Proceedings of the *IFIP/IEEE Joint International Conference on Open Distributed Processing (ICODP) and Distributed Platforms (ICDP)*, Toronto, Canada, May 1997

Using ODP as a Framework for CORBA-based Distributed Applications Management

Alexander Keller, Bernhard Neumair
University of Munich, Department of Computer Science
Oettingenstr. 67, 80538 Munich, Germany
E-Mail: {keller|neumair}@informatik.uni-muenchen.de

Abstract

Today's complex IT infrastructures require an integrated management of networking devices, end systems, and applications. A necessary prerequisite are management models for resources that are generally applicable and useful even in environments that rely on different management architectures. The paper describes a novel approach to management models for (distributed) applications which meets these needs. The approach is based on concepts taken from the ODP viewpoint languages. It is applicable to a wide range of applications and independent of specifics of existing management architectures. This independence is very important as implementors of management systems gain the opportunity of choosing between at least three different management architectures.

In this paper, the different phases of our approach are depicted: Starting from conceptual modeling, we show how the design of the model can be enhanced by using CASE-tools for refining the object classes. Furthermore, we describe a way to transform this model into concrete CORBA objects. This is demonstrated by means of a concrete example, namely a transaction processing monitor. Finally, it is shown how the implementation of the model fits into an integrated management environment spanning different management architectures.

Keywords

RM-ODP, Distributed Applications Management, CORBA, Management Gateways, Application Modeling

1 INTRODUCTION AND MOTIVATION

Today, distributed systems based on the client/server paradigm have made their way into commercial IT infrastructures. The price of the flexibility gained is a more complex technical management of the computing environment. Efficient operation and administration requires an integrated management, in other words administration should be based on a single conceptual platform, a so-called *management architecture*. We need homogeneous management models for all distributed applications aligned with concepts that have been developed for network management and which now are increasingly used for managing end system resources.

This task has become even more complex through the recent introduction of different management architectures (see e.g. (Hegering, Neumair and Gutschmidt 1994)). On the one hand, there is the wellknown OSI management architecture that is primarily used in the telecommunications area; on the other hand, many IETF working groups are trying to extend the scope of the Internet (SNMP) management architecture, i.e. apply it to the management of distributed applications. The recently launched Web-Based Enterprise Management (WBEM) Initiative will introduce another architecture (see e.g. (Hudis and Sinclair 1996)). Additionally, CORBA (Siegel 1996) becomes increasingly important for management applications. Whereas this architecture has not been developed specifically for management applications but to generally support communication and cooperation within distributed applications, it seems promising to use it for management purposes, too. Unfortunately, the current state of CORBA-based management of end systems and applications leads to an isolated management island. Therefore, bridging between the mentioned management architectures is an important research topic today. Necessary for this bridging are generic management models for resources that can be mapped onto every architecture forming an integrated conceptual framework. These models have to describe the characteristics of resources to be administered from a management point of view. Today, the object-oriented approach is the most commonly accepted paradigm in management: Hence, so-called *managed object classes (MOCs)* provide abstractions of real resources for management purposes. Management acts on resources by manipulating the managed objects. For that, we first of all have to derive appropriate MOCs for management of distributed applications. This is done, as described below, based on terms and concepts of the *Reference Model for Open Distributed Processing* (RM-ODP, ISO 10746) in order to ensure the applicability of the MOCs to a wide range of distributed applications.

Our overall goal is to make integrated management, especially of distributed applications, feasible in an environment consisting of different management architectures. The management models have to be applicable in "CORBA-only" environments as well as in "classical" management environments that rely on the OSI– and Internet management architectures: They should be ready for CORBA-based management but also take advantage of the functionality delivered by already existing management platforms.

The paper is organized as follows (see Figure 1): In section 2, we describe our approach to the definition of management models, i.e. MOCs, for distributed applications that is based on the RM-ODP. These models should be applicable to a vast majority of applications and be widely independent of management transport mechanisms and specifics of management architectures. Section 3 then describes a methodology for developing an object model for distributed applications covering not only the generic, but also the application-specific aspects. The focus of the approach lies on the object model's practical applicability for implementing management agents. It makes use of the *Object Modeling Technique (OMT)* as a notation for the computational and the engineering viewpoint languages and permits a seamless transition from abstract object models to concrete implementations in CORBA. An example of this concept is also given in this section. Since today's management environments are heterogeneous through the use

