FAULT MANAGEMENT IN QOS-ENABLED DISTRIBUTED SYSTEMS

Stefan Kätker[1] and Kurt Geihs[2]

[1]IBM Global Services, TMN Center of Competence
Vangerowstr. 18, D-69115 Heidelberg, Germany
[2]University of Frankfurt
P.O.Box 11 19 32, D-60054 Frankfurt, Germany
kaetker@de.ibm.com, geihs@informatik.uni-frankfurt.de

Abstract: Quality of Service (QoS) is an increasingly important concern in todays complex network environments. Delivering QoS guarantees requires an efficient fault management that is specifically tailored to the QoS handling. A model and an algorithm for fault isolation and event correlation in QoS-enabled distributed systems is presented. The model and the algorithm reflect the fact that the notion of a fault in QoS-enabled systems is different from the conventional fault model. The model concentrates on the service dependencies and incorporates different QoS levels. The algorithm reconstructs fault propagation during run-time by exploring the service dependencies.

Keywords: QoS management, service management, fault management, fault isolation

1 INTRODUCTION

With the increasing size and complexity of distributed systems in enterprises as well as in telecommunication environments, Quality of Service (QoS) concerns become increasingly important. QoS addresses the non-functional properties of a service, such as reliability, availability and performance. Delivering QoS guarantees is directly linked to an efficient fault management that is able to handle QoS violations in the context of other faults. Therefore, the management of complex, QoS-enabled distributed systems becomes a very complex and very cost intensive task.

L. Kutvonen et al. (eds.), *Distributed Applications and Interoperable Systems II*

The organizational structure of QoS-enabled distributed systems is such that service users negotiate QoS contracts, i.e. service level agreements, with the service providers. These agreements control the extend, type, quality, and the price of the service. Often penalties are also defined if the agreed service quality cannot be maintained.

Integrated network, system, and service management is a key technology for service providers to ensure the quality of their services and to satisfy the QoS guarantees. Fault management plays a major role to provide the availability and the quality of service in such a system. Today's management systems usually do not have specific tools to support service providers in managing the quality of the service they deliver from a user perspective. The target of most of these systems are the physical and logical resources that are part of a distributed system. In order to manage the QoS that is delivered to the customer, these systems must be able to explicitly represent the service relationships and properties.

This paper presents a modeling framework called Service Dependency Model (SDM) for a service-oriented modeling of distributed systems that explicitly incorporates the modelling of QoS. Different layers and components of the distributed system will be modelled using the same generic concept in order to provide an integrated, service-oriented model of the entire system. The primary goal of this model is to provide support for service-oriented fault management, e.g. the determination of the root cause of a service failure or the determination of the set of services affected by a faulty ressource in the network.

In the following we will make use of the service definition introduced by [1]: *The service delivered by a system is its behavior as it is perceived by the user(s); a user is another system (human or physical) which interacts with the former.*

A **failure** denotes the fact that a service does not deliver its quality as expected, i.e. specified in a service level agreement. A **symptom** is a failure that exists because of another failure in the system. In contrast, a failure is called a **root cause** if this failure is not caused by another failure in the system. **Fault isolation** is the process of identifying the root cause for a given symptom. **Event correlation** groups all problem indicators (i.e. events) whose generation have been caused by the same root cause together.

Section 2 discusses the related work in the two different areas fault isolation/event correlation and service oriented modeling. Section 3 summarizes the requirements for the fault management model before the Service Dependency Model is introduced in Section 4. Section 5 describes an algorithm for fault isolation based on the SDM. An example illustrates the principles and application of the algorithm. The paper ends with concluding remarks in Section 6 that contain references and summaries of two field studies that have been performed applying the SDM to different distributed systems.

2 RELATED WORK

Most known techniques for **fault isolation and event correlation** can be classified by one of four different approaches. Subsequently, these approaches are sketched, the key references are mentioned and the advantages and drawbacks are discussed.

