Web Service Platform.* In: *Advanced Information Systems Engineering*, Band 3520 der Reihe *Lecture Notes in Computer Science*, Seiten 183–192. Springer Berlin / Heidelberg, 2005.

[LHBJ 06] Lopes, Denivaldo, Slimane Hammoudi, Jean Bézivin und Frédéric Jouault: *Mapping Specification in MDA: From Theory to Practice.* In: *Interoperability of Enterprise Software and Applications*, Seiten 253–264. Springer London, 2006.

[LHC 09] LHC Computing Grid: *Memorandum of Understanding for Collaboration in the Deployment and Exploitation of the Worldwide LHC Computing Grid.* Elektronische Quelle URL: `http://lcg.web.cern.ch/LCG/MoU/Brazil/MoU-Brazil-CERN-20Apr09.pdf`, 2009 (letzter Zugriff: 23.12.2009).

[LHdSB 06] Lopes, Denivaldo, Slimane Hammoudi, Jose de Souza und Alan Bontempo: *Metamodel Matching: Experiments and Comparison.* In: *Proceedings of the International Conference on Software Engineering Advances (ICSEA 2006)*, Tahiti, French Polynesia, 2006. IEEE Computer Society.

[Liu 11] Liu, Feng: *Supporting IT Service Fault Recovery with Automated Planning Methods.* Dissertation, Ludwig–Maximilians–Universität München, 2011. to appear.

[Lope 05] Lopes, Denivaldo: *Étude et applications de l'approche MDA pour des plates-formes de Services Web.* Doktorarbeit, Université de Nantes, UFR Sciences et Techniques, Juli 2005.

[LSMK 99] Lück, Ingo, Marcus Schönbach, Arnulf Mester und Heiko Krumm: *Derivation of Backup Service Management Applications from Service and System Models.* In: *DSOM '99: Proceedings of the 10th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management*, Seiten 243–256, London, UK, 1999. Springer-Verlag.

[MaHo 10] Marcu, P. und W. Hommel: *Requirements and concepts for an inter–organizational fault management architecture.* In: *Proceedings of the 9th RoEduNet International Conference*, Sibiu (Hermannstadt), Romania, Juni 2010. IEEE Computer Society.

[MaHo 11] Marcu, P. und W. Hommel: *Inter–organizational fault management: Functional and organizational core aspects of management architectures.* International journal of Computer Networks & Communications (IJCNC), 2011, Volume 3,(Nr. 1):101–117, Januar 2011.

[Malo 87] Malone, T.W.: *Modeling Coordination in Organizations and Markets.* Management Science, 33(10):1317–1332, 1987.

[MaSc 11] Marcu, P. und T. Schaaf: *An information model for inter-organizational fault management.* In: *Proceedings of the 12th IFIP/IEEE Symposium on Integrated Management (IM 2011), IEEE Computer Society Press*, Dublin, Irland, Mai 2011. .

[MCBS 03] Medeiros, Raissa, Walfredo Cirne, Francisco Brasileiro und Jacques Sauvé: *Faults in Grids: Why are they so bad and What can be done about it?* In: *GRID '03: Proceedings of the 4th International Workshop on Grid Computing*, Seite 18, Washington, DC, USA, 2003. IEEE Computer Society.

[MDKP 09] Mitra, Shubhadip, Partha Dutta, Shivkumar Kalyanaraman und Prashant Pradhan: *Spatio-Temporal Patterns for Problem Determination in IT Services.* Seiten 49–56, 2009.

[MiTs 10] Miyazawa, Masanori und Munefumi Tsurusawa: *Fault localization mechanism using integrated resource information model in the next generation network.* In: *NOMS 2000: 12th IEEE/IFIP Network Operations and Management Symposium 'Towards Management of Future Networks and Services'*, April 2010.

[MJTS 07] Mao, Yun, Hani Jamjoom, Shu Tao und Jonathan M. Smith: *Networkmd: topology inference and failure diagnosis in the last mile.* In: *IMC '07: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, Seiten 189–202, New York, NY, USA, 2007. ACM.