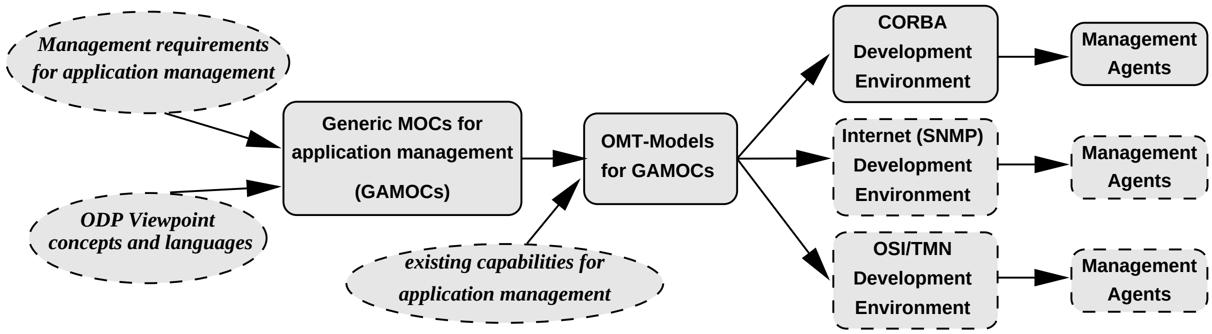

Figure 1 Overview of the approach

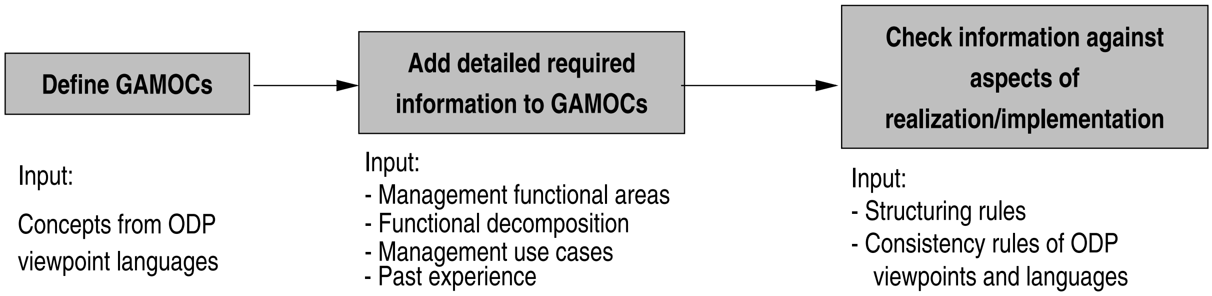

Figure 2 Derivation of GAMOCs

of different management architectures, section 4 deals with the issue of enabling cooperation between these architectures and describes our inter-domain management environment. We demonstrate how gaps between the management architectures can be bridged by the use of management gateways. Section 5 summarizes our approach and concludes the paper.

2 MANAGEMENT MODEL OF DISTRIBUTED APPLICATIONS

The question "what are the characteristics of distributed applications from a management point of view?" is a very current research topic. It is addressed in various recent publications (see e.g. (Pell, Eshghi, Moreau and Towers 1995, Schade, Trommler and Kaiserswerth 1996, W.Hong, Katchabaw, Bauer and Lutfiyya 1995)). Compared to other management areas, there is still a lack of common understanding* of this issue (see e.g. the ongoing discussion related to the development of Application MIBs (*applMib* and *sysApplMib*) for the Internet management architecture). In order to describe these characteristics, we first need a framework for management objects. This framework must allow us to describe relevant aspects of any distributed application in a way that is suitable as a basis for all areas of management. For example, it has to allow the specification of relevant aspects that arise e.g. from software management and distribution on the one hand and from the need to monitor the status of processes and their ability to communicate etc. on the other hand.

Today, there seems to exist only one standard model that is comprehensive enough to address all management issues — the RM-ODP. It is not only standardized but also of growing practical importance due to, among others, the cooperation between the OMG and the ISO. Therefore, we propose to take it as a basis for the definition of the necessary framework for management objects. Our approach consists essentially of three steps (see also Figure 2):

1. *Defining MOCs for applications:* First of all, we have to define the Managed Object Classes themselves, i.e. we have to identify components of distributed applications that are relevant for management purposes. This is done by an analysis of the concepts provided by the ODP viewpoint languages.
2. *Defining and assigning management-relevant information to the MOCs defined in 1.*: This second step is a top-down approach that analyzes requirements of management functional areas, i.e. it addresses the question "what information on resources do we need for management purposes?". The identified information is then assigned to suitable MOCs of the first step.
3. *Ensuring feasibility:* In the third step, the top-down approach is complemented by a bottom-up

*Within other areas, there are models like the OSI Reference Model. Based on these and on many years of experience, there is a common understanding about what makes up a router, switch or other networking devices from a management point of view.

approach based on an analysis of information that is in general available from these resources, and thereby addresses the question “what can reasonably be assumed in general about resources of that class?”

The main benefit of our approach is the wide applicability of the resulting management information to the majority of applications. It takes advantage of the fact that the ODP concepts have been defined with generality as a major design goal. It therefore should foster integrated management of distributed applications. In the following, these mentioned three steps are described in more detail and their application will also be sketched out.

