A Modeling Framework for Integrated Distributed Systems Fault Management

Stefan Kätker
IBM European Networking Center
Vangerowstr. 18, D-69115 Heidelberg, Germany
e-mail: kaetker@heidelbg.ibm.com

Abstract
This paper describes a modeling framework for integrated fault management of distributed systems. The model integrates all different layers of distributed systems such as application, system, and network layer in a single, consistent view. This enables generic management applications to perform their tasks across layer boundaries of the distributed system without knowledge about the specific details. The focus is on fault management issues.

Dependencies between resources critical for the availability of the distributed system are modeled using relationships. Generic fault management applications are hereby enabled to determine the root cause of a distributed system failure automatically.

The SAP R/3 application serves as an example to demonstrate the capabilities of the modeling framework.

Keywords
Network Management, Application Management, Fault Localization, Event Correlation

1 INTRODUCTION

The overall success of the SAP R/3* business transaction application shows that even business critical tasks are now performed using distributed systems. The scalability of client/server applications and the down-sizing trend of the recent years lead to more and more complex installations of distributed systems. Therefore new requirements for availability and maintainability of distributed systems arise. Customers require a single point of control for the entire distributed system. An integrated management solution is needed that covers distributed application, system, and network management.

Although integrated management is a very critical and important task, real integration of network, application, and systems management is missing in today's implementations and concepts of distributed system management. ISO and other standardization bodies

* Trademark of SAP AG, Walldorf, Germany

have started work in this area by work on the Open Distributed Processing (ODP) framework (ISO, 1995). The standardization of ODP currently focuses on distributed system design and implementation rather than on integrated management of these systems. A standard approach for integrated management of distributed systems is missing.

In the literature a few concepts for generic distributed systems management are known. The work of Zimmermann (Zimmermann, 1992) is focussed on design and configuration management of distributed applications. Popien et al. discuss OSI service management based on ODP concepts (Popien et al, 1994). Although these concepts are based on the OSI management framework, network and systems management aspects are not addressed.

In the following a distributed system is considered as different layers of software components (Sloman, 1995). The application layer consists of application software probably distributed across several nodes in the system. The system layer performs basic operating or database system functionality. The communication layer is responsible for the communication between the different parts of the system. It provides functionality according to the ISO Reference Model for Open Systems Interconnection. Distributed system management is performed today by specific management applications for specific parts of the distributed system (e.g. IP management, operating system management, or SAP R/3 management). Each of these management applications has to implement the interfaces with other management applications. Today, the Simple Network Management Protocol (SNMP) (Case et al, 1990) framework is used by most of these management applications. All these different management applications run on a central console, but integrated management is limited to user interface integration.

An integrated view on the distributed system that covers all layers of the distributed system is basically needed for the following two reasons:

1. Reduce the costs for distributed system management
   The existence of a generic and unique view on the resources in a distributed system prevents development of new management applications for each new distributed application or protocol suite. A generic model allows generic management applications to perform their tasks (e.g. fault management) without specific knowledge of the distributed system resources. This reduces the development costs of distributed applications as well as the costs for training and maintenance of these applications, because generic management applications are capable of managing several different distributed applications.
2. Integrated models for integrated management
   Especially for fault, configuration, and performance management purposes, knowledge about the interworking of the different layers of the distributed system is necessary. This knowledge must be represented in a unique and generic way to enable management applications to perform their tasks across layer boundaries. Effective management of distributed systems is only possible if the system to be managed is viewed and represented as a whole instead of different parts.

The interworking relationships between different parts of a distributed system are of particular importance for fault management applications. In a distributed environment, faults occur frequently because of interworking problems between different layers of the system. Faults are propagated through different system components and many symptoms result from the same root fault. In order to separate between root faults and symptoms, and to localize a root fault for a given symptom, a generic view of the distributed system and the dependencies between the different components is necessary.

The current solution for the management of distributed systems using specific fault management applications for specific parts of the distributed system is unsatisfying. However, existing solutions to management of distributed applications, like the management of the SAP R/3 system, have to be integrated into the generic distributed system model.