[MSFY 11] Marcu, P., D. Schmitz, W. Fritz, M. Yampolskiy und W. Hommel: *Integrated monitoring of multi–domain backbone connections.* International journal of Computer Networks & Communications (IJCNC), 2011, Volume 3,(Nr. 1):82–99, Januar 2011.

[MSGL 09] Marcu, P., L. Shwartz, G. Grabarnik und D. Loewenstern: *Managing Faults in the Service Delivery Process of Service Provider Coalitions.* In: *IEEE International Conference on Service Computing (SCC 2009)*, Bangalore, India, 2009. .

[MSGL 09a] Marcu, P., L. Shwartz, G. Grabarnik und D. Loewenstern: *Incident Correlation Method for Heterarchical Service Delivery.* In: *International Conference on Service Science, INFORMS*, Hong Kong, August 2009. .

[MSHT 10] Marcu, P., D. Schmitz, A. Hanemann und S. Trocha: *Monitoring and Visualization of the Large Hadron Collider Optical Private Network.* In: *Proceedings of the 9–th RoEduNet International Conference*, Sibiu (Hermannstadt), Romania, Juni 2010. IEEE Computer Society.

[Musi 09] Musilek, Slavoj: *Best Practices under the framework of IT Service Management and ITIL.* Elektronische Quelle URL: `http://www.parallon.com/n5_Graz_Presentation_SlavojM.pdf`, 2003 (letzter Zugriff: 07.03.2009).

[MWN] *Das Münchner Wissenschaftsnetz (MWN).* http://www.lrz.de/services/-netz/mwn-ueberblick/url.

[NaSe 06] Natu, Maitreya und Adarshpal S. Sethi: *Active Probing Appoach for Fault Localization in Computer Networks.* In: *Proceedings of 4th IEEE/IFIP Workshop on End-to-End Monitoring Techniques and Services (E2EMON 2006)*, Seiten 301–314, Vancouver, Canada, April 2006. IEEE Press.

[NaSe 08] Natu, Maitreya und Adarshpal S. Sethi: *Using temporal correlation for fault localization in dynamically changing networks.* International Journal of Network Management, 18(4):301–314, 2008.

[Nerb01] Nerb, Michael: *Customer Service Management als Basis für interorganisationales Dienstmanagement.* Dissertation, Technische Universität München, 2001.

[NMMetr] *NMWG – Measurements Schema.* http://anonsvn.internet2.edu/svn/nmwg/trunk/nmwg/schema/rnc/.

[NMTop] *NMWG – Topology Base Schema.* http://anonsvn.internet2.edu/svn/nmwg/trunk/nmwg/schema/rnc/topo/nmtopo_base.rnc.

[NMWG] *Network Measurements Working Group (NMWG).* http://www-didc.lbl.gov/NMWG.

[NRS+ 10] Nayak, Tapan, Ramani Routray, Aameek Singh, Sandeep Uttamchandani und Akshat Verma: *End-to-End Disaster Recovery Planning: From Art to Science.* In: *NOMS 2000: 12th IEEE/IFIP Network Operations and Management Symposium 'Towards Management of Future Networks and Services'*, April 2010.

[OGC 00] OGC (Office of Government Commerce) (Herausgeber): *Service Support.* IT Infrastructure Library (ITIL). The Stationary Office, Norwich, UK, 2000.

[OGC 01] OGC (Office of Government Commerce) (Herausgeber): *Service Delivery.* IT Infrastructure Library (ITIL). The Stationary Office, Norwich, UK, 2001.

[OGC 07a] OGC (Office of Government Commerce) (Herausgeber): *Service Strategy*. IT Infrastructure Library v3 (ITIL v3). The Stationary Office, Norwich, UK, 2007.

[OGC 07b] OGC (Office of Government Commerce) (Herausgeber): *Service Design*. IT Infrastructure Library v3 (ITIL v3). The Stationary Office, Norwich, UK, 2007.

[OGC 07c] OGC (Office of Government Commerce) (Herausgeber): *Service Transition*. IT Infrastructure Library v3 (ITIL v3). The Stationary Office, Norwich, UK, 2007.

[OGC 07d] OGC (Herausgeber): *Service Operation*. IT Infrastructure Library v3 (ITIL v3). The Stationary Office, Norwich, UK, 2007.