2.1 Generic Application Managed Object Classes (GAMOCs)

As mentioned above, in other areas such as in network management a common understanding exists about the resources that have to be administered. In OSI management, so-called *Generic Managed Object Classes* (GMOCs, e.g. a class *connection-mode protocol machine*) have been specified and then refined to more specialized classes (e.g. *connection-oriented transport protocol machine*). A similar concept lies behind the *Network Services Monitoring MIB* (see RFC 1565), in this case however restricted to monitoring purposes. For application management, such a generally agreed upon set of MOCs does not yet exist, although some valuable approaches can be found in recent publications (e.g. (Schade et al. 1996)).

Therefore, we propose a “universe of discourse” for application management that is aligned with the standardized concepts contained in the ODP viewpoint languages (ISO 10746, Part 3: Architecture). Analyzing the concepts from a management point of view reveals that the *Computational* and the *Engineering Language* are of highest relevance for our purposes, i.e. these concepts define most of the resources that have to be managed. Concepts from the *Information Language* that cover semantics of information processing, are only of secondary concern for (technical) management. Since integrated management should abstract from the technical realization of the managed resources as far as possible, this also holds for the *Technology Language*. Furthermore it applies to concepts from the *Enterprise Language*. The enterprise viewpoint has to be reconsidered if advanced management concepts like policy-based management are introduced (Neumair and Wies 1996).

Therefore a subset of the viewpoint concepts from a management point of view gives rise to the *Generic Application Managed Object Classes (GAMOCs)* which are a new approach presented herein.

Concepts of the computational language allow the functional decomposition of an ODP system into objects which interact at interfaces. Therefore, MOCs based on them, e.g. *operation interface*, *signal interface* or *binding* are important for software distribution and installation. An engineering specification defines the mechanisms and functions required to support distributed interaction between objects in an ODP system. Concepts of the engineering language like *capsule*, *cluster*, *channel* etc. or, respectively, the GAMOCs derived from them, support – as can be seen easily – monitoring of processes, the connections between them etc., i.e. dynamic or runtime aspects of application management. They can therefore be used e.g. for performance monitoring and fault management. See also Figure 3.

2.2 Management requirements: A Top-down Approach

In this step, the “frames” of GAMOCs have to be filled with detailed management information, i.e. with attributes, notifications, actions etc. This information has to address different requirements that arise, for example, from software management and distribution on the one hand and from the need to monitor the status of processes and their ability to communicate with other processes etc. on the other hand. To cope with the former problem area, we need information about other applications (including their respective versions) that are installed and how they have to be configured. Additionally, we have to know some general characteristics of the underlying platform, e.g. the release number of the operating system,

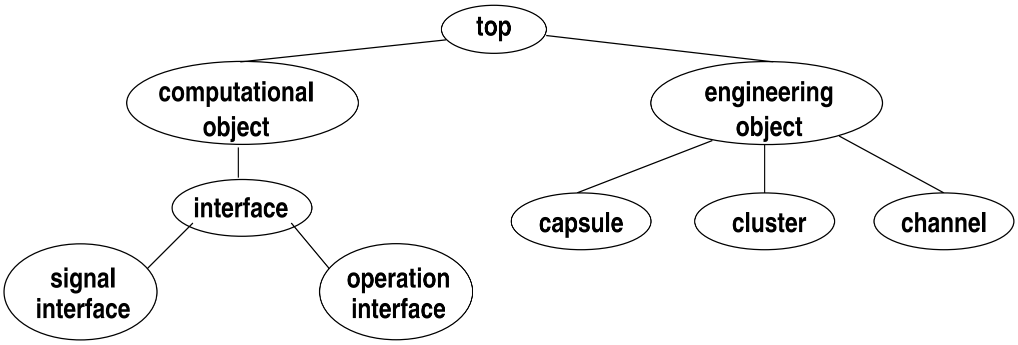

Figure 3 Examples of Generic Application MOCs (GAMOCs)

available disk space etc. In the latter case, required information includes e.g. the status of processes, the end systems they should run on, the status of connections etc. To define this information, we therefore have to use a top-down approach that analyses requirements according to the management functional areas (see e.g. (Kätker 1996)). They stem from a functional decomposition of the whole complex of management activities, i.e. address the question "What information about resources do we need for management purposes?". We have to choose the appropriate GAMOCs representing these resources and map the required information (attributes etc.) to them. Let us have a short look at two examples:

1. Requirements imposed by software distribution and installation can be met by adding appropriate information to GAMOCs related to the computational viewpoint, especially to the different classes of interfaces and binding objects. These can be instrumented to provide information on other software (i.e., interfaces) required by the application.
2. Information on the status of processes and of connections that is obviously necessary for fault management can be mapped to the GAMOCs *capsule* or *channel*, *binder*, *stub*, respectively.

