Quality Aspects in IT Service Management

Gabi Dreo Rodosek

Munich Network Management Team
Leibniz Supercomputing Center
Barer Str. 21, 80333 Munich, Germany
dreo@lrz.de

Abstract. 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. Quality management is an important research topic of IT service management. The paper addresses the problem of the specification of quality of service (QoS) parameters and their mapping to the quality of device (QoD) parameters. It proposes a language for the specification of QoS parameters.

1 Introduction

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. Instead of *resources* such as network devices, end systems and applications, it is necessary to think in terms of *services* and *service quality* ([Dreo 02]).

The concept of a service is a recent advancement in the understanding of networking technology. But what is a service? Probably, there exist as many definitions of a service as there are approaches to address particular problems of service management. Lewis for example describes a service as an abstraction over the enterprise network, and as being composed of components ([Lewi 99]). In [FeHu 98], it is recognized that a service may have several meanings, depending on how an organization or business is structured. The term service is used in general to describe something that is offered to the users of any (networked) system. The reference model of Open Distributed Processing (ODP) defines the term service as a function provided by an object at a computational interface ([ISO 10746-2]). In Intelligent Networks (IN) ([ITU Q.1202]) services and service features are with units of service functionality which are referred to as service independent building blocks (SIBs).

Despite of such an amount of service definitions, a common understanding of a service that provides a unified approach to support the concepts of service

M. Feridun et al. (Eds.): DSOM 2002, LNCS 2506, pp. 82–93, 2002.

management is lacking. A first step towards a *generic service model* has been proposed by the Service Management Task Force (SMTF)[1] in [GHHK 01] and [GHHK 02].

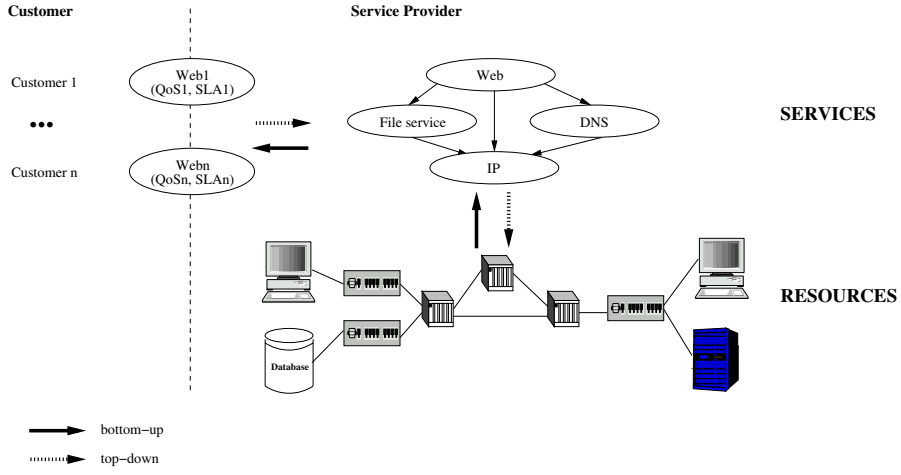

Fig. 1. Top-down versus bottom-up construction of QoS parameters

Quality management or service level management (SLM) is an important research topic of IT service management which is addressing the mapping of services to the resource layer with respect to quality parameters. The goal of the paper is to analyze the construction of QoS parameters and specify a language for the description of the calculation metric.

The construction of QoS parameters can be approached in two ways: (i) from a top-down or (ii) a bottom-up perspective. The current practice among service providers regarding the definition of QoS parameters is *bottom-up*. Service providers typically start with network-centric parameters and then try to mould them into service-centric parameters. This procedure is represented with the solid arrows in Fig. 1. As a consequence current SLAs in the domain of networking are mainly service provider-centric (e.g. usually in terms of reachability, packet loss, error rates, jitter, delay) even though the services they support include E-commerce, Web or Email services. Service providers refer to Internet connectivity services (layer 3 and lower) instead of higher-level customer-centric services. However, from the business viewpoint it is important to know that emails are correctly delivered to the recipient within a certain time interval, or that hits on a business web site are reliable and fast enough to hold the customer's attention. The reason for such an approach is that most current man-

[1] SMTF is a group of researchers within the Munich Network Management team

agement tools are device-oriented and that a *service representation is missing*[2]. Such an approach does not measure the quality of the provided services from the customer's perspective.