Many published concepts and implementations use *artificial intelligence methods* for fault management. To overcome the lack of structure in the rule bases, the second generation of expert systems, called model based reasoning systems [5, 10] use different techniques to represent structural knowledge (using object-oriented models) and heuristical knowledge (using rules).

Fault propagation models describe which symptoms will be observed if a specific fault occurs. [11] for example use the concept of causality graphs to decode the set of observed symptoms. An event modeling language is introduced and an example is given how QoS problems can be handled by the system. *Model traversing techniques* reconstruct fault propagation during run-time by using relationships between managed objects. Based on an object-oriented network model an event correlation system for the physical layer of large telecommunication networks is presented by [4]. Our work in this area has concentrated on event correlation based on object model traversing in Telecommunication Network Management (TMN) systems [6], generic traversal of network layer models [9], and traversal of service dependency graphs [7, 8]. The technique described in this paper is an extension of the service dependency graph based model traversing technique to QoS management.

A similar approach is followed by Gruschke [3] for event correlation in distributed systems, where dependencies between resources are modeled and event correlation is carried out without online interaction with the managed resources. Gruschke's model focusses on availability aspects of distributed systems.

Recently, new approaches for **modeling distributed systems** have been introduced that aim at providing a consistent, high level view on the system to allow management applications to work independently of the heterogeneous, often proprietary models available for different applications and aspects in distributed systems. [13] suggests an architecture based on ODP concepts to hide the specifics of the proprietary models. [12] present a modeling approach for telecommunication networks based on functional dependencies between resources, i.e. sub-networks, in the system. Meta Managed Objects are introduced in [14] to integrate different network models.

The TINA Service Architecture [2] provides means to model services in a telecommunication environment explicitly. The major focus of TINA is not the management of these services itself, but an environment for rapid development of new telecommunication services.

The service model in this paper follows a more pragmatic approach focused on the management of telecommunication services.

The literature survey has shown that appropriate techniques exist for specific areas of fault management. These tools have advantages and disadvantages that make them particularly useful for specific tasks. A high level integration model that provides support for fault management and specifically fault isolation for QoS-enabled distributed systems has not been introduced. Most of the techniques and tools providing fault isolation and event correlation today operate on resource oriented models. Abstract integration models for distributed systems also focus on resources. In order to allow monitoring and management of service level agreements a service oriented view on the distributed system is needed.

3 REQUIREMENTS

In this paper we present an abstract model for distributed systems best suited to support automatic fault isolation including QoS violations. This section lists the key requirements for such a model:

1. Service oriented

   The services that are provided by a distributed system must be the key target for modeling, not the resources that are contained in the system. If a user and QoS oriented view of the system should be provided, services and service level agreements must be modeled explicitly.

2. Generic model

   A generic model enables generic management applications to perform their tasks without specific knowledge about particular resource properties and attributes.

   Thus, fault isolation can be performed without knowledge about a certain resource. The model hides the specifics of the managed resources. It is not required to change the management application to provide fault isolation functionality for a new type of network protocol or hardware. Only the appropriate object models and agents providing these models have to be developed.

3. Abstract, integrated model

   All aspects and layers of the distributed system relevant to fault management have to be represented in one consistent, generic model of the system. Service layer, network layer, and network element layer aspects need to be represented in the model using a unique and consistent modeling scheme.

4. Integrate existing information models

   Information already available in specific information models, e.g. for specific applications like relational databases must be utilized and integrated into the generic model.

While realizing this model it is important to select the appropriate level of abstraction. On the one hand this level should be fairly high to limit the complexity of the fault management system. Using a high abstraction level enables the modeling of all layers of the system and their dependencies in an efficient way. On the other hand fault isolation and diagnosis is a very component specific task. Detailed knowledge and information about the resources, their behaviour, fault models, and test and repair capabilities might be needed.

4 SERVICE DEPENDENCY MODEL

The basic concepts of the Service Dependency Model (SDM) have been introduced in [7, 8]. Here these concepts will be extended to QoS Management.