2.3 Bottom-up Approach: Ensuring feasibility

As mentioned before, the requirements-based top-down approach of the previous section has to be complemented by an accompanying bottom-up approach based on an analysis of the modeled resources. For each GAMOC, we have to answer the question "What can reasonably be assumed in general about resources of that class?" or, in other words, "What can be directly obtained from these resources?". If the answer here is negative for information that is required according to the previous step, we should probably not add it to the GAMOC in order to ensure an efficient implementation. In this case we have to find other GAMOCs or other mechanisms to meet this specific requirement. In terms of interoperability of management, this step is the crucial part of the approach. On the one hand, specifying too little information for the GAMOCs will make integrated management inefficient or even impossible. On the other hand, assuming too much about the characteristics of resources will obstruct interoperability and implementation. If the information specified is not implementable or valid for a vast number of resources, management applications will frequently have to cope with responses like "not implemented" or "not available".

The major problem in contrast to other management disciplines is that only little experience with management of distributed applications exists. Hence, answering the question "What can be assumed about resources of a class?" is by far not trivial.

In our approach, we answer the question again in a way aligned with the RM-ODP. For each of our GAMOCs, we analyse the respective *structuring rules* (ISO 10746, Part 3: Architecture) of the viewpoint languages resulting in detailed hints on what information an application should easily be able to provide to management. An example might illustrate this: The rules related to interfaces or interface templates, respectively, show that information on other software including its version and other management information can be introduced without difficulties.

Another important source of information are the consistency rules that relate different viewpoints. For example, rules related to the consistency of engineering and computational specifications justify the definition of information that gives an answer to the following question: ”We installed software package XY — what are the names of the processes that have to run properly in order to have the application in an usable state?”

2.4 Application of GAMOCs

The GAMOCs defined according to the previous sections are generic in the sense that they (intentionally) do not contain any specifics for certain applications. In order to support the management of a concrete application, they of course have to be further refined.

Consider the following situation: Given is an application modeled according to the ODP viewpoint concepts, i.e. compliant to the RM-ODP. From this application, take especially the computational and engineering specification and use the information contained therein to instantiate generic information in the GAMOCs. Step three of the approach as mentioned above ensures that enough information is present in these specifications. For example, information in the GAMOCs related to the names of processes which have to be monitored in order to ensure availability can now be set to appropriate default values.

Now that we have sketched the whole process of the definition of GAMOCs, we have to focus on a concrete technique that we can use to model these classes and on tool support for this activity. In the following sections we describe the use of OMT and our experiences with bringing these classes to life, that is implementing the model as distributed CORBA objects.

3 OMT-BASED DESIGN OF MANAGEMENT AGENTS

As described in the previous section, the GAMOCs based on the ODP viewpoint languages specify useful object classes for application management. They have to be represented in a notation permitting derivation of more application-specific MOCs that then can serve as a starting point for the design and implementation of appropriate management agents. It is therefore crucial to have a notation permitting refinement of the GAMOCs and definition of relationships between them. Furthermore, it should be possible to transform the resulting MOCs automatically into the description languages used by today’s management architectures such as GDMO/ASN.1 (in the case of OSI) or IDL (for CORBA). This section describes our experiences with an object-oriented methodology that meets these requirements; furthermore, we show the application of our approach developed in section 2 to a real-life application management problem.

3.1 Using OMT as a notation for ODP viewpoint languages

Today, specifications of managed object class interfaces may be described in three different incompatible ways, depending on the kind of management architecture being used for implementing these managed objects: The Internet management architecture uses object-based ASN.1 templates as a notation for managed objects while the OSI framework specifies its MOCs in GDMO/ASN.1 templates in an object-oriented way; CORBA (managed) objects, finally, are defined in OMG IDL. This heterogeneity in the domain of management architectures introduces additional complexity for an integrated management.

As ODP-based GAMOCs and the MOCs derived from them should be applicable to every management architecture, it is necessary to specify them in an architecture-independent "meta-notation". The independence from the concrete information models of the underlying management architectures is therefore assured.

During the recent years, the *Object Modeling Technique* (OMT (Rumbaugh, Blaha, Premerlani, Eddy and Lorensen 1991)) has been successfully applied to a wide range of software engineering projects and is currently in widespread use in the telecommunications domain: The *Telecommunications Information Networking Architecture* (TINA, (de la Fuente and Walles 1994)) makes use of the OMT modeling capabilities for the specification of object classes useful for managing telecommunication entities. A similar approach to modeling abstract ODP classes is described in (Colban and Dupuy 1995). Currently, OMT is enhanced by new features of other object-oriented methodologies and will end up in the *Unified Modeling Language* (Booch, Jacobson and Rumbaugh 1996).

As OMT has proven its usefulness, powerful OMT-compliant CASE-tools which facilitate the complex modeling process are available on the market. Adequate tool support is the prerequisite for a structured design of large-scale software development projects, in particular, if numerous objects and their relationships have to be modeled. Since this is the case in the application management domain, we decided to use OMT for our modeling purposes. Our approach makes use of OMT as a notation for the MOCs derived from the computational and engineering viewpoint concepts and permits a seamless transition from abstract object models to concrete implementations in CORBA. This is due to the fact that OMT-compliant CASE-tools available on the market allow the automatic transformation of OMT models into the notations of standardized management architectures (at least in OMG IDL). A rapid-prototyping approach for the development of CORBA-based management agents can therefore be realized.