The focus of this paper is on fault management of distributed systems. The goal is to sketch a generic distributed system model for fault management purposes and to show how existing management concepts for distributed application management fit into this model. The major purpose of the model is the representation of the relationships between different parts of the distributed system. Modeling of fine-grained system activities and fault diagnosis is left over for more suitable specific fault management applications that might be triggered by generic fault management systems for further fault diagnosis.

After introducing basic concepts of OSI management in Section 2, Section 3 describes the distributed system model proposed here. An example for the integrated model of the distributed SAP R/3 application and TCP/IP networks is given in Section 4.

2 OSI MANAGEMENT

The OSI management framework (ISO, 1991a) is based on an object-oriented representation of the managed resources. These resources are modeled as Managed Object Classes (MOC). MOCs are described using a specification language called "Guidelines for the Definition of Managed Objects (GDMO)" (ISO, 1991b). Instantiations of these classes are called Managed Objects (MOs). These instantiations grouped in so called Management Information Bases (MIBs) are provided by agents that reside in the network. Central management applications (managers) communicate with these agents using standard protocols and services (Common Management Information Protocol, CMIP and Common Management Information Services, CMIS). Readers not familiar with the framework are referred to (Sloman, 1995).

In the context of this paper the main purpose of fault management is given by fault isolation and event correlation functionality. In large scaled networks a single fault leads to a huge number of symptoms visible as events on the network management console. Fault isolation and event correlation techniques are needed to seperate symptom from real fault information in order to minimize distributed system downtime.

2.1 Relationships in OSI Management Framework

Relationships between MOs can be modeled using a GDMO extension. This extension was defined in (Clemm, 1993). In the meantime ISO/IEC has started work to standardize relationships between MOs (ISO, 1994). We are using the concept and notation for the definition of managed relationships defined by Clemm.

In order to model relationships two new templates are provided as a GDMO extension. Using these templates Managed Relationship Classes (MRCs) and Relationship Bindings (RBs) can be specified. A MRC is used to specify the type of the relationship and relationship attributes. Mandatory part of the MRC is the definition of roles the participating MOs are playing in the relationship. Consistency rules, like the characteristics a MO has to support in order to fulfill a certain role, are also defined in the MRC. A relationship binding contains pointers to the MOs participating in a certain relationship instance.

In order to visualize managed relationship and relationship binding instances, we have defined the graphical representation shown in Figure 1. Circles represent managed objects,

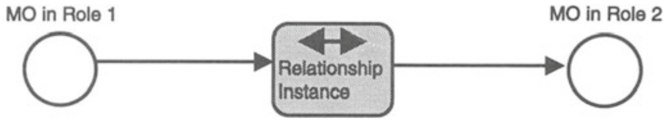

Figure 1 Graphical Representation of Managed Relationships and Relationship Bindings

while rounded rectangles represent MRC instances. Relationship bindings are shown by arcs between managed objects and relationship instances. A pair of arcs connected to the same relationship represents a single relationship binding.

2.2 Virtual Attributes

Another extension of the OSI management framework is the concept of virtual attributes. It was introduced in (Bapat, 1993) to allow externally visible attributes to be defined in terms of other attributes. This concept is also not part of the standard GDMO and is viewed as an GDMO extension. In contrast to the concept of managed relationships the concept of virtual attributes is currently not considered for standardization. However, we feel that this concept is very useful to define generic object models.

A virtual attribute is an attribute which is specified in terms of a value syntax which is the result of a computation involving values of other attributes (which may in turn themselves be virtual). Because this computation is specified as part of the attribute definition, it is assumed that the algorithm implementing this computation is implicitly encapsulated within the MO itself, or in the agent providing the MO. Bapat proposes the syntax shown in the following example to define virtual attributes.

```
aggregateSpeed VIRTUAL ATTRIBUTE
    WITH ATTRIBUTE SYNTAX BitsPerSecond;
    MATCHES FOR Equality, Ordering;
    COMPUTED FROM aggregateSpeedAlgorithm;
REGISTERED AS "..."
```

The `aggregateSpeedAlgorithm` is for example given as the sum
`aggregateSpeed = port1Speed + port2Speed + port3Speed.`

3 A GENERIC MODEL FOR DISTRIBUTED SYSTEMS

Figure 2 shows the desired architecture of an integrated distributed system management platform. Core part of this architecture is the generic **distributed system model (DSM)**. The DSM serves as an interface between the generic management applications

