

CIM/CORBA-basiertes Management verteilter kooperierender Managementsysteme

Alexander Keller

*IBM Research Division, T.J. Watson Research Center
P.O. Box 704, Yorktown Heights, NY, USA
Telephone: (914) 784 7593, Telefax: (914) 784 6183
alexk@us.ibm.com*

Zusammenfassung Web-basierte Technologien wie Portale und e-business Applikationen ermöglichen Unternehmen in zunehmendem Umfang das Outsourcing von IT-Diensten an sog. Application Service Provider (ASP), die ihrerseits Netzdienste von Internet Service Providern (ISP) in Anspruch nehmen. Die daraus resultierende Schichtung von Dienstbringer/Dienstnutzer-Beziehungen bei der Bereitstellung komplexer Anwendungsdienste erschwert jedoch die Lokalisierung und Identifikation der Ursache beim Auftreten von Fehlern. Dies liegt nicht zuletzt daran, daß der Austausch von Managementinformation zwischen den Managementsystemen der Dienstbringer gegenwärtig nur unzureichend funktioniert. Der Kern dieses Problems ist, daß derzeit keine offengelegten Schnittstellen existieren, die einem Dienstanutzer Zugriff auf die Managementsysteme eines Dienstbringers gestatten. Naturgemäß handelt es sich bei solchen *kooperierenden Managementsystemen* um verteilte Anwendungen, die wiederum ihrerseits administriert werden müssen. Es ist daher notwendig, geeignete Managementinformation und -dienste zu definieren, um Managementsysteme in heterogener Umgebung überwachen und steuern zu können.

Der vorliegende Beitrag stellt einen neuen Ansatz zur Lösung dieser Problematik vor, der auf einem Objektmodell zur Instrumentierung verteilter kooperierender Managementsysteme beruht. Dieses Objektmodell wurde unter Berücksichtigung existierender Standards für den Entwurf verteilter Anwendungen (*Reference Model of Open Distributed Processing, RM-ODP*) sowie deren Management (*Common Information Model, CIM*) erstellt und erweitert diese um die spezifischen Eigenschaften von Managementsystemen. Eine CORBA/Java-basierte Implementierung weist die praktische Anwendbarkeit der vorgestellten Konzepte nach.

1 Einführung und Motivation

Die Frage "Was sind die Charakteristiken verteilter Anwendungen aus Sicht des integrierten Managements?" ist derzeit Gegenstand mehrerer Untersuchungen [17, 4, 3]. Im Vergleich zu anderen Managementbereichen (wie zum Beispiel dem Netz- und Systemmanagement) besteht jedoch beim sog. Anwendungsmanagement Unklarheit, wie ein geeignetes Managementmodell der statischen und dynamischen Aspekte verteilter Anwendungen auszusehen hat.

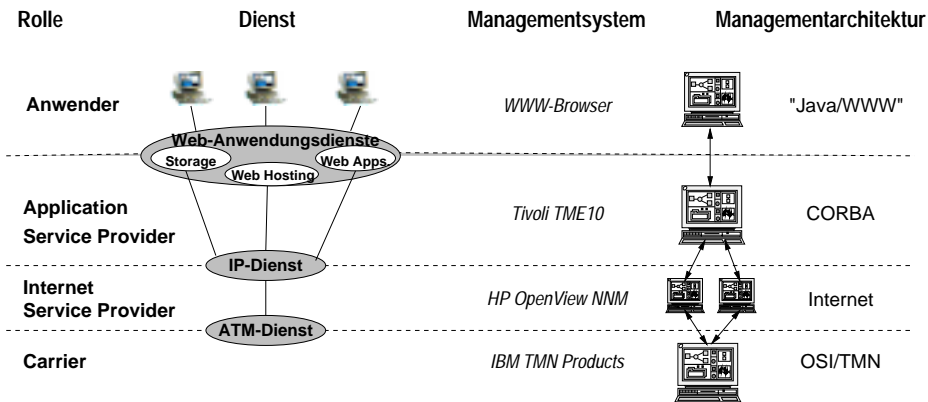


Abbildung 1. Outsourcing von IT-Diensten erfordert die Kooperation von Managementsystemen

Diese Problematik wird durch neuartige web-basierte Technologien wie e-business Applikationen und Portale verschärft, da diese eine wesentlich höhere Flexibilität und Dynamik als traditionelle Client/Server-Architekturen besitzen: Neben technischen Vorteilen wie dem lastabhängigen Zuschalten weiterer Serverknoten in Spitzenzeiten bietet sich Unternehmen insbesondere die Möglichkeit, weitreichendes Outsourcing von IT-Diensten an sog. *Application Service Provider (ASP)* vorzunehmen, um Kostensenkungspotentiale auszuschöpfen. Demgegenüber hat ein ASP (hierzu zählen ebenfalls *Storage Service Provider*, die Speicherkapazität bereitstellen) die Möglichkeit, dieselbe Systeminfrastruktur für zahlreiche Kunden zu verwenden und nutzt seinerseits die entstehenden Synergieeffekte. Insbesondere vollzieht sich Outsourcing von IT-Diensten auf weiteren Ebenen, wie Abbildung 1 verdeutlicht: Ein ASP macht von einem ISP Gebrauch, um IP-Konnektivität zu erhalten, während dieser die Leitungen (bzw. Netzdienste wie ATM oder Frame Relay) eines Telekom-Carriers anmietet. Man erhält somit geschichtete Dienstbringer/Dienstnutzerbeziehungen, an deren Schnittstellen jeweils Dienstgüterevereinbarungen (*Service-level Agreements, SLA*) [19] getroffen werden.

