A GENERIC MODEL FOR IT SERVICES AND SERVICE MANAGEMENT

Gabi Dreo Rodosek
Leibniz Supercomputing Center
Barer Str. 21, 80333 Munich, Germany
dreo@lrz.de

Abstract: Whereas network and system components were in the focus of management research in previous years, nowadays *management of services* dominates management activities. We are witnessing a paradigm shift from *device-oriented* to *service-oriented* management, and with this the need to deal with new challenging management issues. The management of the underlying infrastructure with respect to the delivered services and agreed service level agreements is certainly the fundamental challenge. It is easy to see that all research questions center around the new managed object *service* and its integration with existing device-oriented managed objects (network devices, end systems, applications). Thus, the development of a common definition of a service in terms of a *common generic service model* is essential.

Keywords: Service models, Information Modeling, IT Service Management

1. Introduction

Due to the significant increase in the complexity of enterprise applications and the need to offer distributed *IT services,* we are witnessing that the management focus has turned away from device-oriented to service-oriented management. This does not mean that device-oriented management is not of importance any more. On the contrary, an efficient device-oriented management is a precondition for an efficient IT service management. However, device-oriented management was exclusively in the domain of the provider and it was driven by the objectives of the provider. Nowadays, it is necessary to manage the underlying infrastructure with respect to services offered to customers and agreed service levels agreements. The complexity of IT service management becomes evident with the necessity to cope with service dependencies and the distributed provision of a service upon several resources.

The fundamental issue of IT service management is the development of a common definition of a service in terms of a *common service model.* Despite of an amount of existing service definitions (e.g., 9, 3, 5, 13, 6) a common understanding of a service that provides a unified approach to support the concepts of service management is lacking. A first step towards a *generic service model* has been proposed by the Service Management Task Force (SMTF), a group of researchers of the Munich Network Management team, in 4. We take this service model as a basis for the development of the *common generic service model* that provides the information basis upon which the deployment of service management applications can be approached.

The paper proceeds as follows: Section 2 gives an overview of the methodology for service modeling. The service-centric aspect is addressed with the service template model in section 3. Section 4 addresses the provider-centric aspect by introducing the provider-centric service template model, and section 5 proposes the customer-centric service template model. An example of the applicability of the proposed models is given in section 6. Finally, section 7 concludes the paper.

2. Methodology for Service Modeling

The business reference model, which provides the basis of our discussion, is visualized in Fig. 1, and identifies three aspects that need to be addressed: (i) the service-centric, (ii) provider-centric and (iii) customer-centric aspect of service modeling. The role of a customer refers hereby to an organization whereas the role of a user refers to a individual user. The further discussion refers only to the customer role.

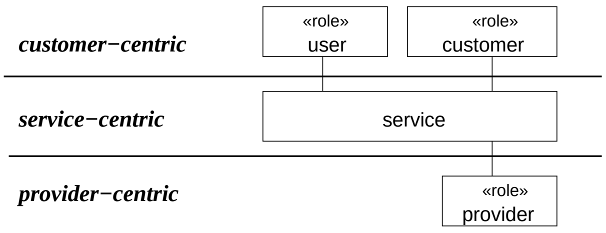

Figure 1. Business reference model (UML notation)

In the following, the identified aspects are addressed in more detail:

Service-centric part. This part refers to the elements of a service which are independent of any provider- or customer-centric issues. The main elements of the service-centric part are as follows. Firstly, the specification of the "abstract" *service functionality* needs to be approached. The service functionality with respect to the service hierarchy consists of two elements: (i) its own functionality and (ii) the functionality of its sub-services. Secondly, the specification of the "abstract" *quality* of the provided service functionality needs to be addressed. Quality of the provided functionality is measured and expressed with *service-centric QoS parameters*.

Provider-centric part. This part addresses the point that services, respectively the service functionality, can be provided in different ways by different providers (e.g., specific policies, specific service provisioning). Primarily, this part addresses the aspect of service provisioning and service operation. Elements of the provider-centric part are: (i) *steps* how to provide, operate, and withdraw services, (ii) the *quality* of services as offered by providers and (iii) the *policies* how to operate services. The workflow aspect is associated with the steps because several persons need to work together on service planning, provisioning, operation, change, and withdrawal. Quality of the provided services is measured with *provider-centric QoS parameters.* These parameters refine service-centric QoS parameters with value ranges that are defined by providers. In general, the specification of values for provider-centric QoS parameters is a trade-off between quality, cost and market demands.

Customer-centric part. A service as provided by a service provider can be offered to various customers. One of the most important elements of a customer-centric part are *customer-centric QoS parameters.* A customer has the possibility to select specific values from the offered provider-centric QoS parameters, respectively the value ranges.