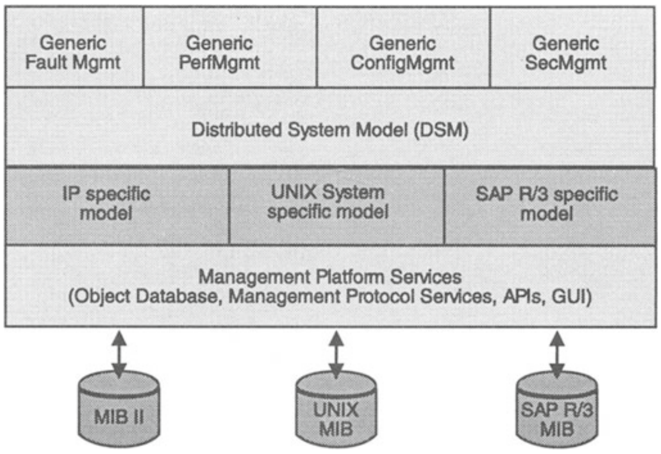
Generic Fault Mgmt
Generic PerfMgmt
Generic ConfigMgmt
Generic SecMgmt
Distributed System Model (DSM)
IP specific model
UNIX System specific model
SAP R/3 specific model
Management Platform Services (Object Database, Management Protocol Services, APIs, GUI)
MIB II
UNIX MIB
SAP R/3 MIB

Figure 2 Architecture of an Integrated Distributed System Management Platform

and the MOs in the specific MIBs. It decouples the management applications from network or application specific details.

The preliminary goal of the integrated distributed system model is to provide the basic source of information for generic management applications. These applications should be enabled to perform their management tasks without special knowledge about the details of the distributed system assembly, configuration and type.

3.1 Model Requirements

Generic fault management of distributed systems requires the following characteristics:

1. *Integrate all aspects and layers of the distributed system in one consistent, generic model of the entire system* The distributed system model should offer an abstraction level covering all the layers of the distributed system. The different types of distributed applications (e.g. client-server or conferencing applications), the different types of operating or database systems and networks must be modeled by a unified representation. This enables generic management applications to perform their tasks without specific knowledge about a specific part or layer of the distributed system and without knowledge about the part of the distributed system a certain resource belongs to.
   The distributed system model must hide the specific information models and specific information access methods (management protocols) from the generic management application.
2. *Integrate existing information models* The distributed system model serves as an interface between the generic management applications and the specific information models for the specific parts of the distributed system, e.g. distributed applications or net-

works. These models have to be integrated into the distributed system model without changes to the existing models.

3. *The DSM must support fault management and automated fault localization* The model has to provide aggregated status information based on the status information derived from the specific information models. The aggregated status information enables the generic fault management application to access information about resource status without having specific knowledge about the resource itself.
   Furthermore information about dependencies between resources that work jointly to provide a service of the distributed system have to be represented by the model. This kind of information is particularly needed for automated fault localization.

In order to fulfill the requirements listed above, the DSM is defined in terms of the OSI management framework because of its powerful object-oriented approach. This framework is particularly useful to define generic object models.

The DSM represents a distributed system by a set of services. A service of a distributed system is for example a TCP connection between two nodes, CPU time provided on a single node, or a complex application service provided by a distributed application. This application uses CPU time, communication, and database services to fulfill its task.

The **service model** describes the services of the distributed system. Service provider relationships link the service representation objects with the objects that represent the resources providing the services.

The **dependency model** describes the dependencies between the services in the distributed system. These dependencies represent which services are used by a certain service in order to perform its task. Figure 3 shows the concept of the DSM using the SAP R/3

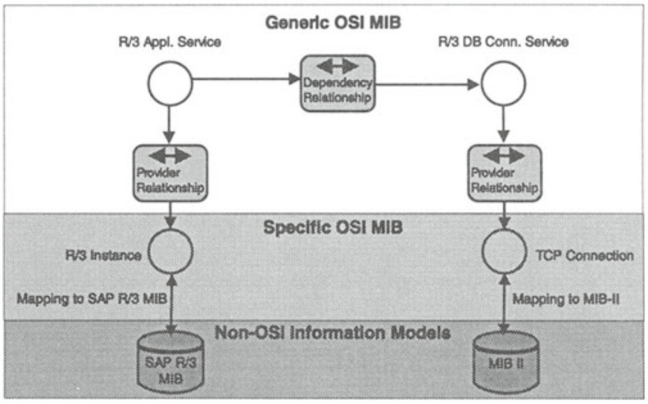

Figure 3 DSM for the SAP R/3 Application

example described detailed in Section 4.1.