Um SLAs jedoch überwachen und durchsetzen zu können, ist es notwendig, daß die Managementsysteme der beteiligten Provider in der Lage sind, Informationen auszutauschen. Wie der rechte Teil von Abbildung 1 verdeutlicht, ist dies jedoch keineswegs selbstverständlich: Neben dem Problem, daß das Zusammenwirken der Managementsysteme unterschiedlicher Hersteller dadurch erschwert wird, daß diese in der Regel auf verschiedenen *Managementarchitekturen* basieren¹, existieren insbesondere keinerlei Festlegungen, welche Art von Information Managementsysteme bereitstellen bzw. welche Dienste diese anbieten sollen. Der vorliegende Beitrag behandelt diese Fragestellung und stellt ein Objektmodell für verteilte kooperierende Managementsysteme vor. Es spezifiziert einen Mindestumfang an Instrumentierung,

¹ Aus Platzgründen ist in diesem Beitrag eine eingehende Behandlung der Problematik einer *Integration von Managementarchitekturen* mittels Management-Gateways nicht möglich; wir verweisen hierzu auf [15, 16, 13], die sich dieser Thematik eingehend widmen.

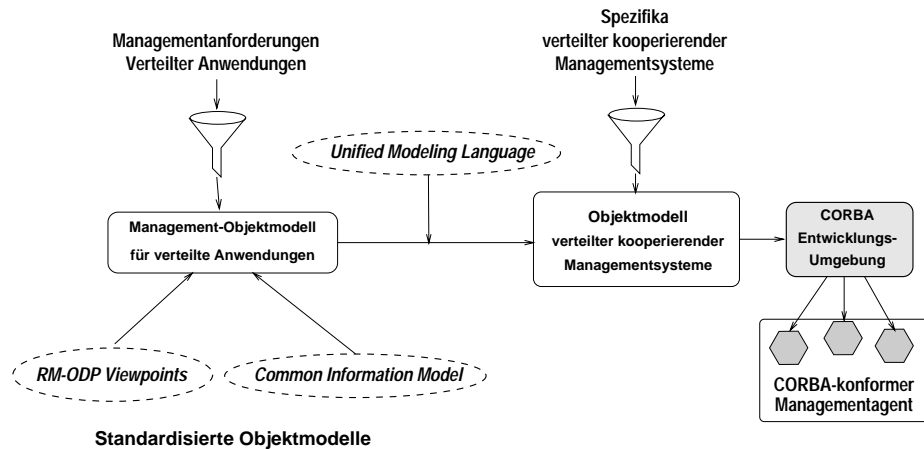


Abbildung 2. Überblick über das Vorgehensmodell des Beitrags

der für die Überwachung und Steuerung von Managementsystemen erforderlich ist. Dies schließt sowohl das Managementsystem selbst (als administrierte Ressource) ein, als auch den einheitlichen Zugriff auf in der Datenbank des Managementsystems gespeicherte Information über von ihm verwaltete Ressourcen (wie z.B. Systeme, Netzkomponenten oder Applikationen).

Das Vorgehensmodell dieses Beitrags ist in Abbildung 2 dargestellt: Wir werden zunächst in Abschnitt 2 die Anforderungen an das Management verteilter kooperierender Managementsysteme vorstellen, die wir anhand mehrerer Szenarien in der Praxis gewonnen haben. Diese Analyse wird uns in Abschnitt 3 helfen, existierende Ansätze im Bereich des Anwendungsmanagements kritisch zu bewerten; besonderes Augenmerk widmen wir hierbei dem ODP-Referenzmodell der ISO sowie dem *Common Information Model (CIM)* der DMTF, da sich diese beiden Ansätze besonders gut für unsere Aufgabenstellung eignen. Wir haben diese beiden Standards als Grundlage unseres Modells gewählt und um die Spezifika verteilter kooperierender Managementsysteme erweitert. Das auf CIM und ODP basierende Modell wird in Abschnitt 4 vorgestellt, wobei wir insbesondere auf die neu definierten Objektklassen und Assoziationen eingehen. Anschließend stellen wir unsere CORBA/Java-basierte prototypische Implementierung vor. Der Beitrag schließt mit einer Zusammenfassung unserer Erfahrungen und zeigt gegenwärtig untersuchte Folgefragestellungen auf.

2 Anforderungsanalyse

Wir befassen uns in diesem Abschnitt mit der Analyse der Spezifika verteilter kooperierender Managementsysteme und leiten daraus die Anforderungen an das in Abschnitt 4 beschriebene Objektmodell ab. Hierbei unterscheiden wir zwischen der Managementinformation d.h. der Datenmenge zur umfassenden Beschreibung eines Systems (Abschnitt 2.1), sowie den Managementdiensten, also dem Funktionsumfang, der für dessen Management erforderlich ist (Abschnitt 2.2).

2.1 Managementinformation

Wie bei anderen Ressourcenarten, so ist auch für Managementsysteme ein Mindestumfang an generischen Konfigurationsdaten (Hersteller, Produktname und -version, Installationsdatum usw.) erforderlich, sowie Attribute für das Fehlermanagement (Supportangaben, Zeit seit der letzten Initialisierung, operationeller und administrativer Zustand). Ebenfalls sollten detaillierte Informationen zu den Teilkomponenten des Systems (z.B. Ereignisverwaltung, Topologiemanager, Datenbank), deren Zustände und deren (Abhängigkeits-) Beziehungen zueinander vorliegen. Eine Zuordnung der Teilkomponenten zu Prozeßnamen und eine Möglichkeit für den individuellen Neustart einzelner Komponenten sollte ebenfalls vorhanden sein.