In the SDM a distributed system is represented by a set of services and a set of dependency relationships. The services represent key services provided by the different

layers of the system. System internal services as well as services visible to the outside are modeled. A service is for example a communication connection provided by a network, CPU time provided on a computer, or a complex application service like a teleconference. Dependency relationships link the different service instances in the model. A dependency relationship is instantiated in the service dependency model, if a service s_1 depends on another service s_2 to provide its service.

A teleconferencing application for example depends on the ATM network connectivity service between the different partners of the teleconference to provide its service to the end user.

If two services s_1 and s_2 are linked with a dependency relationship, the underlying fault model assumes that the failure of service s_1 is caused by service s_2, if s_2 fails as well. The failure of s_2 is called the local root cause of the failure of s_1, if $s1$ and s_2 are directly linked by a dependency relationship.

If the teleconference application does not provide its service and the ATM connection does not provide its service either, it will be assumed that the failure of the ATM connection is the local root cause of the failure of the teleconference application.

The basic model needs further refinement in order to meet the requirements for service-oriented, integrated QoS Management. The following modeling constructs will be introduced:

- Quality level
  Describes the QoS that is provided in a unique and generic fashion.
- Quality view
  Structures the set of quality levels according to certain criteria, e.g. performance or availability.
- Dependency relationship with quality levels and views
  The extended concept of dependency relationships that is aware of quality levels and views.

Before we discuss further details of the modeling concept, the integration of this model with the environment will be sketched by the architecture shown in Figure 1.

The SDM serves as an interface between the generic management applications and the specific information models for the specific parts or layers of the system, e.g. the MIBs for the teleconference application or a specific ATM network management MIB. In general these specific MIBs are called system specific resource models in the following.

4.1 Quality Level

In the base concept only two different states of a service were known. A service could either be enabled (no failure) or disabled (failure). For QoS management a discrete, multi element service state set is needed to be able to model several levels of service degradation.

Each service in the system will be measured according to a quality level. The set of quality levels is finite and well ordered. Using this generic set of quality levels it is

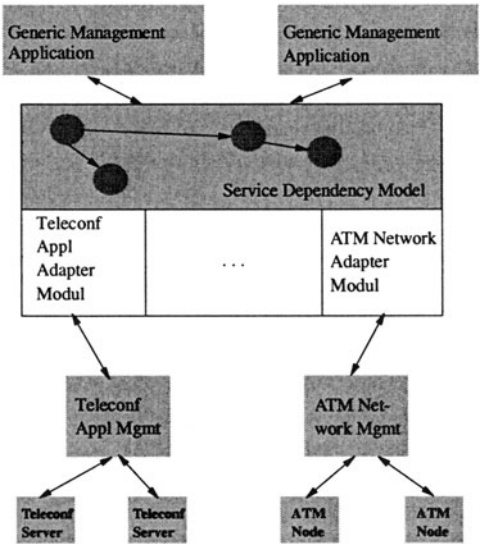

Figure 1. Integration of the SDM in the Environment.

possible to compare QoS of different layers or components in the system, e.g. the ATM network layer and the teleconference application layer. A specific quality level for a service will be computed from the attributes of a service object by applying a **quality level function**. Quality level functions are specific for specific service classes. By using these functions very specific QoS descriptions can be transformed into generic service quality levels. Quality level functions are the major interface between the generic SDM and the system specific resource model.

Figure 2 shows the concept of the quality level function. The QoS is translated to a quality level by applying a quality level function. Input for this function are attributes of a service that describe the QoS specifically for this type of service. Output is a generic quality level. The quality level function provides a mechanism to evaluate the QoS of a service in a generic fashion.

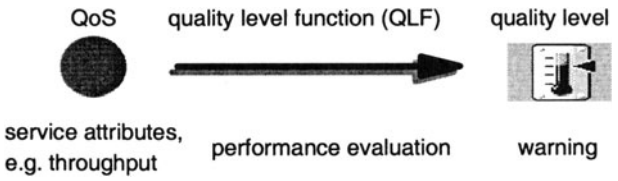

Figure 2. Mapping specific QoS to generic quality levels.