To address the problem area from a customer-centric perspective, we propose an approach that is opposite to the current trend among service providers. It is purposely *top-down*. The objective is to start with the definition of *customer-centric QoS parameters* and try to map them on *basic QoS parameters* and afterwards on *QoD parameters*, as visualized in Fig. 1 with dotted arrows. We define *QoD* parameters as being specific MIB variables, *basic QoS* parameters as QoS parameters of basic services (i.e. services that do not depend on sub-services), and *aggregated QoS* parameters as QoS parameters of aggregated services (i.e. services that depend on sub-services).

Three aspects need to be addressed: (i) specification of customer-centric QoS parameters, (ii) gathering of necessary service-related information from resources in order to gauge SLAs over time, and (iii) identifying means by which to configure and control resources such as network devices, end systems or applications with respect to selected services and SLAs. The benefit of such an approach is in obtaining *customer-centric and service-oriented QoS parameters*. Reasons that this approach was not feasible so far were: (i) a missing service description, (ii) a missing description of service dependencies, as well as (iii) a missing description of the distributed realization of services upon resources. Besides, a problem which needs to be faced with the top-down approach are missing QoD parameters, respectively MIB variables, which are required by the top-down definition of customer-centric QoS parameters. A summary of the discussion of both approaches is visualized in Fig. 2.

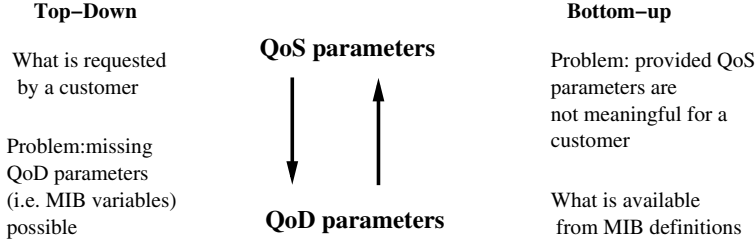

Fig. 2. Summary of the top-down versus bottom-up approach

A key concept of both approaches is to specify the formula of aggregating QoD parameters to basic QoS parameters and furthermore to aggregate QoS parameters in terms of a *calculation metric*. The top-down approach of describing QoS parameters and the associated calculation metrics requires the ability to specify the calculation metrics in terms of a high-level *calculation metric language*. What is lacking is such a common language that provides a unified

[2] Rudimentary approaches to provide a service view are proposed by CA and Tivoli. However, these can be considered only as first steps towards service-oriented management tools.

approach of describing calculation metrics of QoS parameters. An analysis of the requirements for such a language provides the basis for the specification of the language.

The paper proceeds as follows: Section 2 analyzes the requirements for a calculation metric language by pointing out the need to deal with (i) QoD parameters, (ii) basic QoS parameters and (iii) aggregated QoS parameters. The calculation language QUAL is described in section 3. Finally, Section 4 concludes the paper.

2 Requirements for the Calculation Metric Language

Requirements for the calculation metric language are derived from the scenario as depicted in Fig. 1. In the previous section we introduced the terms (i) aggregated QoS, (ii) basic QoS and (iii) QoD parameters. Relations between the parameters are defined as follows: $QoS_A = f(QoS_B)$, and $QoS_B = f(QoD)$. As a result, two mappings need to be addressed:

- Aggregated QoS parameters to basic QoS parameters (i.e. $QoS_A \rightarrow QoS_B$),
- Basic QoS parameters to QoD parameters (i.e. $QoS_B \rightarrow$ QoD.)

Another issue which should be mentioned is the "interpretation" of the obtained values. This is part of an SLA, and is performed with respect to the agreed thresholds according to the following specification

if (`QoSParameter[value]` > `threshold`) **then action** or
if (`QoSParameter[value]` $\in$ critical_value_set) **then action**

The further discussion is focused on the calculation of values of QoS parameters and the identification of the requirements for a calculation metric language to support the (i) mapping of aggregated QoS parameters $\rightarrow$ basic QoS parameters, and (ii) the mapping of basic QoS parameters $\rightarrow$ QoD parameters.