We will proceed with a detailed specification of the attributes for the three identified parts of a service. A systematical identification of the attributes and their refinement requires to follow a *methodology for service modeling.*

The methodology for service modeling specifies the steps necessary for the development of the service model to address the service-, provider- and customer-centric aspects of a service. As depicted in Fig. 2, these steps are as follows:

1 The specification of the **service-centric** part of a service which is addressed by the development of the **service template model**. Often, the abbreviation **STM (service template model)** will be used.

2 The specification of the **provider-centric** part which is addressed by the development of the **provider-centric service template model**. Often, the abbreviation **provider-centric STM** will be used.

3 The specification of the **customer-centric** part of a service which is addressed by the development of the **customer-centric service template model**. Often, the abbreviation **customer-centric STM** will be used.

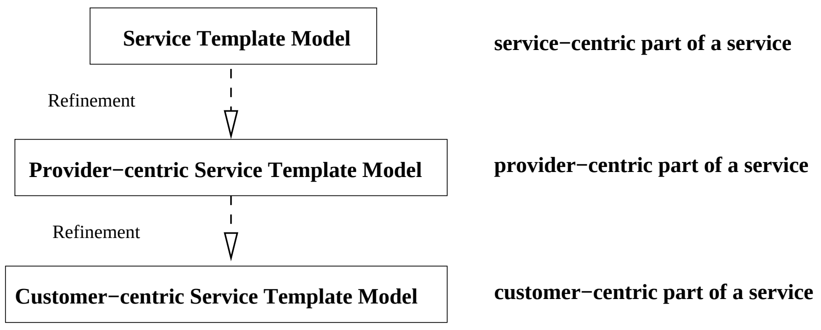
Service Template Model
service−centric part of a service
Refinement
Provider−centric Service Template Model
provider−centric part of a service
Refinement
Customer−centric Service Template Model
customer−centric part of a service

Figure 2. Methodology for service modeling

The order of the steps is defined by the fact that a service is an "abstract" functionality which needs to be realized (i.e., provided) by a provider. Different providers may provide the same services differently because of for example environmental specifics, policies, type of customers. Different customers may subscribe to the offered services. This is visualized in Fig. 2 with the introduction of the three models. The relation between the service models is defined such that (i) the *service template model* is refined to a *provider-centric service template model*, and (ii) that the *provider-centric service template model* is further refined to a *customer-centric service template model*.

Covering various aspects of a service (service-, provider- and customer-centric) with various models gives us flexibility and addresses with this the requirement of providing the same service by different providers to different customers.

To clarify the meaning of the introduced service models, we extend the model layer with the instance layer: a service template model is instantiated to a *service template instance,* a provider-centric service template model to a *provider-centric service*

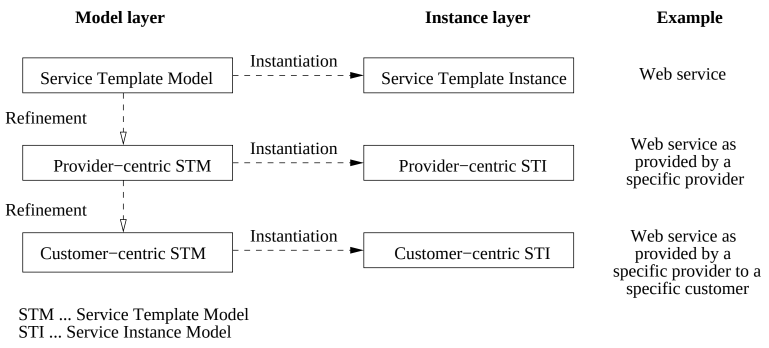

Figure 3. Model and instance layer explained on the Web service example

template instance, and the customer-centric service template model to the *customer-centric service template instance*. This is visualized in Fig. 3, where horizontally the instantiation of models to instances is visualized, and vertically the refinement of models. Let us explain the meaning of the model and instance layer on an example. An example of the service template instance is the Web service. An example of a provider-centric service template instance is if the "Web service is provided by a specific provider (e.g., Leibniz Supercomputing Center)". Furthermore, the "Web service customer-centric STI" layer represents in such a case a Web service as provided by a specific provider (e.g., Leibniz Supercomputing Center) to a specific customer (e.g., University of Munich).

As identified by the methodology, the first model to deploy is the *service template model.*

3. Service Template Model

The **service template model** addresses the elements of the service-centric part of a service (i.e., the service part which is independent of any customer- or provider-centric issues). To identify this service-centric part, the starting point of our discussion is the definition of a **service**.

We define a *service as a* **functionality** *that is provided with a certain* **quality** *and* ***cost*** *at a Service Access Point* **(SAP)**. Elements of the service-centric part are discussed in the following.