3.2 The Service Model

The service model models all the services offered by the distributed system for internal and external use in terms of service instance MOs. See Figure 4 for the description of the `serviceInstance` MOC.

A TCP connection between two nodes is for example represented by a TCP connection service provided by the TCP connection MO. CPU time, hard disk, or virtual memory provided by the operating system are as well modeled as services in the same way as services that distributed applications offer to the end user.

```
serviceInstance MANAGED OBJECT CLASS
   DERIVED FROM "ISO/IEC 10165-2":top;
      serviceInstancePackage PACKAGE;
   BEHAVIOR
      serviceInstanceBehavior BEHAVIOR;
      DEFINED AS " ... ";
   ATTRIBUTES
      serviceInstanceId       GET,
      serviceInstanceType     GET,
      availabilityState       GET,
      qualityOfService        GET;
```

```
   NOTIFICATIONS
      create,
      deletion,
      stateChange;
REGISTERED AS  ... ;

availabilityState VIRTUAL ATTRIBUTE
   WITH ATTRIBUTE SYNTAX
    Attribute-ASN1Module.AvailabilityStatus;
   MATCHES FOR EQULALITY,
        SET-COMPARISON, SET-INTERSECTION;

   COMPUTED FROM availStateAlgo;
REGISTERED AS ...;
```

Figure 4 Service Instance MOC

The most important attributes for fault management purposes are the quality of service and the availability state of the service. If, e.g., the service is no longer provided as specified, i.e. a failure occurs, the availability state changes and a failure notification will be sent to the manager.

3.3 Refinement of the Service Model

In order to link the generic service model with specific information models that represent the different parts of the distributed system a refinement of the generic DSM is necessary. The refinement consists of the following three parts:

1. *Definition of specific service instances by inheritance from the generic service MOC.* Specific services, e.g. a TCP connection service, are derived from the generic service instance MOC. Attributes of a specific service were added to the service instance MOC (examples are given in Section 4). Entities representing the service provider resources are not part of the generic DSM. MOs that describe these resources can be found in the specific MIB that describes the specific part of the distributed system.
2. *Definition of service provider relationships* A service provider relationship MRC links the service model with the specific MOs representing the service provider resources. The service provider role of the relationship is fulfilled by the specific MO representing the service provider resource.
3. *Usage of aggregated attributes* The availability state attribute and other attributes of the generic service instance MOC that are used by the generic management applications

depend on certain attributes of the service provider MO. The availability state attribute can be specified as a function over attributes of the service provider MO. The concept of virtual attributes is used to specify this aggregation. The specific service instances MO derived from the service instance MOC defines the algorithm used to compute the virtual attribute.

4. *Integration of non-OSI information models via mapping objects* Non-OSI information models are integrated into the DSM by mapping objects. Mapping objects are MOs defined in terms of the OSI Management Framework. They represent instances of the non-OSI information model and map the information and protocol functionality from the OSI to the non-OSI world and vice versa. In (Abeck et al, 1993) an approach to integrate SNMP into the OSI management concepts is presented. An implementation of a general SNMP - CMIP gateway is part of the OSIMIS platform (Pavlou et al, 1994).

3.4 The Dependency Model

The dependency model serves as the basic knowledge source for fault management purposes. It represents the dependencies between the services in the DSM. The dependencies are used by the generic fault localization algorithm to determine the root cause of a symptom by following the dependency chain from the symptom service to the service that has caused the root problem. In (Kätker, 1995) an algorithm for fault localization based on dependency analysis is presented.