Relevante Parameter für das Abrechnungsmanagement sind: die Art der Lizenz (nodelocked, floating, Campus) und Zähler für die maximale sowie die aktuelle Anzahl simultaner Benutzer. Zahlreiche kommerzielle Managementsysteme werden mit Lizenzservern ausgeliefert, die diese Informationen bereitstellen. Für den Betrieb des Managementsystems ist die Funktionsfähigkeit seines Lizenzservers maßgeblich.

Das Sicherheitsmanagement spielt eine wesentliche Rolle in verteilten Umgebungen, da Managementsysteme verschiedener Dienstleister kooperieren müssen: Jedes Managementsystem muß Auskunft über seine administrative Domäne sowie die von ihm verwalteten Agenten, untergeordneten Managementsysteme und Gateways geben können. Ferner müssen geeignete Mechanismen zum Schutz gegen unbefugten Zugriff sowie eine Administrationsmöglichkeit für autorisierte Interaktionen mit Partnersystemen vorhanden sein.

Informationen für das Leistungsmanagement beinhalten Parameter zur Konfiguration von Caching-Mechanismen für häufig benötigte Managementdaten (wie z.B. die Cachegröße oder die Gültigkeitszeiträume von Ressourcendaten). Zähler zur Messung der Bearbeitungs- und Antwortzeiten sowie zur Ermittlung der Auslastung oder der Anzahl der Anfragen über benutzerdefinierte Zeiträume liefern gute Hinweise über vorhandene Engpässe.

2.2 Managementdienste

Dienste für das Konfigurationsmanagement haben zum Ziel, neue Systeme problemlos in eine bestehende Managementinfrastruktur einzubringen, indem diesen Konfigurationsprofile anhand ihrer Domänenzugehörigkeit bzw. Funktionsumfang dynamisch zugewiesen werden. Hierzu muß ein Managementsystem die Definition von Konfigurationsprofilen gestatten, die u.U. an untergeordnete Systeme verteilt werden. Es muß außerdem in der Lage sein, selbst solche Profile zu erhalten und zu verarbeiten.

Große praktische Relevanz für das Fehlermanagement haben Operationen zur Überprüfung des Zustands einzelner Komponenten eines Managementsystems sowie Dienste zum Starten, Stoppen und Wiederaufsetzen nach Auftreten eines Fehlers. Maßnahmen zur Überprüfung der Ressourcendatenbank auf Konsistenz sowie das Ausführen diverser Administrations- und Diagnoseprozeduren sollten in regelmäßigen Intervallen erfolgen. Beim Auftreten von schweren Fehlern sind die Ereignis- und Fehlerlogs eines Managementsystems eine wichtige Informationsquelle;

einheitliche Mechanismen zum Betrachten, Auswerten und Durchsuchen hinsichtlich frei definierbarer Kriterien sind ebenfalls von hoher Bedeutung.

Die Konfiguration von Diensten zur Ereignismeldung, die Initiierung von Managementoperationen auf Partnersystemen und die Registrierung für spezielle Ereignistypen erfordert das Vorhandensein geeigneter Sicherheitsdienste und -policies.

Leistungs- und Abrechnungsdienste wie z.B. das Generieren von Berichten über die Nutzung von Ressourcen (Statistiken, Trends, nach diversen Kriterien wie z.B. Durchsatz, Verzögerung oder Last aufbereitete Systemdaten) stellen eine wesentliche Planungsgrundlage dar. Die Identifikation relevanter Systeme und Parameter sowie die Kombination der obengenannten Basisgrößen werden aus den spezifischen Gegebenheiten und Zielvorgaben einer Unternehmung abgeleitet. Folglich müssen Mechanismen bereitgestellt werden, die eine flexible Kombination der Meßgrößen sicherstellen, um aus Ressourcendaten aussagekräftige Managementinformation zu gewinnen.

3 Existierende Ansätze für das Anwendungsmanagement

Die ersten Ansätze für das Management verteilter Anwendungen stammen von der Internet Engineering Task Force: Die Spezifikationen der IETF Application MIBs (*applMib* [11] und *sysAppMib* [14]) sind jedoch auf die Gegebenheiten eines speziellen Betriebssystems zugeschnitten² und die Problematik des objektbasierten IETF-Ansatzes beeinträchtigt deren Anwendbarkeit: Ein beträchtlicher Anteil der in der Application MIB definierten Tabellen dient lediglich dazu, die Managementinformation nach unterschiedlichen Kriterien zu indizieren. Ferner ist anzumerken, daß sich beide MIBs ausschließlich auf die passive Überwachung (Monitoring) von lokal installierten Anwendungen beschränken. Weitere Ansätze für das Management verteilter Anwendungen sind nachstehend aufgeführt.

Der IEEE 1387.2 *POSIX Software Administration Standard* [7] definiert Managementinformation, die auf Softwareverteilung und -installation zugeschnitten ist. Es handelt sich hierbei um statische Information, die die Struktur von Softwarepaketen (in einem UNIX-Umfeld) beschreibt.