Functionality of a Service. The functionality of a service consists of two parts: (i) the functionality of the service itself, and (ii) the functionality of sub-services that are involved in the service provisioning with respect to the service hierarchy. As a consequence, *service dependencies* are another element of the service-centric part of a service which are addressed later on. A quite common approach to describe the service functionality is with service functional building blocks. A precise specification of the service functionality, respectively the service functional building blocks, requires to include *functional parameters* into the service template model as well. Functional parameters are associated with service functional building blocks. For example, in order to send an email, it is necessary at least to specify the email address of the recipient (i.e., *send_email(EmailAddress)*). Functional parameters can be in general of all types as used in programming languages. An example is a character string or a sequence (e.g., Email address).

Service Access Point. In the customer-provider scenario a SAP is the point at which service functionality is accessed by a customer and provided by a service provider. The SAP definition of the OSI reference model cannot be used directly for our purposes, because such a strict layering of services is not existent in a service hierarchy. Services on the same layer can use other services on the same layer. Therefore, we define a SAP as *a point, where a service requests the functionality of another service or vice-versa provides its functionality to another service (regardless of any layering of services).* From the implementational point of view, a SAP can vary from a simple router interface to a complex application call. A router interface is an example of a SAP that is used by an ISP (Internet Service Provider) whereas an application call is an example for a SAP that is used typically by an ASP (Application Service Provider). To distinguish between these kinds of SAPs, we introduce the term *type* of a SAP. Another issue is also the necessity to identify the *location*, respectively the resource that realize the SAP.

Quality of Service (QoS) Parameters. These parameters are used to measure the quality of the provided service functionality at the SAP. Service functionality can be measured with one or more QoS parameters, and vice-versa a QoS parameter may measure the quality of various services (e.g., availability). There exists a many-to-many relationship between a QoS parameter and a service.

QoS parameters can be classified in general into (i) qualitative and (ii) quantitative parameters. Qualitative parameters express the quality in ranges such as gold, silver and bronze or yes and no, aggregating several quantitative or qualitative parameters under each term. Quantitative parameters measure the quality of parameters in concrete quantities and values (e.g., availability of 99,98%). A precise specification of QoS parameters is one of the most important issues of quality management and includes several elements, as depicted in Fig. 4. First, it is necessary to identify relevant QoS parameters for a service and specify the semantics of these parameters. Additionally, it is necessary to specify also the value type and the possible value ranges of the parameters. The value type specifies that a parameter is measured for example in percentage and that the value range may be from 0% till 100%. In the case a service depends on other sub-services, a QoS parameter (e.g., QoS3 in Fig. 4) is an aggregated parameter of the basic parameters QoS1 (i.e., a QoS parameter of sub-service i) and QoS2 (i.e., a QoS parameter of sub-service j). To distinguish between basic and aggregated QoS parameters, we introduce the term *parameter type.* Basic QoS parameters are aggregations of QoD parameters (i.e., MIB variables).

The description of QoS parameters includes the following elements (as visualized in Fig. 4): (i) specification of the semantics of a QoS parameter, (ii) specification of the value type and value range, (iii) specification of the parameter type (basic or aggregated), (iv) definition of the calculation metrics (i.e., how QoS parameters are aggregated or deduced from other QoS parameters or from QoD parameters), and (v) identification of Quality of Device parameters (QoD). The identified elements are analyzed in the following in more details:

Semantics of QoS parameters. An explicit and precise description of the semantics of QoS parameters from the customer's and provider's perspective is essential to omit possible misunderstandings. In most cases, such a description is per-

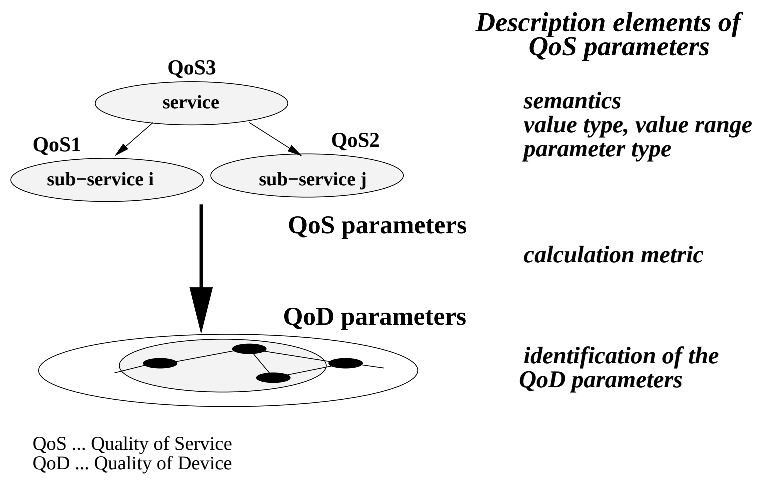

Figure 4. Description elements of QoS parameters

formed in a free-form text although a more formal description would be more appropriate.

An alternative approach to describe the semantics in more details is to express the service quality with several QoS parameters. An example should clarify this statement. Assume that a customer and a provider agree on the QoS parameter *availability* with a specific value. This value can be met if a service is unavailable once for a long time period or the service is unavailable very often for short time intervals. In the first case, such a long unavailability of a service may cause more serious problems than the short outages. Thus, it would be appropriate to specify additionally to the *availability* also that the parameter *MTTR* (Mean-Time-To-Repair) should not exceed a certain time interval.