The dependency model is defined by the MRC `depRelationship` and the RB `depRelBinding` (see Figure 5). Dependency relationships are relationships between two services. To provide its service, the service fulfilling the depends-on role depends on the service fulfilling the dependent service role to provide its service.

```
depRelationship MANAGED
                RELATIONSHIP CLASS
    DERIVED FROM "A. Clemm":mrTop;
    ROLE dependentService
       CARDINALITY (1 ... many)
       REQUIREMENTS
            serviceInstancePackage;
    ROLE depends-onService
       CARDINALITY (1 ... many)
       REQUIREMENTS
            serviceInstancePackage;
    ATTRIBUTES
        priority        GET-REPLACE;
REGISTERED AS " ...";
```

```
depRelBinding RELATIONSHIP BINDING
   RELATED CLASSES
     MOC serviceInstance WITH ROLE
                  dependentService,

     MOC serviceInstance WITH ROLE
                 depends-onService;

   BEHAVIOR depRelBindingBeh;
REGISTERED AS " ...";
```

Figure 5 Description of the Dependency Model

A TCP connection service between two TCP Service Access Points (SAPs) for example depends on the availability of the underlying IP connection service between the corresponding IP SAPs in order to provide its service to higher layer protocols or applications. This dependency is modeled by a dependency relationship MRC.

The relationship binding `depRelBinding` models the link between the dependency type and the service instances that are related. It serves as the link between the service model and the dependency model.

The `priority` attribute of the dependency relationship specifies the order in which the algorithm will follow the dependencies. If a certain service under investigation depends on more than one other services the dependencies with the highest priority (lowest value) is inspected first. If two dependencies have the same priority the order is random.

Refinement of the dependency model is performed by definition of specific dependency relationships derived from the generic relationship model.

The dependency relationship binding is represented graphically by an arrow from the service instance fulfilling the dependent service role to the dependency relationship instance and by an arrow from the dependency relationship instance to the service instance fulfilling the depends-on role.

3.5 Fault Management based on the DSM

This section sketches briefly the use of the DSM for generic fault management applications. The DSM can be used to provide automated fault isolation and event correlation efficiently.

In case of a fault in the distributed system the fault management application receives a notification indicating an availability state change from Up to Down of a service instance object. The application in turn investigates all dependencies for that service instance and checks the availability state of the services the current service depends on. If the availability state of one of these services is Down, the event signaling the Down availability state is treated as a symptom of the depends-on service. The fault isolation process is now started for this depends-on service. For further details see (Kätker and Paterok, 1994).

In order to enable the fault management application to provide fault localization and event correlation for arbitary distributed applications like the SAP R/3 application, instances of the DSM have to be created that represent important services and dependency relationships of the system. Changes to the generic fault management application are not necessary in order to provide this functionality.

4 MODEL FOR THE SAP R/3 APPLICATION

This section presents an comprehensive example to demonstrate the use of the DSM for modeling different layers of the DSM. Section 4.1 contains a DSM for the SAP R/3 application, while Section 4.2 sketches a model for the network layer the SAP R/3 application depends on. An overview of the model is given in Figure 7.

4.1 Application Layer Model

SAP R/3 provides services for hundreds of users distributed over multiple application servers. Each application service, e.g. adding a sales order, uses implicitly R/3 core resources which again will be distributed over several systems (Buck-Emden and Gulimow, 1995). A cluster of R/3 systems consists of a central database server and several application servers connected to the database server to store and retrieve data needed to perform the application tasks. Management of R/3 systems is performed using the SNMP management framework. Information about R/3 systems on a node are represented in the SAP R/3 SNMP MIB and provided by special SNMP subagents.

An R/3 instance MOC is defined to serve as a mapping object between the distributed system model and the R/3 SNMP MIB. See Figure 3 for the configuration of the R/3 model. A detailed model for the SAP R/3 application based on the service model described here is given in (Kätker, 1995).

4.2 Network Layer Model

In the following a dependency model for a TCP/IP (Comer, 1991) network is sketched. The model can easily be adopted to other packet switching network technologies like IPX or LAN MAC layer protocols. Routes in an IP network that consist of several hops between intermediate nodes and other basic characteristics of routes (e.g. the loop free property, completeness, etc.) have to be represented by the DSM. Several types of services, dependency relationships, virtual attributes, and mapping objects to instances in the MIB-II are used to perform this task.