Die *Distributed Software Administration (XDSA)* Spezifikation der Open Group [20] erweitert IEEE 1387.2 und verfügt daher ebenfalls über keinerlei Managementinformation, die sich etwa mit den dynamischen bzw. Laufzeitaspekten einer Anwendung befaßt. Vielmehr bestehen die Erweiterungen hauptsächlich darin, diverse Befehle (*swinstall*, *swlist*, *swmodify* etc.) sowie ein Beschreibungsformat für Installations-Images festzulegen.

Die *Tivoli Application Management Specification (AMS)* [1] definiert allgemeingültige Objektklassen für das Management verteilter Anwendungen sowie deren Beziehungen zur Systeminfrastruktur und kann als Vorläufer von CIM (siehe Abschnitt 3.2) angesehen werden.

Das *Application Response Measurement (ARM) API* [2] und seine Erweiterungen [10] haben zum Ziel, Antwortzeitmessungen von Transaktionen durchzuführen, um

² die Übersicht der laufenden Anwendungen entspricht der UNIX-Prozeßtabelle und die darin enthaltenen Informationen können somit nicht auf jeder Systemplattform ermittelt werden.

Aussagen über die Bearbeitungszeit von Aufträgen (und damit das Leistungsvermögen der Anwendungen) zu treffen. Grundlage hierfür ist, daß die betreffende Anwendung eine ARM-Schnittstelle anbietet, was jedoch in der Praxis nur selten der Fall ist.

Es ist festzuhalten, daß die existierenden Ansätze hauptsächlich auf die statischen Aspekte verteilter Anwendungen abstellen und hier einige Überschneidungen bezüglich der bereitgestellten Informationsmenge bestehen. Demgegenüber wird dem *dynamischen* Verhalten verteilter Anwendungen kaum Aufmerksamkeit gewidmet, obwohl dies – wie die Ausführungen in Abschnitt 1 gezeigt haben – von ausschlaggebender Bedeutung ist.

3.1 Das ISO Reference Model of Open Distributed Processing (RM-ODP)

ODP [8] definiert generische Begriffe, Konzepte und Regeln zur Beschreibung verteilter Anwendungen. Obwohl ODP nicht explizit auf das Management dieser Anwendungen eingeht, so werden doch Aussagen bezüglich der allgemeingültigen Eigenschaften verteilter Anwendungen getroffen; man kann davon ausgehen, daß die Struktur beliebiger verteilter Anwendungen den Festlegungen des ODP folgt und ODP somit einen generischen Minimalumfang an Informationen über verteilte Anwendungen bereitstellt, der auch für deren Management genutzt werden kann.

Unsere Analyse der ODP-Konzepte nach Managementgesichtspunkten ergibt, daß 2 der insgesamt 5 in ODP definierten sog. *Viewpoint Languages* – nämlich die *Computational* und *Engineering Languages* – wichtig sind, da sie diejenigen Gegebenheiten verteilter Anwendungen beschreiben, die besondere Relevanz für das Management besitzen. Konzepte, die in der *Information Language* festgelegt werden, beziehen sich auf semantische Eigenschaften des Informationsaustauschs und sind nicht unmittelbar für (technisches) Management relevant. Da eines der Ziele integrierten Managements darin besteht, von den implementierungstechnischen Spezifika einer Ressource zu abstrahieren, ist auch die *Technology Language* für unsere Zwecke schlecht geeignet. Die Verwendung der *Enterprise Language* gestattet die Definition von unternehmensbezogenen Kriterien wie Rollen, Aufgaben und Zuständigkeiten, die jedoch für die hier untersuchte Fragestellung zu abstrakt sind.

Die Konzepte der **Computational Language** gestatten die funktionale Zerlegung eines Systems in interagierende Objekte. Management-Objektklassen wie *operation*, *signal* oder *binding* sind wichtig, um Beziehungen zwischen Managementobjekten zu beschreiben. Es sei an dieser Stelle festgehalten, daß keine andere (Management)Architektur ähnliche Konzepte anbietet, obwohl diese essentiell zur Ermittlung dynamischer Beziehungen zwischen verteilten Anwendungen sind. Die in der *Computational Language* definierten Objektklassen sind in Abbildung 3 dargestellt.

Die **Engineering Language** definiert Mechanismen und Komponenten für die Interaktion zwischen Objekten in einem verteilten System. Beispiele für aus dieser Sprache abgeleitete Objektklassen sind *node*, *capsule*, *cluster*, *channel*, die für dynamische Aspekte des Anwendungsmanagements hilfreich sind. Die Definition des *capsule*-Konzepts aus [9] verdeutlicht dies: “A configuration of engineering objects forming a single unit for the purpose of encapsulation of processing and storage”; *capsule* entspricht daher einem Prozeß, der innerhalb eines Betriebssystems abläuft.