Value type and value range. A further point of discussion is the description of the value type and with this associated the value range of a QoS parameter. Similar, as with the definition of the types of functional parameters, types of QoS parameters can be those used in programming languages. In some cases, the value range can be derived from the value type. For example, a common value type for the QoS parameter *availability* is to express this parameter as a percentage value. The derived value range is from [0% - 100%]. In some cases, however, such an automatic derivation is not possible, and therefore it is necessary to separate between the value type and the value ranges of QoS parameters. For example, the value range for the QoS parameter *Customer Satisfaction Index (CSI)*, measuring the quality of the hotline support from the customer's perspective, can be specified in various ranges.

Parameter type. QoS parameters can be of the following types with respect to the previous discussion: (i) basic or (ii) aggregated. Basic QoS parameters are calculated directly from QoD parameters (i.e., MIB variables), whereas aggregated QoS parameters are an aggregation of several basic QoS parameters.

Calculation metric. The calculation metric specifies how QoD parameters are combined, respectively aggregated, together to obtain the requested QoS parameters. Calculation metrics are used to aggregate (i) QoD parameters $\rightarrow$ basic QoS parameters and (ii) basic QoS parameters $\rightarrow$ aggregated QoS parameters. A necessity hereby is the specification of a *calculation language* in order to describe the calculation metrics on the service layer appropriately, as proposed in 2.

Identification of QoD parameters. QoD parameters are relevant MIB variables or other data (e.g., log files) which can be gathered by device-oriented management tools. If following a top-down approach, the identification of the relevant QoD parameters, in case they exist, is certainly a challenging issue and depends on the application scenario.

Service Dependencies. As known services depend on sub-services in terms of a service hierarchy. The goal of this discussion is to address the description of the *depends on* relation between services and sub-services. It should be noted that we refer to dependencies which are visible from the outside (i.e., so-called inter-dependencies), and not to intra-dependencies within a service (e.g., dependencies between software components of a service). Service dependencies can be presented in terms of a directed acyclic layered graph. We refer to this graph as a service dependency graph. The nodes of the graph represent the services and sub-services whereas the directed edges represent the *depends on* relation between services and sub-services. It should be stressed that the dependency relation does not only exist between services on the higher layer and sub-services on the lower layer, but also between services on the same layer. For example, the proxy and the database services depend on the DNS (name service). Weights or other attributes may be assigned to edges of the service dependency graph if the graph is used for certain purposes such as root cause analysis, configuration issues, calculation of QoS parameters or service provisioning. For our purposes it is enough to model the dependency relation as a directed relation between a service and its sub-services and having the ability to assign attributes to the edges of the service dependency graph.

Service Cost. The service template model needs to address, additionally to the service quality, also the cost of the provided services. The service cost is measured with cost parameters, and is in relation with the provided service functionality and the quality. Similarly as with QoS parameters, we distinguish between (i) the cost for a whole service and (ii) the cost of separate service functional building blocks. Furthermore, it is necessary to distinguish between *one-time* cost for the service provisioning and the *running* costs. In order to measure the running costs, it is necessary to identify (i) accountable units of a service and the (ii) cost for a unit. The following tasks need to be performed: (i) the identification of service-related accountable units, (ii) specification of the measurement methodology and reference point, (iii) specification of tariff models to determine the service cost. The tariff model is, however, not part of the service model but part of the SLA.

The summarized view of the service template model is shown in Fig. 5.

4. Provider-Centric Service Template Model

The next step in the methodology of service modeling is the development of the **provider-centric service template model**. The provider-centric STM addresses the