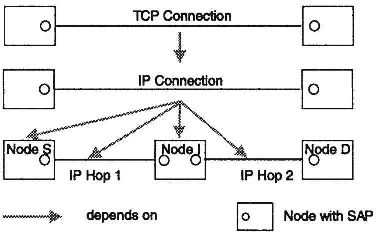

Figure 6 TCP/IP Protocol Layering

Figure 6 shows a TCP connection over an IP network. The TCP connection uses and depends on an IP connection, that consists of an IP route through the network build by several IP hops. Figure 7 shows how this topology is represented in the DSM. For this example three key dependencies for an operational IP connection are identified: There is a complete path from the source SAP to the destination SAP, this path is loop free, and all hops on this path between two intermediate nodes are operational, i.e. they provide a connection service between these two nodes.

In order to model these dependencies, service MOCs have to be defined to represent the services the IP connection service depends on. Furthermore dependency relationship MRCs are needed to model the dependencies between the IP connection service and the services mentioned above. Each of the dependencies listed above leads to a different service type, i.e. service MOC derived from the generic service instance MOC, a different relationship MRC, and a relationship binding.

Modeling the Complete Path Dependency

In order to provide a complete path from the source SAP to the destination SAP all intermediate nodes have to know about the next hop SAP (the next SAP directly reachable from the current node). The routing tables of the intermediate nodes have to contain an entry for the destination IP network. This knowledge is modeled by a next hop service. An instance of this service is defined by the destination SAP and the providing node. If the providing node knows the next hop to the destination SAP, the availability state of the next hop service is up, otherwise it is down. For each intermediate node on the route a next hop dependency binding exists that links the next hop services of all nodes with the IP connection service MO for the IP connection under investigation.

Modeling the Loop Free Path Dependency

This dependency cannot be represented by a number of services offered by different resources. The loop free path dependency is a logical dependency on the property of the entire path. The underlying route determination algorithm will discover the loop property if it exists. For details on this algorithm see (Kätker and Paterok, 1994). A single loop free path servicę MO will be instantiated per route. The availability state of this service is defined by the underlying route determination algorithm. This behavior can be specified easily by use of virtual attributes.

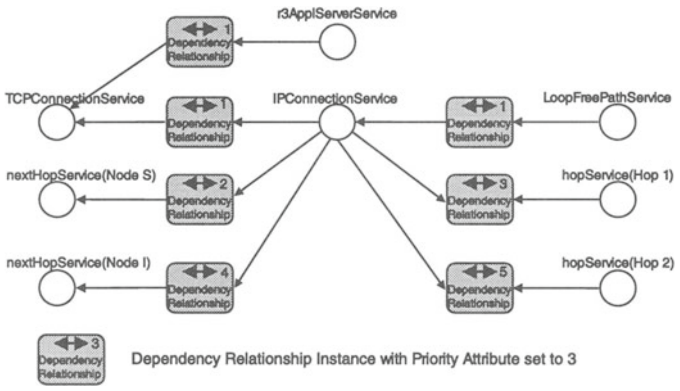

Figure 7 Integrated Model for the SAP R/3 Application using TCP/IP

Modeling the Hop Operational Dependency

As for the complete path dependency, one instance of a hop service will be created for each IP hop on the route. For each hop on the route a hop dependency and a hop relationship binding is created to link the hop services with the IP connection service MO. The hop service instance is defined by the two directly connected SAPs. The availability state of the hop service is up, if there is IP connectivity between the directly connected SAPs.

In order to ensure that the fault management application following the dependency chain will follow the right dependencies, ordering of the dependency relationships is necessary. If an IP connection is not available, the loop free path dependency must be checked first before the complete path and hop dependencies can be checked according to the order of the providing MOs in the route. The general rule is to check global route properties first, and after that the other dependencies according to the order of the provider MOs in the route.

The priorities noted in the dependencies shown in Figure 7 define the investigation order of the algorithm. If the dependencies of the IP connection service are inspected, the loop free path service is check first, after that the next hop service of node S and the hop service of hop 1 are checked. If all these services are available, the lower priority dependency relationships are investigated.