The concept of quality levels and quality level functions will be illustrated using the known example of the teleconferencing system. The QoS of the application is defined based on user requirements by parameters like picture and sound quality (e.g.

picture frequency and picture sound synchronization). On the ATM network layer parameters like throughput, jitter, and delay are used to characterize QoS. In order to enable a generic management application to evaluate the QoS without requiring specific knowledge about the details of the QoS definition on each layer, these definitions are transfered into generic quality levels. In this example three quality levels (`normal`, `warning`, `critical`) are used. The following functions q_{ATM} and q_{TCA} define the quality level functions for the ATM connection service and the teleconferencing service respectively.

$$q_{\mathrm{ATM}} = \begin{cases} \texttt{normal} & \Leftrightarrow & \texttt{throughput} > 128 \wedge \texttt{delay} < 0.1 \\ \texttt{warning} & \Leftrightarrow & 64 \leq \texttt{throughput} < 128 \vee 0.1 < \texttt{delay} \leq 0.5 \\ \texttt{critical} & \Leftrightarrow & \texttt{throughput} < 64 \vee \texttt{delay} > 0.5 \end{cases}$$

$$q_{\mathrm{TCA}} = \begin{cases} \texttt{normal} & \Leftrightarrow & \texttt{frequency} > 20 \\ \texttt{warning} & \Leftrightarrow & 10 \leq \texttt{frequency} < 20 \\ \texttt{critical} & \Leftrightarrow & \texttt{frequency} < 10 \end{cases}$$

Figure 3 shows the difference between usual QoS mappings and the mapping described in this paper. Usually QoS parameters are mapped from layer n to layer $n-1$, see e.g. [15]. Here the QoS parameters from all layers are mapped to a common, generic set of quality levels.

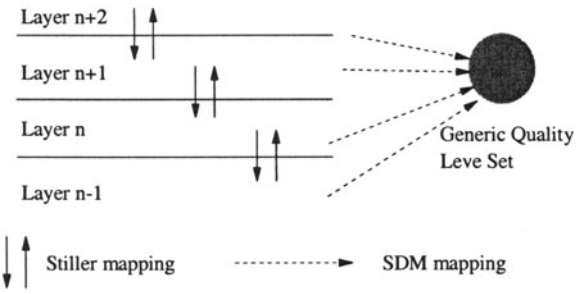

Figure 3. Comparison of QoS mapping concepts.

Quality level functions are not part (or member functions) of service objects primarily for two different reasons. First, more than one type of evaluation should be possible for a specific service class. For example, it should be possible to evaluate the performance and the availability of a specific service using two different service quality level functions. Second, service objects are usually specific to certain components or layers of distributed systems. In order to avoid double instantiation of service objects in the system specific resource model and the SDM it is desirable to have SDM specific pieces of service objects separated from the service objects themselfs. Using this mechanism objects representing the resources that provide services can usually be reused as objects representing the service. The quality service level function can act

as the primary interface between the system specific resource model and the generic SDM.

4.2 Quality Views

The purpose of a quality view is to group certain quality level functions for different services classes that evaluate the service classes using the same criteria. A performance view for example will group all quality level functions that evaluate the performance of a service object while an availability view will contain all service quality level functions that evaluate the availability of a service. All service quality level functions belonging to the same view must have the same image.

Figure 4 shows the performance and the availability view on the ATM connection and the teleconferencing application from the example above. Although the QoS parameters for these two components of the distributed system are different (ATM uses for example jitter, delay, and throughput as base parameters, while the application asks for interruption free video and clear voice transmission), the service quality levels are comparable. For each service class a different service quality level function is

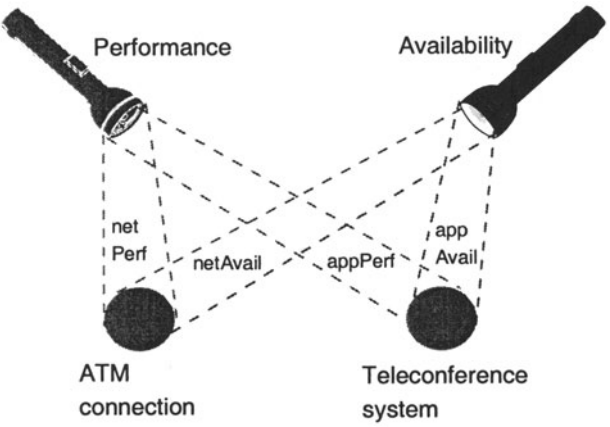

Figure 4. Service View Example.