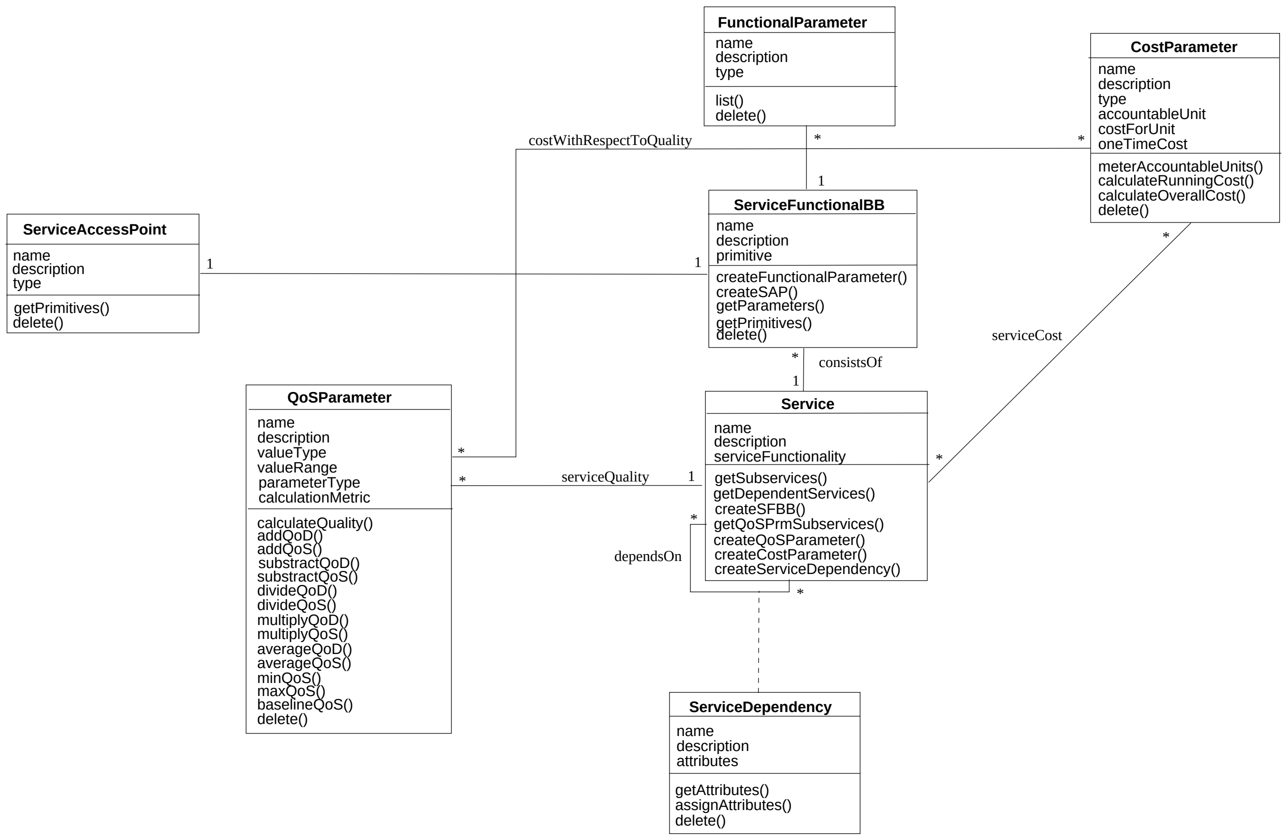

Figure 5. Design of the service template model

provider-centric issues which refer primarily to service provisioning and operation. The goal of this discussion is to identify the elements of the provider-centric part of a service.

Based on the provider's primarily objectives of service provision and operation, the following elements need to be addressed additionally in the provider-centric service template model: (i) specification of the steps for service provisioning, operation, change and withdrawal of a service, (ii) specification of the quality of the provided services, (iii) specification of policies to define the operation of services,(iv) specification of a *Customer Service Management (CSM)* to give customers a transparent view of the quality of the provided services. These elements are analyzed in more details in the following discussion.

Steps. Each phase of the service life cycle consists of several steps in order to realize the goal of the phase. For example, planning steps are necessary to plan the provision of a service. This includes mainly the identification of the resources to provide the service and the development of an operational concept for this service. However, planning steps are not part of the provider-centric service template model.

Provisioning steps are necessary for the configuration of resources involved in the service provision. A step or an action is thus associated with a resource, and it can be a simple execution of a script (e.g., `traceroute www.lrz.de`) or a complex process (e.g., *configure_database(database_server)). Operational steps* refer to steps necessary for the configuration of device-oriented management tools to monitor and control resources which are involved in the service provision. The goal is to gain a *service view* based on the device-oriented information as provided by management tools. One of the challenges hereby is the identification of the involved resources and the appropriate configuration of the device-oriented management tools. Another issue is the dynamic changing environment. For example, it is necessary to cope with services that are "moved" around the infrastructure, with changes in the functionality of management tools or new services that are introduced. Current management tools almost give no support to cope with the mentioned change dynamics. There is a lot of manual work associated with the tracking of changes. *Withdrawal steps* refer to the reconfiguration of resources involved in service provisioning as well as the reconfiguration of management tools involved in service monitoring and control.

Problems associated with the specification of steps are as follows: (i) it is necessary to specify the steps with an adequate granularity, and (ii) to assure the up-to-dateness of the steps. To cope with the high change dynamics, an alternative approach to maintain the steps is as follows: firstly, to specify the steps in a generic, parameterized way, and secondly, to improve the granularity and up-to-dateness of the steps with real experience during the application of the steps. Obviously, an ideal solution would be to have steps defined as executable scripts which could be executed automatically by existing tools. Unfortunately, such granularity is difficult to achieve and also to maintain due to the enormous effort. Such a detailed specification has already proved their weakness in the fault management area.

Provider-centric QoS parameters. The value range of service-centric QoS parameters is changed in the provider-centric service template model with respect to provider specifics.