[OGC 07e] OGC (Office of Government Commerce) (Herausgeber): *Continual Service Improvement*. IT Infrastructure Library v3 (ITIL v3). The Stationary Office, Norwich, UK, 2007.

[OGP 03] Oppenheimer, David, Archana Ganapathi und David A. Patterson: *Why do internet services fail, and what can be done about it?* In: *USITS'03: Proceedings of the 4th conference on USENIX Symposium on Internet Technologies and Systems*, Berkeley, CA, USA, 2003. USENIX Association.

[OMG 01] *Model Driven Architecture*, Jul 2001.

[OMG 03] *MDA Guide*. http://www.omg.org/mda/, Jun 2003.

[OSI 89] *Information processing systems - Open Systems Interconnection - Basic Reference Model - Part 4: Management framework*. Technischer Bericht ISO/IEC 7498-4, ISO/IEC, 1989.

[OSI 94] *Information technology - Open Systems Interconnection - Basic Reference Model: The Basic Model*. Technischer Bericht ISO/IEC 7498-4, ISO/IEC, 1994.

[PaHa 07] Paradis, Lilia und Qi Han: *A Survey of Fault Management in Wireless Sensor Networks*. J. Netw. Syst. Manage., 15(2):171–190, 2007.

[PAMM 98] Paxson, V., G. Almes, J. Mahdavi und M. Mathis: *Framework for IP Performance Metrics*. Technischer Bericht, USA, 1998.

[PDPT 10] Pouyllau, H, R Douville, V Piperaud und R Theillaud: *Architecture for Inter-Domain Service Delivery*. In: *NOMS 2000: 12th IEEE/IFIP Network Operations and Management Symposium 'Towards Management of Future Networks and Services'*, April 2010.

[PeMe 06] Petrasch, Roland und Oliver Meimberg: *Model Driven Architecture Eine praxisorientierte Einfhrung in die MDA*. dpunkt.verlag, 2006.

[per 08] *perfSONAR MDM release 3.0 - Product Brief*, 2008.

[PrSt 09] Prieto, Alberto Gonzalez und Rolf Stadler: *Adaptive real-time monitoring for large-scale networked systems.* In: *IM'09: Proceedings of the 11th IFIP/IEEE international conference on Symposium on Integrated Network Management*, Seiten 790–795, Piscataway, NJ, USA, 2009. IEEE Press.

[QBRZ 05] Qiu, Lili, Paramvir Bahl, Ananth Rao und Lidong Zhou: *Troubleshooting multihop wireless networks.* In: *SIGMETRICS '05: Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, Seiten 380–381, New York, NY, USA, 2005. ACM.

[RBM 06] Rahman, H.A., K. Beznosov und J.R. Marti: *Identification of sources of failures and their propagation in critical infrastructures from 12 years of public failure reports.* In: *Proceedings of the Third International Conference on Critical Infrastructures (CRIS06)*, Alexandria, VA, USA, 2006. .

[RBM 09] Rahman, H.A., K. Beznosov und J.R. Marti: *Identification of sources of failures and their propagation in critical infrastructures from 12 years of public failure reports.* Critical Infrastructures, 5(3), 2009.

[RBO+ 04] Rish, I., M. Brodie, N. Odintsova, S. Ma und G. Grabarnik: *Real-time Problem Determination in Distributed Systems Using Active Probing.* In: *Proceedings of the 9th IFIP/IEEE International Network Management and Operations Symposium (NOMS 2004)*, Seiten 133–146, Seoul, Korea, April 2004. .

[RFC 2679] Almes, G. and Kalidindi, S. and Zekauskas, M.: *A One-way Delay Metric for IPPM.* Technischer Bericht, USA, 1999.

[RFC 2680] Almes, G. and Kalidindi, S. and Zekauskas, M.: *A One-way Packet Loss Metric for IPPM.* Technischer Bericht, USA, 1999.

[RFC 3393] Demichelis, C. and Chimento, P.: *IP Packet Delay Variation Metric for IP Performance Metrics (IPPM).* Technischer Bericht, USA, 2002.