The concept of using CORBA as an implementation technique is backed by the liaison statements between the Object Management Group (OMG) and the ISO: It is likely that some CORBAservices may become ODP functions; in turn, the OMG will adopt the ODP trader specification as a CORBAservice. There is a strong agreement between the OMG and the ODP working groups to merge their standards in order to deliver a consistent framework for the specification and development of distributed object-oriented applications.

3.2 Transforming GAMOCs into CORBA objects

After the modeling phase consisting of our previously identified three steps (see section 2) is accomplished, a detailed set of information reflecting the generic and the specific management aspects of the application is available. GAMOCs such as *cluster*, *capsule*, *channel* or *node* including their properties have been derived mainly from computational and engineering viewpoint concepts and specified using the OMT notation. As we want to smoothly transform this object model into an implementation in form of a CORBA-compliant management agent (see Figure 4), it is important to have the possibility of performing an *automated* transformation from the OMT notation into the OMG *Interface Definition Language (IDL)*.

Our experiences with CASE-tools available on the market (StP 1995) have shown that the automatic generation of IDL object descriptions works quite well. However, some small syntactical modifications of the IDL output may be necessary in order to make them pass through the IDL compiler of the CORBA toolkit without problems. These modifications reflect the specifics of currently available CORBA development toolkits and can be automated through altering the grammar of the CASE-tool's code generator. It is expected that these problems will disappear as the CORBA toolkits become more mature.

It is now the implementor's choice to decide in which programming language the management agent actually has to be implemented. This depends on the available mechanisms for retrieving the desired information from the application. After that, the IDL object descriptions are given to the IDL compiler which generates the *implementation skeleton* of the agent in a concrete programming language through standardized *language mappings*. Finally, the newly written agent code is compiled and linked; the result

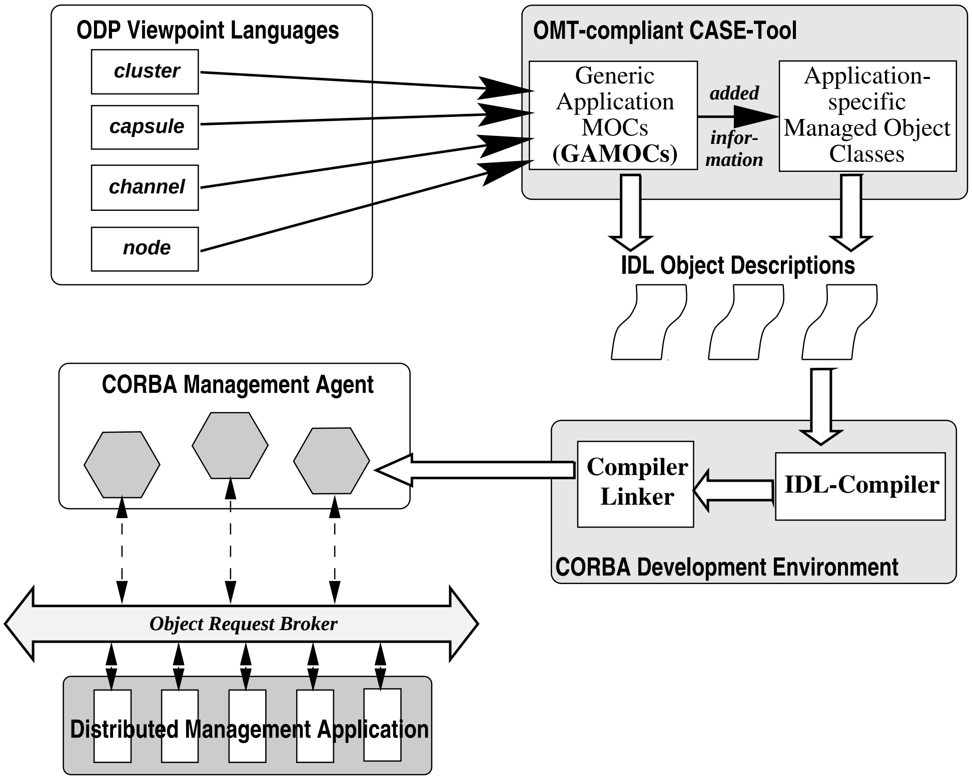

Figure 4 Overview of the transformation process

is a fully CORBA-compliant management agent consisting of a set of managed objects representing the management interfaces of a distributed application.

3.3 Case study: Managing the CICS TP Monitor

The purpose of TP Monitors is to ensure the transaction properties (ACID) of applications. A widely used TP monitor is the IBM *Customer Information Control System (CICS)*. Its high complexity yields a strong demand for powerful monitoring and administration concepts. Due to the heterogeneity of the underlying systems (from PCs to mainframes), having an integrated view is particularly important for managing the distributed application CICS. In this section, we will describe how the GAMOCs can be applied to CICS-management; the underlying scenario is the installation of new application software. As mentioned in section 2.2, information relevant for software distribution and installation is found in GAMOCs related to the computational viewpoint; they serve therefore as base MOCs for CICS-specific MOCs.