used for a certain view. A single service can therefore be evaluated under different circumstances using different views. In the example in Figure 4 the service quality level functions `netAvail` and `appAvail` are belonging to the availability view, while the functions `netPerf` and `appPerf` belong to the performance view. Formally, a view is given by a set of tuples of service classes and service quality level functions. Each service class that should be treated by the view is listed in the set with it corresponding service quality level function.

4.3 Dependency Relationships using Quality Levels and Quality Views

Dependency relationships link a service s_1 and a service s_2 if s_1 depends on s_2 to provide its service. Dependency relationships must take into account the view and the quality level of both services. In light of this requirement, a dependency relationship must specify the following two basic parameters:

1. What is the quality requirement on the service in the provider role? The view and expected service quality level must be given.
2. What are the quality characteristics of the service in the user role that can be provided by this service, if the service in the provider role meets its quality requirement. Here, also view and quality level must be specified.

Using these two parameters it can be easily specified that a teleconferencing service can only provide quality level `normal` under the performance view, if the underlying ATM connection service provides at least the quality level `warning` under the performance view.

The following attributes are needed in order to specify these parameters in dependency relationships.

User role: A service object in the user role represents the user of a service.

User view: The user view defines the view under which the service in the user role will be evaluated.

Quality goal: The quality goal is the quality level that the service in the user role could potentially provide, if the service in the provider view would meet its quality requirements.

Provider role: This attribute holds a reference to the service object in the provider role.

Provider view: The provider view defines the view under which the service in the provider view will be evaluated.

Quality requirement: The quality requirement describes the quality level that the service in the user role expects from the service in the provider role under the view given in the provider view.

A dependency relationship with the above attributes has the following semantics: The service in the user role can only provide the quality level given in the attribute quality goal under the view given in the user view, if the service in the provider role provides at least the quality level given in the quality requirement attribute under the view given in the provider view attribute. Figure 5(a) shows the graphical representation of the dependency relationship with all its attributes.

The dependency relationship given in Figure 5(b) describes that the teleconferencing service a (in the user role) can only provide the quality level `normal` under the

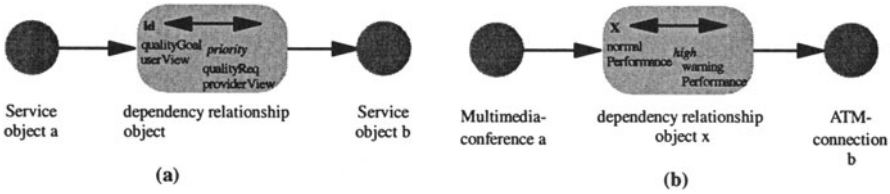

Figure 5. Graphical representation of a dependency relationship (a) and example (b).

performance view, if the ATM connection service *b* (in the provider role) provides at least the quality level `warning` under the performance view.

A dependency relationship serves as a formal method to model service level agreements.

Example

In this section we will illustrate the SDM concepts using the example of a simple teleconferencing application. Consider a teleconferencing application that uses an ATM network to connect several clients that participate in a teleconference with the coordinating teleconferencing server. For the purposes of this example we assume that the performance of the teleconferencing server depends on the CPU utilization and the utilization of the paging space in the machine that runs the server. In addition, teleconferencing clients depend on the server and on the ATM connection between the client and the server. Figure 6 shows the SDM of this configuration.

Dependency relationships 5 and 6 are of particular interest. Here, the user view and the provider view are different. These examples show how the fault isolation algorithms using the SDM can change the view while they are traversing the SDM. The CPU and the paging space services are evaluated using a utilization view in order to provide a common view that is useful for availability (100 % utilization means no availability) and performance (more than about 70 - 80 % utilization means reduced performance) evaluation.