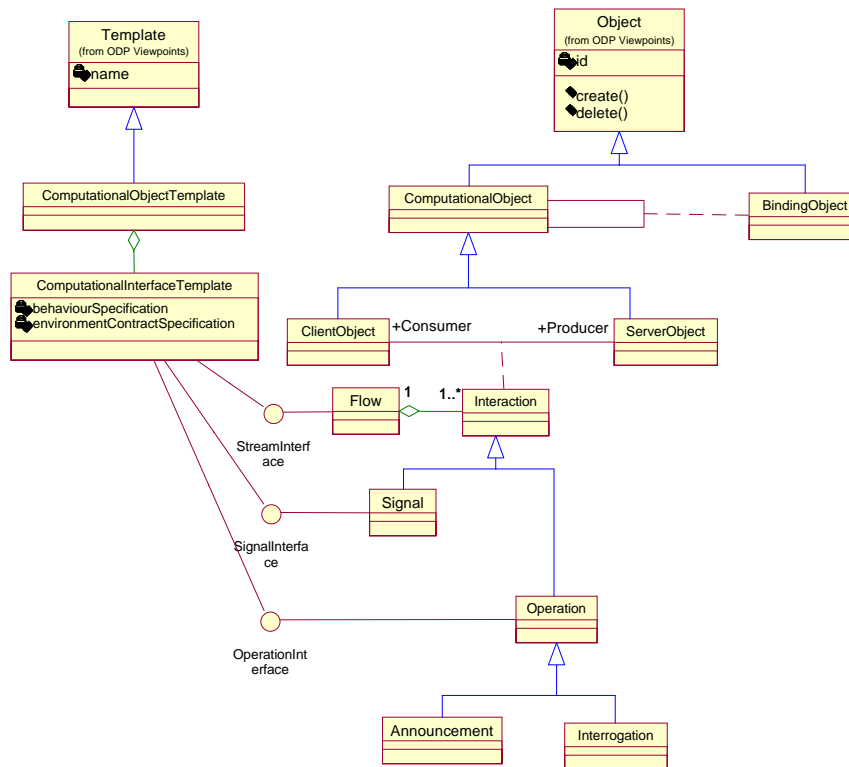


Abbildung 3. Die Objektklassen des ODP Computational Viewpoint

3.2 Das Common Information Model der Distributed Management Task Force

CIM [6] ist ein objektorientierter Ansatz zur Definition von Managementobjekten für beliebige Ressourcen sowie deren Beziehungen zueinander. Zur Vermeidung von Mehrfachvererbung macht CIM weitreichenden Gebrauch von Assoziations- bzw. Aggregationsbeziehungen: Diese Beziehungen werden als Assoziationsklassen modelliert. Das *Core Schema* definiert grundlegende abstrakte Ressourcenklassen wie z.B. *service access point*, *service*, *product*, *system*, *logical device* sowie einen Mechanismus, um einen oder mehrere Parameter einer Ressource zu modifizieren (*setting*). Ferner sind mehrere *Common Schemas* definiert [5]: Das *System Schema* erweitert das Core Schema um Objektklassen für Aufträge, Betriebssysteme, Prozesse, Threads, Dateisysteme, Endsysteme und Cluster. Das *Application Schema* behandelt das Management verteilter Anwendungen und ist für die in diesem Beitrag behandelte Problemstellung besonders relevant: Es definiert Managementobjekte für verteilte Anwendungen (*application system*), deren logische (*software feature*) und physische Komponenten (*software element*). Zusätzlich werden diverse Prüfverfahren (Kompatibilität bzgl. Betriebssystemversion, verfügbarem Speicherplatz usw.) sowie Aktionen (*reboot*, *execute program*) festgelegt. Die Summe

der in den insgesamt 14 CIM Schemas definierten Objektklassen liegt gegenwärtig bei ca. 250. Die von CIM verwendete Notation ist die *Unified Modeling Language (UML)* [18], was die Verwendung am Markt erhältlicher Entwicklungswerkzeuge ermöglicht. Die Abbildung der CIM-Schemas in weit verbreitete Programmier- und Beschreibungssprachen (OMG IDL, C++, Java, XML) ist somit gesichert.

Die Evaluierung bestehender Ansätze hat gezeigt, daß derzeit keine Managementarchitektur in der Lage ist, die vielfältigen Aspekte sowie den Lebenszyklus verteilter Anwendungen auch nur annähernd abzudecken. Von den betrachteten Ansätzen erscheinen CIM und ODP am erfolgversprechendsten, da sich beide ergänzen; wir haben daher diese beiden Ansätze als Basis für unser Objektmodell gewählt.

4 CIM/ODP-basiertes Modell verteilter Managementsysteme

Die Kooperation verteilter Managementsysteme bedingt, daß die Managementinformation sowie die Dienste aller teilnehmenden Managementsysteme offengelegt und damit für Partnersysteme zugreifbar sind. In diesem Abschnitt beschreiben wir das von uns entworfene Objektmodell, welches diese Anforderungen erfüllt. Wir benutzen UML als Notation zur Abbildung der Klassen unseres CIM/ODP-basierten Objektmodells verteilter Managementsysteme auf CORBA, womit wir unsere prototypische Implementierung durchgeführt haben.

4.1 Designüberlegungen bei der Definition der Basisklassen

Die Beschreibung in Abschnitt 3.2 hat verdeutlicht, daß der Umfang an bereits definierter Managementinformation CIM zu einer soliden Grundlage für die Modellierung verteilter Systeme und Anwendungen macht. Problematisch ist jedoch, daß gegenwärtig der Laufzeitaspekt nur unzureichend abgedeckt wird: Dynamische Bindungen, Warteschlangen, Anfragen oder Signale sind noch nicht in CIM enthalten.

Die ODP *Computational Language* hilft, diese Lücke zu füllen, da in ihr exakt diese Begriffe definiert sind (vgl. Abbildung 3). Die ODP *Engineering Language* bietet jedoch keinen Mehrwert gegenüber CIM, da beide Standards sowohl die Bestandteile, als auch die Ausführungsumgebung einer verteilten Anwendung beschreiben (z.B. Threads, Prozesse, Betriebssystem, Endbenutzersystem).