5 CONCLUSION

In this paper a modeling framework for distributed system fault management is presented. The major purpose of the distributed system model is to provide an integrated view of the resources and interdependencies of the distributed model. This enables generic fault management applications to provide automatic fault localization across layer boundaries of the distributed system. The approach uses relationship managed objects to represent dependencies between arbitrary objects of the distributed system. The generic management applications deal only with abstract service and relationship objects. They provide their services without specific knowledge of the distributed system details. The generic distributed system model provides important functionality for automated fault localization and event correlation. By following the dependency chain in the model fault management applications are capable of determining root faults across layers of a distributed system. New types of distributed applications or network protocols can be added to the DSM to provide fault management functionality without changing the generic fault management application. This helps increasing the availability of a system, minimizes the downtimes of distributed systems, and reduces the costs of distributed systems.

The example has demonstrated that the concept of the distributed system model is capable of representing complex environments in distributed systems in a unique and consistent way.

Although the focus of this paper was on fault management, the generic model can be extended in order to enable other management disciplines, like performance or configuration management, to perform their tasks in a generic way across layer boundaries of a distributed system. A more general and detailed framework for the definition of dependency relationships and the service model has to be investigated in order to provide this additional functionality.

ACKNOWLEDGMENTS

For continuing interesting discussions I want to thank Prof. Dr. Kurt Geihs, University of Frankfurt and the members of the IBM ENC Systems- and Network Management Department.

REFERENCES

S. Abeck, A. Clemm, U. Hollberg (1993) Simply Open Management: An Approach for the Integration of SNMP into OSI Management Concepts, IFIP TC6/WG 6.6 Symp. on Integrated Network Management, 361-75.

S. Bapat (1993) Richer Modeling Semantics for Management Information, IFIP TC6/WG 6.6 Symp. on Integrated Network Management, 15-28.

R. Buck-Emden, J. Gulimow (1995) Die Client/Server Technologie des SAP-Systems R/3, Addision-Wesley, Bonn, 2nd edition.

J.D. Case, M.S. Fedor, M.L. Schoffstall, J.R. Davin (1990) A Simple Network Management Protocol, Internet Activity Board, Request for Comments 1157.

D.E. Comer (1991) Internetworking with TCP/IP, Volume I; Principles, Protocols, and Architecture, Prentice Hall, Englewood Cliffs, New York.

A. Clemm (1993) Incorporating Relationships into OSI Management Information, 2nd IEEE Network Management and Control Workshop, Tarrytown, NY.

ISO/IEC 7498-4 standard Information Processing Systems - Open Systems Interconnection - Basic Reference Model - Part 4: Management Framework.

ISO/IEC 10165-4 standard: Information Technology - Open Systems Interconnection - Management Information Services - Structure of Management Information, Part 4: Guidelines for the Definition of Managed Objects.

ISO/IEC 10165-7 draft standard: Information Technology - Open Systems Interconnection - Structure of Management Information - Part 7: General Relationship Model.

ISO/IEC 10746 draft standard: Open Systems Interconnection - Data Management and Distributed Processing - Basic Reference Model of Open Distributed Processing.

S. Kätker, M. Paterok (1994) System zur Überprüfung eines Datenübertragungsnetzwerks, submitted for european patent, No GE 994 021.

S. Kätker (1995) Integration of Application and Network Fault Management, Proceedings of GI/ITG Workshop 2. Arbeitstreffen Entwicklung und Management verteilter Anwendungssysteme, Krehl-Verlag, Münster, ISBN 3-931546-00-4.

G. Pavlou, K. McCarthy, S. Bhatti, G. Knight (1995) The OSIMIS Platform: Making OSI Management Simple, Proceedings of the 4th International Symposium on Integrated Network Management, 480-93.

C. Popien, A. Küpper, B. Meyer (1994) A Formal Description of Open Distributed Processing (ODP) Trading Based on Guidelines for the Definition of Managed Objects (GDMO), Journal of Network and Systems Management, Vol. 2, No. 4, 383-400.

M. Sloman (1995) Network and Distributed Systems Management, Addison-Wesley, Wokingham/England.

M. Zimmermann (1992) Management of Distributed Applications, IFIP/IEEE Workshop on Distributed Systems, Operations and Management, München.

BIOGRAPHY

Stefan Kätker received his Diploma in Computer Science from the University of Erlangen-Nürnberg, Germany, in 1992. Since 1992 he is working in a research position at the IBM European Networking Center in Heidelberg. Research interests include SNMP and TMN based fault, application and distributed systems management.