5 THE FAULT ISOLATION ALGORITHM

Starting point of the fault isolation algorithm is the notification about a failure in the system. Goal of the fault isolation is to determine the root cause of the failure.

A failure in the system in the sense of the SDM is given by a triple of service object, view, and quality level function that is no longer provided by the service object under the given view. If a teleconferencing application should provide the quality level `normal` under the performance view, but does only provide the level `warning` this is considered as a failure.

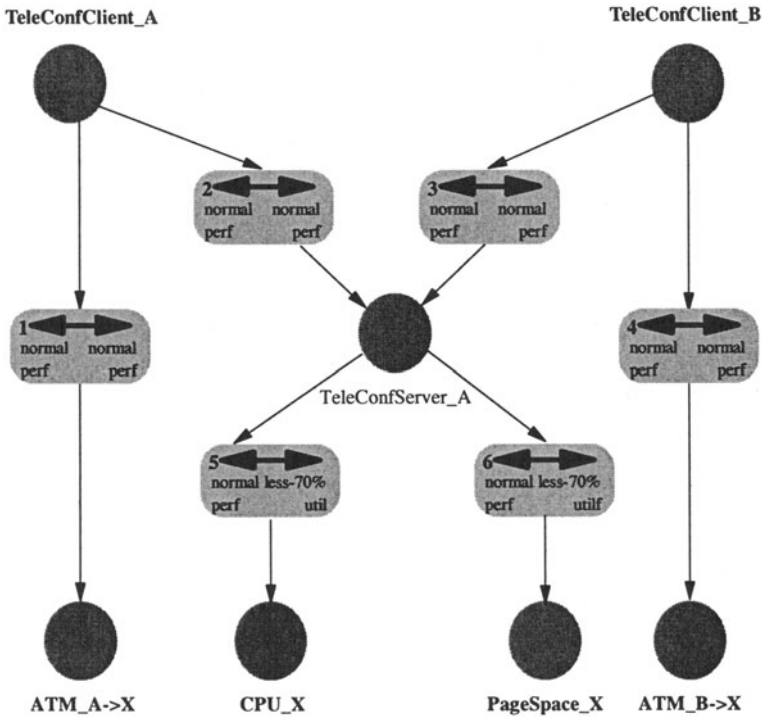

Figure 6. Service Dependency Model of the Teleconferencing System.

Figure 7 shows the details of the algorithm, where $\mathbf{a} = (s, v, \omega)$ describes a failure (service s does not provide quality level ω under view v. $U_l(\mathbf{a})$ is the set of local root causes for the failure $\mathbf{a}$.

Based on this information the algorithm starts to determine the dependency relationships that match the failure description, i.e. where the failing service matches the user role, the view given in the failure description matches the user view and the quality level that was not met matches the quality goal. After that, the algorithm checks the provider services of all these dependency relationships. It is checked, whether the service in the provider role currently provides the quality level given in the quality requirement under the view given in the provider view. In order to check this condition, the quality level function corresponding to the class of the service object in the provider role is calculated from provider view. After that, this quality level function is executed for the service object in the provider role.

If a service in the provider role does not provide the required quality level, the search will be continued with this service object. If more than one underlying service does not meet the quality requirement, it is assumed that multiple root causes exist for the symptom and the search will be continued with all failing services in parallel. If