SLAs. Another element of the provider-centric service template model which needs to be addressed are Service Level Agreements (SLAs), respectively the technical part of an SLA. A provider needs to think of SLAs that he can offer to customers.

Policies. A provider has certain provider-centric constraints that have an influence on the operation of services. Operational rules of service operation are specified as *policies.* Examples of policies are to make a service available 24 hours a day, 7 days a week or to deny everything what is not explicitely allowed. An adequate description of policies requires the development of an appropriate specification language, as proposed for example by Sloman et al. in 1. Besides, topics such as policy conflict resolution (10) need to be addressed as well. Our focus lies solely on the application of existing work done in policies to our problem area.

Customer Service Management (CSM). A requirement from the customer's side is to have a customer-centric and transparent view on the provided quality of the subscribed services. A provider needs to offer a set of reports about service quality that a customer can configure. Beside accessing reports about the provided service quality, customers want to interact with the provider for example to report problems, order new services, select new service qualities and service costs, track the resolution of problems. In other words, customers want to have access to a CSM. The objective of a provider is to develop a CSM application with the functionality to cover all phases of a service life cycle where a customer is involved in.

Customer Service Management Access Point (CSMAP). A CSMAP is the interface between a customer and a provider over which a customer can access the CSM application and the requested information. From the conceptual point of view, the meaning of a CSMAP is similar to a SAP.

Beside the description of the provider-centric service template, another important provider issue needs to be mentioned as well. A provider needs in some cases a more "global" view of the provided services to customers due to operational issues (e.g., availability of the whole IP backbone versus the availability of the individual router interfaces). We have referred to this view as the "all customers" view, and to the services as management services. Such a global view is important to monitor or achieve a good utilization of resources or to act appropriately in case of failures. Again, assume that a provider offers connectivity services to his customers, and that a SAP to a customer is a router interface. From the customer's perspective, only the availability of his SAP (e.g., the router interface) as well as the provided quality of service at the SAP is of importance. It is irrelevant for him whether other router interfaces or even routers are unavailable so far his quality of service is not affected. However, from the provider's perspective, the availability of the whole IP backbone is of relevance as well to act appropriately in the case of performance degradations or failures. Therefore, an example of such a management service is *IP backbone* and its availability.

As already emphasized, management services are treated in the same way as application-oriented services with respect to monitoring QoS parameters and costs. The only difference is that internal services have, in general, no penalties, in the case of violating the thresholds, associated with them (in most cases), and that in general the reporting and escalation mechanisms may be different.

A service provider uses the provider-centric service template model, or precisely the provider-centric service template instances, to build a **library of services** (e.g., Web, Mail, UHD, backup, remote printing) that he provides to his customers with the specified provisioning and operational steps. Several services can be combined together to *service packages* (e.g., Internet access including name service, access service, connectivity service etc.). A customer may now select between these packages