Wir haben daher die Konzepte der ODP *Computational Language* mit den CIM *Core*, *System* und *Application Schemas* kombiniert, um die Basisklassen unseres Objektmodells zu erhalten, die wir anschließend um die Spezifika verteilter kooperierender Managementsysteme erweitern. Somit ist bereits ein beträchtlicher Umfang an wichtiger Information für das Anwendungsmanagement nahe an der Wurzel der Vererbungshierarchie vorhanden: Beschreibungen von Applikationen, deren Diensten und Komponenten sowie ihre Beziehungen zu Betriebssystemprozessen sind bereits an dieser Stelle spezifiziert. Die Tatsache, daß einzelne Komponenten (sowie die gesamte Anwendung) gestartet, angehalten und wiederaufgesetzt werden können findet man hier ebenso wieder, wie Hinweise auf Bindungen zur Laufzeit. Hierbei handelt es sich um Managementinformation, die in traditionellen Managementarchitekturen (wie z.B. dem SNMP-basierten Management) lediglich in Ressourcen-spezifischen Beschreibungen festgelegt ist und somit für jede Anwendung eigens neu definiert werden muß.

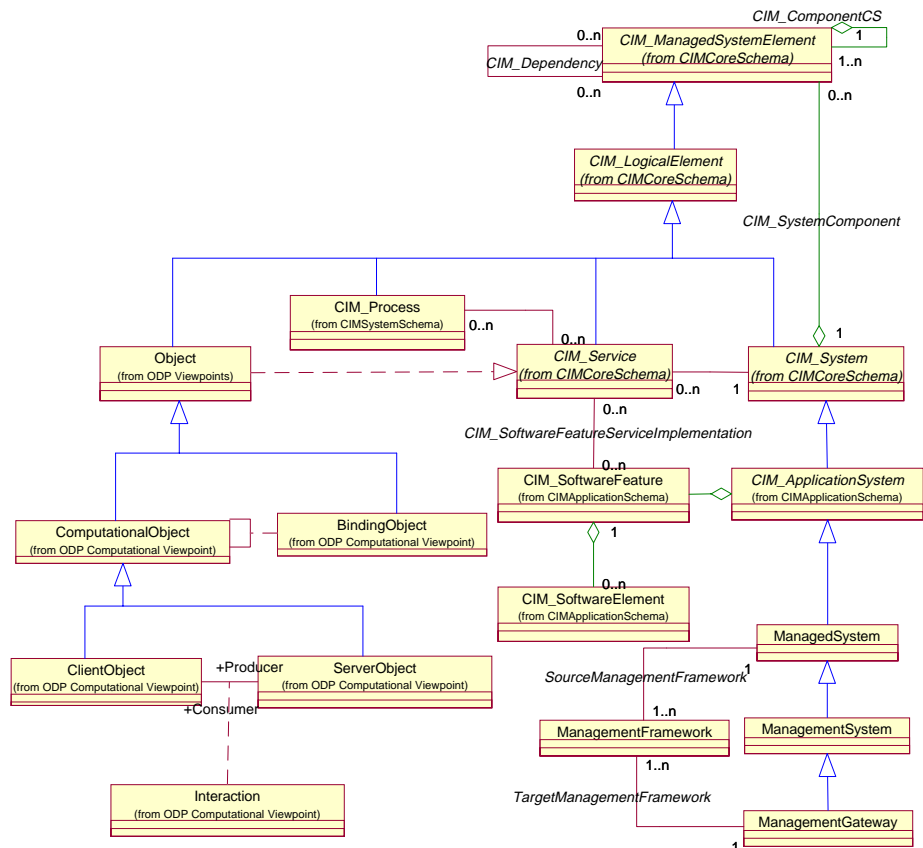


Abbildung 4. CIM/ODP-basiertes Objektmodell verteilter Managementsysteme (Ausschnitt)

4.2 Managementsystem-spezifische Objektklassen

Die 3 relevanten CIM Schemata (Core, System, Application) enthalten ca. 75 Objektklassen und ebensoviele Assoziationen; von ca. einem Drittel dieser Objektklassen haben wir zusätzliche Klassen abgeleitet, um die spezifischen Aspekte von Managementsystemen zu berücksichtigen. Die graphische Darstellung des gesamten Objektmodells ist in diesem Beitrag nicht möglich, weshalb wir in dem in Abbildung 4 dargestellten Klassendiagramm auf folgende Managementinformation verzichten müssen: Die Objektklassen für Ereignis- und Fehlerlogs sowie die von den CIM-Klassen *Check* und *Action* abgeleiteten Objekte, z.B. zur Überprüfung der Systemgegebenheiten. Außerdem entfallen sämtliche Objektklassen mit Bezug zu Produkt- und Bestellinformationen. Ferner konnten wir Attributdefinitionen (Name, Datentyp, Default-Wert, Flags für schreib-/lesbare Attribute, CIM Qualifiers) sowie Methodensignaturen (Argumente, Rückgabewerte) nicht übernehmen, da diese die Lesbarkeit der Abbildung stark beeinträchtigt hätten. Der in Abbildung 4 dargestellte Ausschnitt des Objektmodells illustriert jedoch die wesentlichen Überlegungen.