[RMJW 09] Reidemeister, Thomas, Mohammad Ahmad Munawar, Miao Jiang und Paul A. S. Ward: *Diagnosis of recurrent faults using log files.* In: *CASCON '09: Proceedings of the 2009 Conference of the Center for Advanced Studies on Collaborative Research*, Seiten 12–23, New York, NY, USA, 2009. ACM.

[RMJW 10] Reidemeister, Thomas, Mohammad Ahmad Munawar, Miao Jiang und Paul A. S. Ward: *Identifying Symptoms of Recurrent Faults in Log Files of Distributed Information Systems.* In: *NOMS 2000: 12th IEEE/IFIP Network Operations and Management Symposium 'Towards Management of Future Networks and Services'*, April 2010.

[Scha 08] Schaaf, Thomas: *IT–gestütztes Service–Level–Management — Anforderungen und Spezifikation einer Managementarchitektur.* Dissertation, Ludwig–Maximilians–Universität München, Dezember 2008.

[Scha 05] Schauerhammer, Karin: *X-WiN als Netzressource im Grid.* DFN Mitteilungen, 69(November):16–19, 2005.

[Schif 07] Schiffers, Michael: *Management dynamischer Virtueller Organisationen in Grids.* Dissertation, Ludwig–Maximilians–Universität München, Juli 2007.

[Schif 11] Schiffers, M.: *Verlsslichkeitsaspekte in Grids.* Habilitation, Ludwig–Maximilians–Universität München, 2011. to appear.

[Schm 08] Schmitz, David: *Automated Service-Oriented Impact Analysis and Recovery Alternative Selection.* Dissertation, Ludwig–Maximilians–Universität München, Juli 2008.

[ScZa 00] Schwartz, S. H. und D Zager: *Value-oriented network management.* In: *NOMS 2000: IEEE/IFIP Network Operations and Management Symposium 'The Networked Planet: Management Beyond 2000'*, Seiten 715–728, Honolulu, Hawaii, USA, April 2000. .

[SH 10] S. Hammoudi, W. Alouini, D. Lopes M. Huchard: *Towards A Semi-Automatic Transformation Process in MDA: Architecture, Methodology and First Experiments*, Seiten 48–76. 2010.

[SLCA 09] Sousa, José, Denivaldo Lopes, Daniela Barreiro Claro und Zair Abdelouahab: *A Step Forward in Semi-automatic Metamodel Matching: Algorithms and Tool.* In: *Enterprise Information Systems*, Band 24 der Reihe *Lecture Notes in Business Information Processing*, Seiten 137–148. Springer Berlin Heidelberg, 2009.

[SSS 09] Song, Yang, Anca Sailer und Hidayatullah Shaikh: *Problem classification method to enhance the ITIL incident and problem.* In: *IM'09: Proceedings of the 11th IFIP/IEEE international conference on Symposium on Integrated Network Management*, Seiten 295–298, Piscataway, NJ, USA, 2009. IEEE Press.

[StSe 04a] Steinder, M. und A. Sethi: *A Survey of Fault Localization Techniques in Computer Networks.* Science of Computer Programming, Elsevier, 53(1):165–194, 2004.

[StSe 04b] Steinder, Malgorzata und Adarshpal S. Sethi: *Multi-domain Diagnosis of End-to-End Service Failures in Hierarchically Routed Networks.* In: *NETWORKING*, Seiten 1036–1046. IFIP, 2004.

[StSe 04c] Steinder, Malgorzata und Adarshpal S. Sethi: *Probabilistic fault localization in communication systems through incremental hypothesis updating.* Computer Networks, 45(4):537–562, 2004.

[StSe 04d] Steinder, Malgorzata und Adarshpal S. Sethi: *Probabilistic fault localization in communication systems using belief networks.* IEEE/ACM Transactions on Networking (TON), 12(5):809–822, 2004.

[StSe 07] Steinder, Malgorzata und Adarshpal S. Sethi: *Multidomain Diagnosis of End-to-End Service Failures in Hierarchically Routed Networks.* IEEE Trans. Parallel Distrib. Syst., 18(3):379–392, 2007.

[StVa 98] Stanley, Gregory und Ramesh Vaidhyanathan: *A Generic Fault Propagation Modeling Approach to On-line Diagnosis and Event Correlation.* In: *3rd IFAC Workshop on On-line Fault Detection and Supervision in the Chemical Process Industries*, Solaize, France, Juni 1998. .