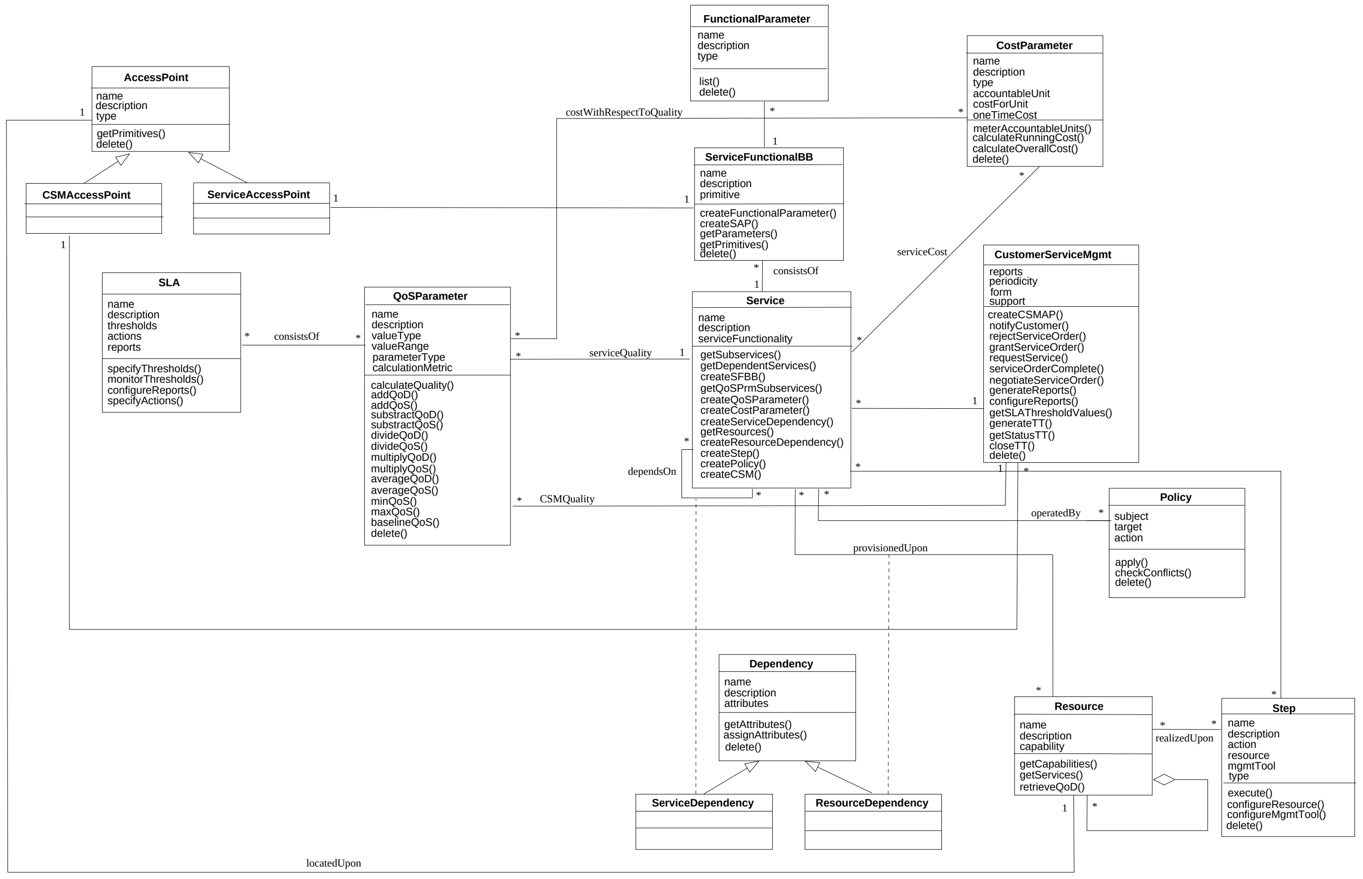

Figure 6. Design of the provider-centric service template model

or customize certain quality parameters (e.g., select the availability of a service greater than 99,98%).

The summarized view of the provider-centric service template model is shown in Fig. 6.

5. Customer-Centric Service Template Model

The last step of the methodology is the development of the **customer-centric service template model** to address the customer-centric issues. The customer-centric service template model is obtained by refining the provider-centric service template model and by adding attributes that represent customer-centric characteristics. This means that we add a class `Customer` to the provider-centric service template model as visualized in Fig. 6.

Customer-centric QoS parameters. They are a refinement of the provider-centric QoS parameters. A customer can select specific values from value ranges of QoS parameters which are offered by a provider. He may request also other values. In such a case, the negotiation about these values is started. The selected value is in relation with the cost. Another issue is that customer-centric quality parameters should express the quality of a service in a customer-centric way. Current practice is that QoS parameters are expressed as provider-centric parameters, such as packet or cell loss, reachability etc. The reason is obvious. Such QoS parameters are device-oriented and can be measured with existing device-oriented management tools.

Customer type. The customer type is used to distinguish between an intra- or inter-organizational provision of a service. This means to distinguish whether a service is provided to an external or to an internal (within an organization) customer. This is necessary because different processes (e.g., escalations, billing) may be used for an internal or external service provision.

Customer-centric CSM. A customer-centric CSM represents those reports about the provided quality of service that a customer has individually selected or configured from the available reports of the provider. A customer has the possibility to configure (i) what reports to access (e.g., specific QoS parameters, service levels), (ii) the periodicity of generating reports (monthly, daily, weekly etc.), (iii) the way, respectively the form, of providing reports (e.g., via email, Web, fax, paper). Beside the configuration of such reports, a customer may order new services, report problems with respect to his SLA, track resolution of problems etc. The customer-centric CSM can be considered as the customer's individual management interface to the provider.

6. Example

A simplified example of a problem management service should clarify the applicability of the proposed models. The class diagram in Fig. 6 extended with the class `Customer` should be consulted for that.

Service: Problem Management (PM) service with the functionality to support the resolution of customer-reported problems.

Dependency: The PM service depends on sub-services such as DNS, IP, database sub-service, file sub-service;

Service Functional Building Blocks: Examples are (i) documentation of problem resolution in trouble tickets, and (ii) workflow support. Examples of primitives are *submit(Problem), accessStatusProblem(TT-Number).*

Functional parameters: Examples are `Problem` which is a description of the reported problems or `TT-Number` to identify the trouble ticket which documents the reported problem.

QoSParameter: Examples are *first response time* and *average resolution time.* Value type for both parameters is real whereas the value range for both parameters is defined in the SLAs. The *first response time* is an example of a basic QoS parameter and the parameter *average resolution time* is an example of an aggregated QoS parameter.

SLA: An SLA consists of various QoS parameters for which thresholds, actions and reports are defined. An example of a threshold for the QoS parameter *first response time* is less than 4 hours in the case the reported problem is a critical one.