Zunächst muß ausgedrückt werden, daß Managementsysteme selbst verteilte Anwendungen sind, die ihrerseits überwacht und gesteuert werden müssen. Hierzu leiten wir von der (abstrakten) CIM-Objektklasse *ApplicationSystem* eine neue Klasse *ManagedSystem* ab, die sämtliche zu managenden Objekte repräsentiert. Diese ist mit *ManagementFramework* assoziiert, um die Spezifika der Managementarchitektur (Name, Version, Protokoll, Namenskonventionen und weitere Eigenschaften) zusammenzufassen, denen die Ressource unterliegt. Anschließend leiten wir von *ManagedSystem* die Klasse *ManagementSystem* ab, was den obengenannten Sachverhalt ausdrückt. Ein Management-Gateway ist ein Managementsystem, das zusätzlich eine Umsetzung zwischen unterschiedlichen Architekturen (Informationsmodelle, Protokolle) vornimmt; dies ist der Grund für die Assoziation *TargetManagementFramework*, die *ManagementGateway* – zusätzlich zur von *ManagedSystem* geerbten Assoziation *SourceManagementFramework* – mit *ManagementFramework* verbindet.

Die Tatsache, daß ein Managementsystem aus mehreren logischen Komponenten (Topologiemanager, Zustandsmonitor etc.) besteht, die ihrerseits in Form unterschiedlicher Installationseinheiten implementiert sind, wird durch die Enthaltenseinsbeziehungen zwischen den entsprechenden CIM-Klassen *ApplicationSystem*, *SoftwareFeature* und *SoftwareElement* ausgedrückt. Ferner implementiert ein *SoftwareFeature* einen oder mehrere Dienste (*Service*), was mit der Assoziation *SoftwareFeatureServiceImplementation* verdeutlicht wird. Ein oder mehrere instantiierte Dienste wiederum können in Form eines Prozesses (*Process*) ablaufen, was neben herkömmlichen, prozeßbasierten Applikationen auch insbesondere nach neuen Programmierparadigmen (EJBs, Java Servlets, virtuelle Maschinen, CORBA Components) implementierte Software einschließt.

Besonders hilfreich zur Darstellung des Sachverhalts, daß ein (Management-) System eine beliebige Anzahl unterschiedlicher Managementobjekte beinhalten kann, ist das *Composite* Design Pattern, das *System* mit *ManagedSystemElement* sowie dessen Subklassen verbindet. Es bildet in unserem Fall die Grundvoraussetzung dafür, daß wir sowohl die softwaretechnischen und konzeptionellen Bestandteile des Managementsystems selbst modellieren können (rechter Teil der Grafik), als auch die von ihm verwalteten Ressourcen, deren zugehörige Managementobjekte in dessen Datenbank abgespeichert sind (linker Teil). Diese Managementobjekte sind Instanzen der ODP-Objektklassen, welche Client- und Server-Objekte (sowie deren Interaktionen und Bindungen) darstellen. Ein Partnersystem ist somit in der Lage, das instrumentierte Managementsystem selbst sowie die von ihm verwalteten Ressourcen zu administrieren.

4.3 Prototypische Implementierung

Wir haben die kommerzielle Managementplattform *Tivoli NetView* sowie ein von uns entwickeltes CMIP/SNMP Management-Gateway [12] instrumentiert und die CORBA-basierten Managementagenten in Java implementiert. Die C-APIs des Managementsystems wurden von uns mit IDL-Kapseln versehen, was uns die Möglichkeit bietet, über CORBA darauf zuzugreifen. Client-seitig haben wir einerseits eine web-basierte Benutzerschnittstelle auf der Basis des JavaORBs *VisiBroker for*

Java realisiert, sowie ebenfalls einen Zugang für ein weiteres Managementsystem geschaffen, was uns den Zugriff auf den Agenten auf zweierlei Arten gestattet.

Die Wiederverwendung von CORBA-Diensten war ebenfalls ein Ziel unserer Arbeit. Insbesondere das Navigieren durch die zahlreichen Assoziationen sowie das Aufzählen der Objekte am jeweils anderen Ende einer Assoziation sollte durch die CORBA-Dienste *Query* sowie *Relationship* geleistet werden. Bedauerlicherweise war die uns zur Verfügung stehende Implementierung dazu nicht in der Lage, weshalb wir die entsprechende Funktionalität innerhalb der Objekte zeitaufwendig selbst implementieren mußten. Zukünftig verfügbare CIM Object Manager werden dies jedoch leisten können, was den Implementierungsaufwand deutlich verringern wird.

Als problematisch erwies sich ebenfalls die Implementierung eines hinreichend feingranularen Sicherheitskonzeptes zum Zugriff auf die in den Systemen gespeicherte Ressourceninformation. Schließlich wird ein Provider nur diejenige Informationsmenge offenlegen, die ein Dienstanutzer unbedingt benötigt. Ein auf Zielvorgaben basierendes Viewkonzept könnte dies leisten, jedoch befinden sich die Arbeiten der IETF/DMTF *Policy*-Arbeitsgruppen noch in einem frühen Stadium; insbesondere sind gegenwärtig keine *Policy Engines* verfügbar, die eine flexible und einheitliche Vergabe und Überprüfung von Zugriffsrechten gestatten. Unser Prototyp stützt sich daher auf die (proprietären) Mittel von *NetView* zur View-Definition ab.