If new applications have to be installed, detailed program- and transaction-specific information is required, e.g. the location, the version and the parameters of the subsystems the application relies on. It is also necessary to determine the relationships of transactions, applications, and users to the systems on which they are defined and to the corresponding security and database systems. In order to cope with this huge amount of information, it has to be structured and filtered according to specific criteria. Given a transaction identifier it must be possible, for example, to determine the corresponding region and the systems from which the transaction can be issued remotely. Analogous requirements exist for all resources having multiple occurrences in the same domain and representing the same functionality. Starting from our software installation scenario, we are able to identify the required management information and the necessary management functionality (Fischer 1996):

- **Management information**: Transactions, applications, files and users affected by the change; the already installed version; the region; the security system and its administrator; the database system and its administrator.
- **Management functionality**: Deactivate transactions, applications and files; determine the current version; determine the prerequisite software for the new version of an application; apply resource definitions in the regions subject to the change; activate transactions, applications and files; inform security and database system administrators.

After the relevant MOCs provided by RM-ODP have been identified (step 1 of our approach), it is now necessary to map the required information to the GAMOCs (step 2; top-down approach). As will be seen, some kind of information is already contained in the computational and enterprise languages. The remaining items have to be resolved by specifying additional CICS-specific MOCs together with their attributes and methods. These have to be checked against aspects of implementation, i.e. the question "is the required information available from the system?" must be answered (step 3; bottom-up approach). The direct mapping of the management information to the GAMOCs can be done as follows:

- A transaction can be regarded as a *sequence of interactions*, i.e. a *flow*, occuring at *interfaces* with the necessary management information therefore being available from the respective *computational interface templates* and *interface signatures*.
- The relationship between the CICS systems and databases or security systems can be looked at from the computational viewpoint as a permitted *sequence of interactions* at the corresponding *interfaces* or, from the engineering viewpoint, as a *channel*, i.e. a configuration of *stubs*, *protocol objects* etc. Again, management information is available from the respective *interface templates* or *signatures* or stem from *channel rules*.
- Files of the CICS system are examples of *clusters*, i.e. single units for checkpointing, recovery and migration.
- Certain aspects of CICS-regions have to be modelled as *binding objects*. This provides information about the resources of these regions (e.g. terminals, applications, files etc.)

These examples show that the MOCs derived from ODP viewpoint concepts support a broad range of tasks identified by management use cases. The information provided by them is the basis for execution of management functions from a console or for the application of ODP functions (e.g. *deactivate*, *reactivate*, *recovery* or *checkpointing*) for management purposes. Our experiences have shown that this is also true for entities and tasks identified in other scenarios dealing with different kinds of distributed applications. We were therefore able to implement our GAMOC-based management model in a very straightforward fashion by applying the transformation process described in section 3.2. The results are CORBA-compliant distributed objects for management purposes interacting via an ORB. Together, they form a "CICS management agent".

4 INTEGRATION IN THE MANAGEMENT ENVIRONMENT

The result of the process described in the previous sections are native CORBA management agents. In order to exploit the functionality presented by these agents, it is necessary to provide means of implementing CORBA-compliant management applications. The obvious way to achieve a native CORBA-based management solution is to develop management applications and management platforms as well as agents in the form of distributed CORBA objects communicating through an ORB. Unfortunately, at the current stage of available implementations, scalable management solutions based exclusively on CORBA are not yet possible. Furthermore, integrated management environments span different architectural domains. Therefore, so-called management gateways (see e.g. (Kalyanasundaram and Sethi 1994, McCarthy, Pavlou, Bhatti and Neuman 1995)) bridging the gaps between the involved management architectures are needed. By using gateways, we are able to manage services, systems and networks in three different management architectures from a single point of control. It is even possible to apply most of the power of the OSI management architecture to any resource in the different architectural domains. Therefore, our design guideline was to take the "best of breed" of the three management architectures (OSI, Internet, CORBA). We have used off-the-shelf products and tied them together in order to form an integrated service, system and network management prototype.