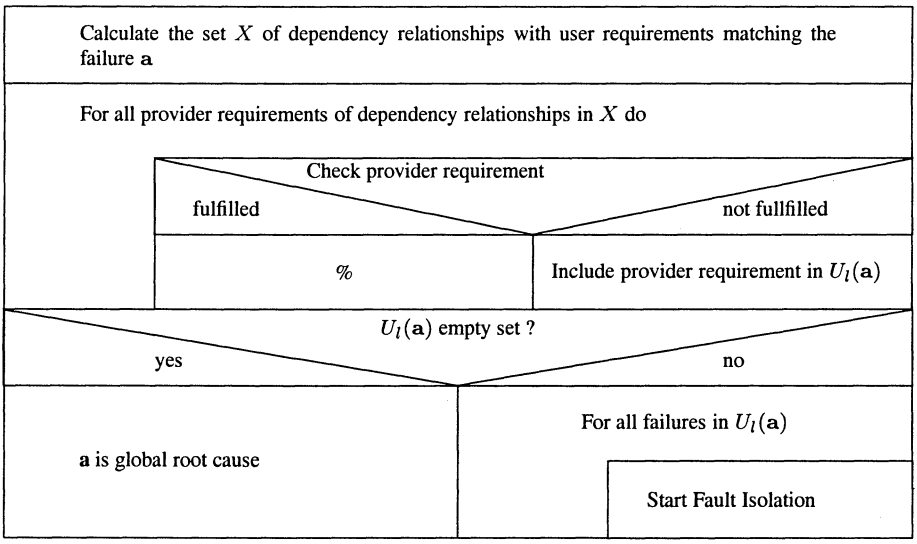

Figure 7. Fault Isolation Algorithm.

no underlying service fails or there do not exist matching dependency relationships the current failure under investigation is considered to be a root cause.

Example

The fault isolation algorithm will be illustrated in this section using the example SDM shown in Figure 6.

Consider that the CPU utilization of the CPU in node X is more that 80 %. As a consequence, the teleconference server running on node X cannot provide optimal performance. This also impacts the performance of the teleconferencing clients running on node A and node B.

The algorithm start with the notification of the failure, i.e. that the teleconferencing client does not provide quality level `normal` under the performance view $\mathbf{a} = (\texttt{TeleConfClient\_A}, \textit{perf}, \texttt{normal})$. The query for dependency relationships that match this failure provides relationships 1 and 2. The provider sides of these relationships are investigated. The ATM_A→X service is tested by applying the appropriate quality level function. It shows that the service provides the quality level `normal` under the performance view, so it cannot be the local root cause for the failure **a**. Investigation of the service `TeleConfServer_A` shows that it does not provide the required quality level `normal` under the performance view. The failure $\mathbf{b} = (\texttt{TeleConfServer\_A}, \textit{perf}, \texttt{normal})$ is therefore considered as the local root cause for the failure **a**. The algorithm continues with the investigation of failure **b**. By looking at dependency relationships 5 and 6 and their respective provider sides it is

determined that the failure $\mathbf{c} = (\texttt{CPU\_X}, \texttt{util}, \texttt{less} - 70\%)$ is the global root cause for failures **a** and **b**.

The algorithm terminates here since there are no dependency relationships defined with service CPU_X acting in the user role.

6 CONCLUSION

Providing applications with their desired quality of service will be a key requirement in future distributed environments. QoS management includes the handling of QoS violations. In this paper we have discussed this topic in the context of general fault management. QoS introduces new requirements to fault management systems since the conventional fault/non-fault model is not sufficient for QoS attributes that may assume different QoS levels which may be more or less acceptable from the user perspective. The presented generic service dependency model captures the specific characteristics of QoS management. Based on the model we have developed an algorithm for the automatic detection of root faults in QoS-enabled distributed environments.

The SDM has been used in two field studies (SAP R/3 application and Web based Tele-Education system on top of a ATM VPN) to provide integrated, service-oriented fault management. Both projects have successfully underlined the feasibility of the approach. The SAP R/3 prototype used the TMN architecture as a basis for the SDM implementation, while the Tele-Education system was modeled using a CORBA based implementation of the SDM.

QoS management has only recently become a major focus of distributed system research. Most of the QoS developments so far have occurred in the networking arena. From the viewpoint of network and system management QoS handling introduces additional requirements that demand an integrated management approach. It will take much more research and application experience to fully understand the technical implications of QoS-enabled distributed systems.

References

[1] CARTER, W.C.: A time for reflection, *Proceedings of 12th IEEE International Symposium on Fault Tolerant Computing (FTCS-12)*, 41, 1982.