5 Zusammenfassung und Ausblick

Wir haben in diesem Beitrag einen neuen Ansatz für CIM/CORBA-basiertes Management verteilter kooperierender Managementsysteme vorgestellt. Die Notwendigkeit hierfür ergibt sich aus der Tatsache, daß das stark zunehmende Outsourcing von IT-Dienstleistungen die Kooperation der Managementsysteme aller beteiligten Partner erfordert, um eine schnelle Ermittlung von Fehlern über Providergrenzen hinweg zu gewährleisten. Dies erfordert die Offenlegung eines Mindestumfangs an Managementinformation und -diensten. Einige charakteristische Beispiele hierfür wurden vorgestellt. Die zentrale Fragestellung des Beitrags ist, wie Managementsysteme – als ein Spezialfall verteilter Anwendungen – modelliert werden können, um eine möglichst große Menge an Managementinformation und -diensten von generischen Modellen für das Anwendungsmanagement zu übernehmen. Wir haben bestehende Ansätze eingehend untersucht und die Kombination von ODP und CIM gewählt, da sich diese Modelle gut ergänzen und eine Vielfalt an wichtiger Information beinhalten. Darauf aufbauend, haben wir ein Objektmodell entworfen und implementiert, das sowohl die Steuerung eines Managementsystems selbst ermöglicht, als auch den Zugriff auf die Daten der von ihm verwalteten Ressourcen gestattet. Eine prototypische Implementierung weist die praktische Anwendbarkeit unseres Ansatzes nach.

Unsere gegenwärtigen Arbeiten befassen sich mit der Migration der Implementierung in die von der DMTF gegenwärtig entwickelte *Web-based Enterprise Management (WBEM)* Infrastruktur. Die dabei aufgeworfenen Fragestellungen hinsichtlich der Filterung und Übermittlung von Ereignismeldungen sowie der Sicherheit und Skalierbarkeit in großen Umgebungen werden es uns ermöglichen, die praktische Anwendbarkeit dieser Architektur zu beurteilen.

Literatur

- [1] Application Management Specification. Version 2.0. Tivoli Systems. (November 1997)
- [2] Application Response Management. Version 2.0. Tivoli Systems. (November 1997)
- [3] Bauer, M., Bunt, R. et al.: Services Supporting Management of Distributed Applications and Services. *IBM Systems Journal* 36(4), 508 – 526 (1997)
- [4] Brusil, P., Hellerstein, J., Lutfiyya, H.: Applications Management – Current Practices, Research Results, and Future Directions. *Journal of Network and Systems Management* 6(3), 361 – 366 (September 1998)
- [5] Bumpus, W., Sweitzer, J., Thompson, P., Westerinen, A., Williams, R.: Common Information Model: Implementing the Object Model for Enterprise Management. J. Wiley & Sons 2000
- [6] Common Information Model (CIM) Version 2.2. Specification. Distributed Management Task Force. (Juni 1999)
- [7] IEEE Standard for Information Technology - Portable Operating System Interface (POSIX) - System Administration - Part 2: Software Administration. IEEE Standard 1387.2. The Institute of Electrical and Electronics Engineers. (1995)
- [8] Open Distributed Processing – Reference Model. IS 10746. International Organization for Standardization and International Electrotechnical Committee. (1995)
- [9] Open Distributed Processing – Reference Model – Part 3: Architecture. IS 10746-3. International Organization for Standardization and International Electrotechnical Committee. (1995)
- [10] Johnson, M. W., Smead, S.: Beyond ARM 2.0 - API Extensions that enable pervasive Service Level Instrumentation. Computer Measurement Group. Dezember 1998
- [11] Kalbfleisch, C., Krupczak, C., Presuhn, R., Saperia, J.: Application Management MIB. RFC 2564. IETF. (Mai 1999)
- [12] Keller, A.: Tool-based Implementation of a Q-Adapter Function for the seamless Integration of SNMP-managed Devices in TMN. In Sahin, V. (Hrsg.): *Proceedings of the IEEE/IFIP Network Operations and Management Symposium (NOMS 98)*. IEEE Press. Februar 1998
- [13] Keller, A., Neumair, B.: Interoperable Architekturen als Basis eines integrierten Managements. In Zitterbart, M. (Hrsg.): *GI/ITG-Fachtagung Kommunikation in Verteilten Systemen*. Springer Verlag, Februar 1997
- [14] Krupczak, C., Saperia, J.: Definitions of System-Level Managed Objects for Applications. RFC 2287. IETF. (Februar 1998)
- [15] Pavlou, G.: A Novel Approach for Mapping the OSI-SM/TMN Model to ODP/OMG CORBA. In Sloman, M., Mazumdar, S., Lupu, E. (Hrsg.): *Proceedings of the 6th IFIP/IEEE International Symposium on Integrated Network Management*. IEEE Publishing, Mai 1999
- [16] Soukouti, N., Hollberg, U.: Joint Inter-Domain Management: CORBA, CMIP and SNMP. In Lazar, A., Saracco, R., Stadler, R. (Hrsg.): *Proceedings of the 5th IFIP/IEEE International Symposium on Integrated Network Management*. Chapman and Hall, Mai 1997
- [17] Sturm, R., Bumpus, W.: Foundations of Application Management. J. Wiley & Sons 1998
- [18] OMG Unified Modeling Language Specification. Version 1.3 ad/99-06-08. Object Management Group. (Juni 1999)
- [19] Verma, D.: Supporting Service Level Agreements on IP Networks. Macmillan Technical Publishing 1999
- [20] Systems Management: Distributed Software Administration. CAE Specification C701. The Open Group. (Januar 1998)