As the OSI management architecture yields by far the largest set of management functionality, we decided to take an OSI-compliant management platform as the core of our distributed management prototype (see Figure 5). For the sake of brevity, we can only sketch the approach. The *IBM NetView TMN*

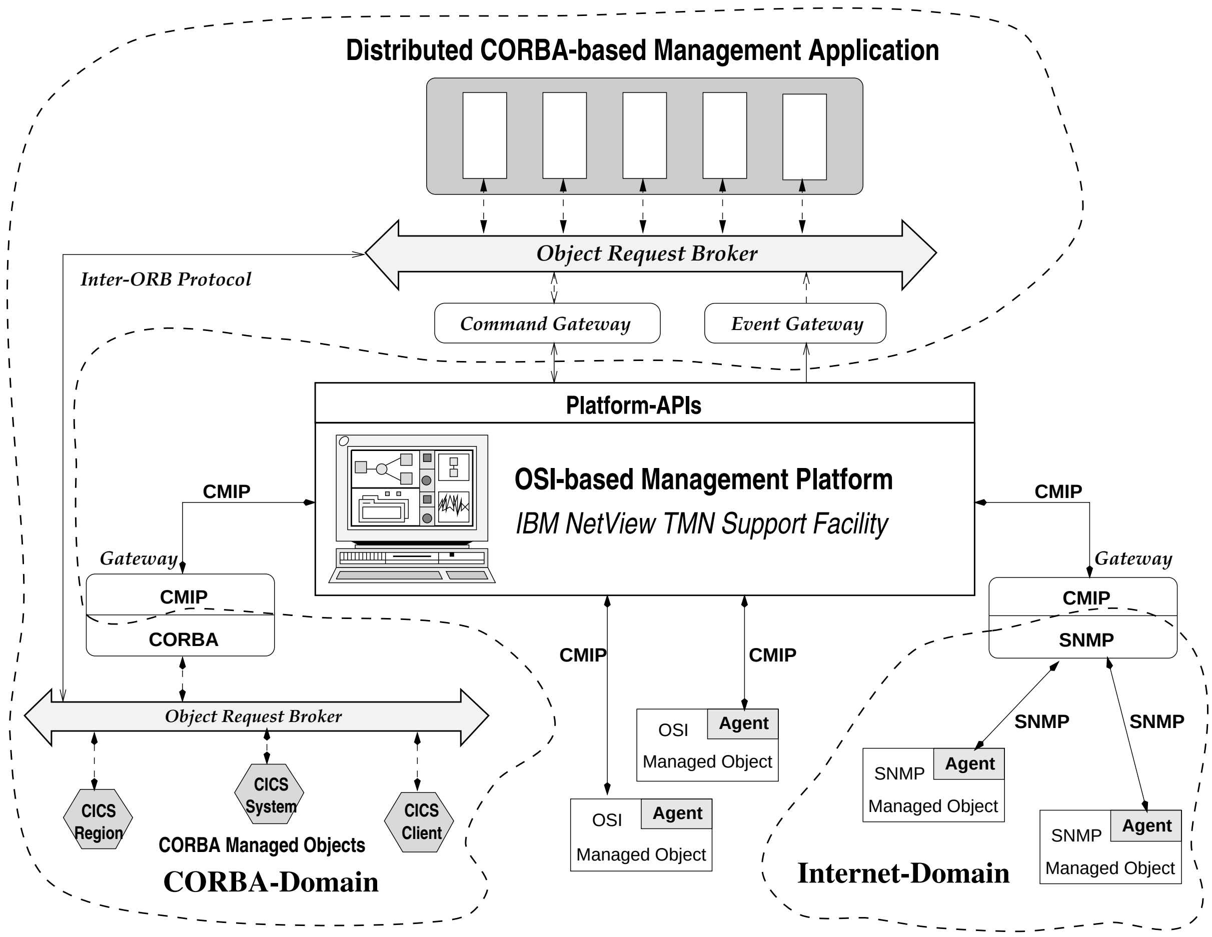

Figure 5 Interoperability between different management architectures

Support Facility (Feridun, Heusler and Nielsen 1995) provides all the necessary features for managing OSI/TMN-compliant management agents. Since SNMP agents are widespread, an integrated management solution needs to take into account the Internet management domain. It was therefore necessary to develop a *CMIP/SNMP management gateway* (Langer 1996) for this purpose allowing the management of SNMP agents from an OSI managing system. Our implementation makes use of the IIMC *(ISO-Internet Management Coexistence)* concepts for achieving interoperability between the OSI and Internet management architectures. In management architectures such as the Internet framework that have no notion of a management functional model, the application of management functionality "borrowed" from other architectures is particularly useful. A typical example for this is to enhance the Internet management architecture with management functionality like threshold monitoring or event processing defined by the OSI Systems Management Functions. The integration of CORBA into our distributed management system has also been done through the gateway approach making the defined ODP-compliant GAMOCs and the derived application-specific MOCs accessible from different management architectures.

5 SUMMARY AND CONCLUSION

The paper has shown a novel approach to management models for (distributed) applications. This approach is based on standardized concepts specified within RM-ODP, especially in the definition of languages for the computational and engineering viewpoints. It also takes into account general applicability to a wide range of applications as well as aspects of realization and implementation. Nevertheless, it is independent of specifics of the different existing management architectures. The models lead to management interfaces that are widely consistent across different applications; the goal of this consistency is to foster integrated application management. For example, fault management applications monitoring the status of systems and processes can be implemented in a much more efficient way when consistent interfaces are present. Another example of the benefits of our approach is the mentioned support for pre-installation tests in the field of software management. This has been demonstrated by a concrete example, the CICS transaction processing monitor: By applying the defined GAMOCs, we were already able to cover a wide range of CICS management use cases. This is due to the fact that these contain the knowledge about properties common to all kinds of (distributed) applications.