[2] CONCHON, A., HELLEMANS, P.: TINA Service Architecture, Alcatel *Telecommunications Review*, No. 1, pp. 68-74, 1998.

[3] GRUSCHKE, B.,: A New Approach for Event Correlation based on Dependency Graphs,*Proc. of the 5th Workshop of the Open View University Association: OVUA'98.*

[4] HOUCK, K., CALO, S.B., FINKEL, A.: Towards a Practical Alarm Correlation System, *Proc. 4th IFIP/IEEE International Symposium on Integrated Network Management*, 519-30, 1995.

[5] JAKOBSON, G. AND WEISSMANN, M.D.: Alarm Correlation, *IEEE Network*, pp. 52-59, 1993.

[6] JORDAAN, J.F. AND PATEROK, M.: Event Correlation in Heterogeneous Networks Using the OSI Management Framework, *Proc. ISINM'93, IFIP TC6/WG 6.6 International Symposium on Integrated Network Management*, 683-95, 1993.

[7] KÄTKER, S.: A Modelling Framework for Integrated Distributed Systems Fault Management, *Proc. IFIP/IEEE International Conference on Distributed Platforms*, 186-98, 1996.

[8] KÄTKER, S. AND GEIHS, K.: A Generic Model for Fault Isolation in Integrated Management Systems, *Journal of Systems and Network Management – Special Issue on Fault Management*, Vol. 5, No. 2, 109-30, 1997.

[9] KÄTKER, S. AND PATEROK, M.: Fault Isolation and Event Correlation for Integrated Fault Managenent, *Integrated Network Management V – Integrated management in a virtual world*, Chapman and Hall, London, 583-96, 1997.

[10] KEHL, W. AND HOPFMÜLLER, H.: Model-Based Reasoning for the Management of Telecommunication Networks, *Proc. ICC'93, IEEE International Conference on Communications*, 13-17, 1993.

[11] OHSIE, D., MAYER, A., KLIGER, S., YEMINI, S.: Event Modeling with the MODEL Language, *Integrated Network Management V – Integrated management in a virtual world*, Chapman and Hall, London, 625-37, 1997.

[12] MEIRA, D.M. AND NOGUEIRA, J.M.S.: Modelling a Telecommunication Network for Fault Management Applications, *IEEE Network Operations and Management Symposium (NOMS' 98)*, 723-732, 1998.

[13] NEUMAIR, B.: Distributed Applications Management based on ODP Viewpoint Concepts and CORBA, *IEEE Network Operations and Management Symposium (NOMS' 98)*, 559-569, 1998.

[14] SEITZ, J.: Meta Managed Objects, *Integrated Network Management V – Integrated management in a virtual world*, Chapman and Hall, London, 650-60, 1997.

[15] STILLER, B.: Hierarchical Mapping of Enhanced QoS Parameters Based on OSI Protocols, *Third IEEE Workshop on the Architecture and Implementation of High Performance Communication Subsystems (HPCS'95)*, 1995.

Biographies

Stefan Kätker received his Diploma in Computer Science from the University of Erlangen-Nürnberg, Germany, in 1992. Since 1992 he is working in the Systems- and Network Management Department at the IBM European Networking Center in Heidelberg. Currently he is architect and scientific consultant for TMN fault management products and solutions. Research interests include SNMP- and TMN-based fault, application, and distributed systems management.

Kurt Geihs is a professor for Computer Science at the University of Frankfurt. His research and teaching is focussed on distributed systems and operating systems. Current research projects concentrate on network and system management, service trading in open distributed systems, and performance modelling and analysis.

Before joining the university he worked for IBM at the IBM European Networking Center in Heidelberg, Germany. His research areas were network operating systems, open distributed processing and system management. In 1988-89 he was on assignment to the IBM Thomas J. Watson Research Laboratory in Hawthorne, New York, developing software for high-speed network attachments.

Prof. Geihs received a "Diplom-Informatiker" degree from the Technical University Darmstadt, Germany, a M.S. in Computer Science from the University of California, Los Angeles, California, and a Ph.D. from the Technical University Aachen, Germany.