[SZG 03] Swany, Martin, Jason Zurawski und Dan Gunter: *NMWG- Schema Developers Guide.* http://www.cis.udel.edu/ swany/nmwg/devguide-05-dec-04.pdf, 2003.

[TaAS 08] Tang, Yongning und Ehab Al-Shaer: *Towards Collaborative User-Level Overlay Fault Diagnosis.* In: *INFOCOM 2008: proceeding of the 27th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies*, Seiten 2476–2484, Phoenix, AZ, USA, apr 2008. IEEE.

[TASB 05] Tang, Yongning, Ehab S. Al-Shaer und Raouf Boutaba: *Active Integrated Fault Localization in Communication Networks.* In: *Proceedings of the 9th IFIP/IEEE International Symposium on Integrated Network Management (IM 2005)*, Seiten 543–556, Nice, France, May 2005. .

[TASZ 07] Tang, Yongning, Ehab S. Al-Shaer und Bin Zhang: *Toward Globally Optimal Event Monitoring & Aggregation For Large-scale Overlay Networks.* In: *IM 2007: Proceedings of the 10th IFIP/IEEE International Symposium on Integrated Network Management*, Piscataway, NJ, USA, 2007. IEEE Press.

[Uebe 09] Ueberhorst, Stefan: *ITIL v3 ist für viele noch ein zu heißes Eisen.* Elektronische Quelle URL: `http://www.computerwoche.de/knowledge_center/software_infrastruktur/1875989/#`, 2008 (letzter Zugriff: 01.03.2009).

[UlSc 06] Ullmann, Klaus und Karin Schauerhammer: *Operational Model for E2E links in the NREN/GÉANT2 and NREN/Cross-Border-Fibre supplied optical platform.* Technischer Bericht, Géant2, 2006.

[Vahs 05] Vahs, Dietmar: *Organisation. Einfhrung die Organisationstheorie und praxis.* Schffer-Poeschel, Stuttgart, 2005.

[VPM+ 07] Vianna, Ricardo Lemos, Everton Rafael Polina, Clarissa Cassales Marquezan, Leandro Bertholdo, Liane Margarida Rockenbach Tarouco, Maria Janilce Bosquiroli Almeida und Lisandro Zambenedetti Granville: *An Evaluation of Service Composition Technologies Applied to Network Management.* In: *10th IFIP/IEEE International Symposium on Integrated Network Management*, Seiten 420–428, Munich, 2007. .

[VRYK 03a] Venkatasubramanian, Venkat, Raghunathan Rengaswamy, Kewen Yin und Surya N. Kavuri: *A review of process fault detection and diagnosis: Part I: Quantitative model-based methods.* Computers & Chemical Engineering, 27(3):293 – 311, 2003.

[VRYK 03b] Venkatasubramanian, Venkat, Raghunathan Rengaswamy, Kewen Yin und Surya N. Kavuri: *A review of process fault detection and diagnosis: Part II: Qualitative models and search strategies.* Computers & Chemical Engineering, 27(3):313 – 326, 2003.

[VRYK 03c] Venkatasubramanian, Venkat, Raghunathan Rengaswamy, Kewen Yin und Surya N. Kavuri: *A review of process fault detection and diagnosis: Part III: Process history based methods.* Computers & Chemical Engineering, 27(3):327 – 346, 2003.

[W3C 01] W3C: *Web Descriprion Language 1.1 (WSDL)*, 2001.

[W3C 07] W3C: *Web Descriprion Language 2.0 (WSDL)*, 2007.

[XuDe 05] Xu, Peng und Ralph Deters: *Fault-Management for Multi-Agent Systems.* In: *SAINT '05: Proceedings of the The 2005 Symposium on Applications and the Internet*, Seiten 287–293, Washington, DC, USA, 2005. IEEE Computer Society.

[Yamp 09] Yampolskiy, Mark: *Maßnahmen zur Sicherung von E2E-QoS bei Verketteten Dienste.* Dissertation, Ludwig–Maximilians–Universität München, Oktober 2009.

Index