The technical realization of our approach was another target of the paper: We described how we applied an off-the-shelf CASE-tool to the problem domain of application management and how we successfully implemented a CORBA-compliant application management agent in the context of an inter-domain management scenario. The acquired solution permits management of applications across the boundaries of the OSI, Internet and CORBA management architectures and is therefore not only application-independent but also architecture-independent. Due to the heterogeneity of the underlying environments, we believe that these two properties (application-independence and architecture-independence) are the key issues for the success of integrated application management.

Acknowledgements

The authors wish to thank the members of the Munich Network Management (MNM) Team for helpful discussions and valuable comments on previous versions of the paper. The MNM Team directed by Prof. Dr. Heinz-Gerd Hegering is a group of researchers of the University of Munich, the Munich University of Technology, and the Leibniz Supercomputing Center. We are also grateful to the anonymous referees for their comments.

6 REFERENCES

Booch, G., Jacobson, I. and Rumbaugh, J. (1996). *The Unified Modeling Language for Object-Oriented Developmant, Documentation Set*, Rational Software Corporation.

Colban, E. and Dupuy, F. (1995). Analysis and Design of a Management Application using RM-ODP and

OMT, *Proceedings of the IFIP International Conference on Open Distributed Processing ICODP'95, Brisbane, Australia*.

de la Fuente, L. A. and Walles, T. (1994). Management Architecture, *Version 2.0*, Telecommunications Information Networking Architecture Consortium.

Feridun, M., Heusler, L. and Nielsen, R. (1995). Implementing OSI Agents for TMN, *IBM Research Report RZ 2759*, IBM Research Division, Zurich Research Laboratory.

Fischer, B. (1996). *Entwurf und Implementierung eines Objektmodells für das Management von TP-Monitoren*, Master's thesis, Technische Universität München.

Hegering, H.-G., Neumair, B. and Gutschmidt, M. (1994). Cooperative Computing and Integrated System Management — A Critical Comparison of Architectural Approaches, *Journal of Network and Systems Management* **2**(3): 283–316.

Hudis, I. and Sinclair, A. (1996). HyperMedia Management Protocol Overview, *Internet draft*.

Kalyanasundaram, P. and Sethi, A. (1994). Interoperability Issues in Heterogeneous Network Management, *Journal of Network and Systems Management*.

Kätker, S. (1996). A Modeling Framework for Integrated Distributed Systems Fault Management, *Proceedings of the IFIP/IEEE International Conference on Distributed Platforms ICDP'96, Dresden, Germany*.

Langer, M. (1996). *Entwurf und Implementierung eines CMIP/SNMP Gateways*, Master's thesis, Technische Universität München.

McCarthy, K., Pavlou, G., Bhatti, S. and Neuman, J. (1995). Exploiting the power of OSI Management for the control of SNMP-capable resources using generic application level gateways, *Proceedings of 4th International Symposium on Integrated Network Management*, Chapman & Hall.

Neumair, B. and Wies, R. (1996). Case Study: applying Management Policies to manage Distributed Queuing Systems, *Distributed Systems Engineering Journal*.

Pell, A., Eshghi, K., Moreau, J. and Towers, S. (1995). Managing in a distributed world, *Proceedings of 4th International Symposium on Integrated Network Management*, Chapman & Hall.

Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F. and Lorensen, W. (1991). *Object-Oriented Modeling and Design*, Prentice-Hall International, Inc.

Schade, A., Trommler, P. and Kaiserswerth, M. (1996). Object Instrumentation for Distributed Applications Managemen t, *Proceedings of the IFIP/IEEE International Conference on Distributed Platforms ICDP'96, Dresden, Germany*.

Siegel, J. (1996). *CORBA Fundamentals and Programming*, John Wiley & Sons, Inc.

StP (1995). *Software through Pictures Technical White Paper*.

W.Hong, J., Katchabaw, M. J., Bauer, M. A. and Lutfiyya, H. (1995). Modeling and Management of Distributed Applications and Services Using the OSI Management Framework, *Proceedings of the 12th International Conference On Computer Communication (ICCC'95)*.

7 BIOGRAPHY

Alexander Keller received his Diploma in Computer Science from the Munich University of Technology, Germany, in 1994. Since then he is a Ph.D. student at the University of Munich doing research on distributed systems and application management, emphasizing on CORBA as the underlying management architecture.

Bernhard Neumair received his Diploma and a Ph.D. in Computer Science from the Munich University of Technology, Germany. Currently he is a researcher at the Ludwig-Maximilians University in Munich. His research interests include communication architectures and architectures and concepts for an integrated management of distributed computing environments.