CostParameter: Accountable units for the PM service could be the number of problems (documented in trouble tickets) that can be reported by a customer in a certain time period. In addition a `costForUnit` could be associated to these reported problems.

Step: An example of a provisioning step is *configure_database(database_server). configure_NetworkMgmtPlatform(database_server);* is an example of an operational step. Withdrawal steps are the same steps as in the case of provisioning and operation, however, to withdraw the PM service.

CSM: CSM and the associated interface CSMAccessPoint define the attributes and operations that a customer can initiative over the CSM. Examples of operations are:
getProblemStatus(TT-Number);
getServiceQualityReport(average_resolution_time);
getServiceCostReport(month).

Customer: The type of a customer can be either internal or external.

Resource: Example of a resource is the database server as well as other network devices, providing IP sub-service, or end systems, providing DNS.

7. Assessment and Conclusions

The relevance of the developed service models can be summarized as follows: (i) definition of a **modeling approach** $\rightarrow$ object-oriented approach; (ii) definition of a **syntax for the description** of service-related management information $\rightarrow$ UML; (iii) specification of **service-related management information** and **service information** (e.g., what is a service, what is a SAP) in a customer-provider environment; (iv) specification of **relations** between the identified managed objects of IT service management (e.g., service and resource dependencies, cost and quality associations). An important contribution of the service models in addition is that they define elements which should be included in an SLA.

The proposed service model for IT service management represents the first attempt to provide a generic customer- and quality-oriented model for services which can be used as a common information basis for various applications. The proposed model defines commonly needed service-related terms, concepts and structuring rules in a general and unambiguous way. According to the methodology, a *service template model* is proposed to model service-centric issues of a service. A *provider-centric service template model* is further introduced to address provider-centric requirements mainly with respect to service provisioning and management. Lastly, a *customer-*

centric service template model is proposed to cover customer-centric aspects of service usage.

Areas for future work include: (i) further refinement of the specifications of service models (e.g., adding arguments to operations, specifying diagrams on the implementation level); (ii) using the proposed approach to specify a **library of services**. Such a library would include a set of *service templates* describing various services. Providers could access such library, select service templates and customize them into provider-centric service templates, and into customer-centric service templates respectively; (iii) building appropriate service management development tools (e.g., editors, consistency checkers) for the description of service templates, provider-centric and customer-centric service templates. Providers could use these tools to describe their specific services.

Acknowledgments

The author wishes 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 is a group of researchers of the Munich Universities and the Leibniz Supercomputing Center of the Bavarian Academy of Sciences. It is directed by Prof. Dr. Heinz-Gerd Hegering.

References

[1] N. Damianou, E. Dulay, N. Lupu, and M. Sloman. The ponder policy specification language" *in: proceedings of the workshop on policies for distributed systems and networks, bristol, springer-verlag, lncs 1995*. pages 18–39, January 2001.

[2] G. Dreo Rodosek. *A Framework for IT Service Management*. habilitation thesis, University of Munich, June 2002.

[3] P. Ferguson and G. Huston. *Quality of Service - Delivering QoS on the Internet and in Cooperate Networks*. John Wiley and Sons, ISBN 0-471-24358-2, 1998.

[4] M. Garschhammer, R. Hauck, H.-G. Hegering, B. Kempter, I. Radisic, H. Rølle, H. Schmidt, M. Langer, and M. Nerb. Towards generic service management concepts — a service model based approach. In Pavlou et al. [11], pages 719–732.

[5] R. Gopal. Unifying Network Configuration and Service Assurance with a Service Modeling Language. In Stadler and Ulema [12], pages 711–725.

[6] Open Distributed Processing – Reference Model – Part 2: Descriptive Model. IS 10746-2, International Organization for Standardization and International Electrotechnical Committee, 1993.

[7] M. Langer, S. Loidl, and M. Nerb. Customer service management: A more transparent view to your subscribed services. pages 195–206, October 1998.

[8] A. Lazar, R. Saracco, and R. Stadler, editors. *Proceedings of the Fifth IFIP/IEEE International Symposium on Integrated Network Management V (IM'97)*, San Diego, USA, May 1997. Chapman & Hall.

[9] L. Lewis. *Service Level Management for Enterprise Networks*. Artech House Inc., ISBN 1-58053-016-8, 1999.

[10] E. Lupu and M. Sloman. Conflict Analysis for Management Policies. In Lazar et al. [8], pages 430–443.

[11] G. Pavlou, N. Anerousis, and A. Liotta, editors. *Proceedings of the Seventh IFIP/IEEE Integrated Network Management VII (IM'01)*, Seatle, WA, May 2001. IEEE Publishing.

[12] R. Stadler and M. Ulema, editors. *NOMS 2002 IEEE/IFIP Network Operations and Management Symposium — Management Solutions for the New Communications World*, Florence, Italy, April 2002. IEEE/IFIP.

[13] Service Architecture. Tina baseline, TINA Consortium, June 1997.