2.1 Aggregated QoS Parameters → Basic QoS Parameters

Requirements for the calculation metric language for the mapping of aggregated QoS parameters to basic QoS parameters result from the service hierarchy and the dependency relation between services and their sub-services. Attributes can be assigned to the dependency relation for various application scenarios. The following discussion identifies those attributes which are relevant for quality management. Attributes of service dependencies are defined by the dependency relation itself which means whether a service depends on a (sub)-service or not. This can be expressed with an *AND* operator. For example, if a service S depends on a sub-service SS_1 AND sub-service SS_2, this is expressed as follows

$$S \text{ depends\_on } (SS_1 \text{ AND } SS_2)$$

Accordingly, the aggregation of QoS parameters of these services is as follows

$$QoS_A(S) = f(\ QoS_B(SS_1)\ \text{AND}\ QoS_B(SS_2))$$

The semantics of the AND operator needs to be specified for each QoS parameter separately. For example, for the QoS parameter *availability* the AND operator is defined as follows

$$QoS_1\ \text{AND}\ QoS_2 := QoS_1 * QoS_2$$

In other words the Web service is available if for example the file, name and the connectivity sub-service are available as well. The AND operator which is used for the aggregation of QoS parameters is expressed with mathematical and statistical operators, and is also used for the aggregation of QoD parameters.

The semantics of the OR operator for the availability examples is as follows. We assume that we have a primary connection between A and B as well as a backup connection. In the case the primary connection is not available, traffic is transferred over the backup one. The semantics of the OR operator in this case would be as follows:

$$\begin{aligned} QoS_1\ \text{OR}\ QoS_2 &:= QoS_1, \text{ traffic transferred over the primary connection;} \\ &:= QoS_2, \text{ traffic transferred over the backup connection.} \end{aligned}$$

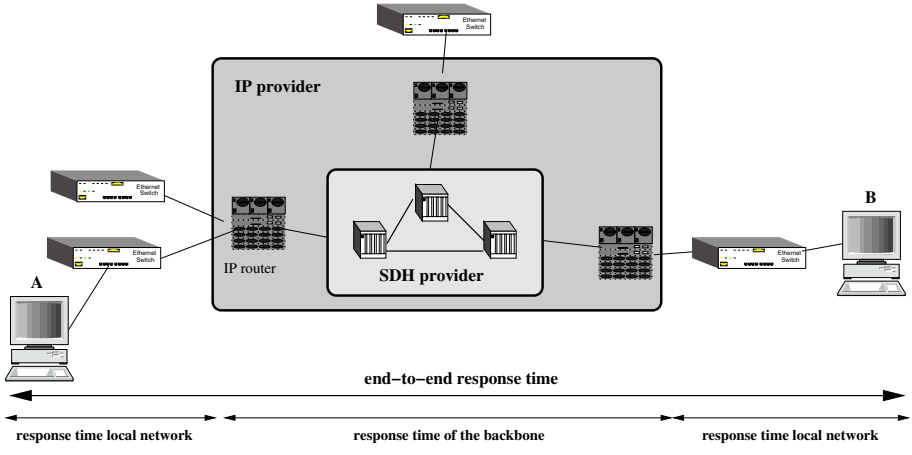

Fig. 3. End-to-end response time

Another representative QoS parameter is *response time*. Let us analyze the (simplified) scenario as depicted in Fig. 3. Let us assume that a customer subscribes to a virtual private network (VPN) service to connect his locations in various countries. One of the customer-centric QoS parameters he agrees on with his IP service provider is the *end-to-end response time*. End-to-end means from an end system in location A to an end system in location B. Accordingly, the IP service provider agrees on a response time with his Synchronous Digital Hierarchy (SDH) provider (an underlying service in terms of a service hierarchy) to be able to assure the agreed response time with his customer.

As depicted in Fig. 3, the end-to-end response time is a sum of the response time within the local network where A is located, the response time in the backbone and the response time of the local network where B is located. In order to calculate the response time, we recognize that the service S VPN (with respect to the calculation of the response time) depends on the sub-service SS_1 "local connectivity" AND on the SS_2 sub-service "backbone connectivity" AND on the sub-service SS_3 "local connectivity" expressed as

$$S \text{ depends on } (SS_1 \text{ AND } SS_2 \text{ AND } SS_3)$$

Accordingly, the aggregated QoS parameter QoS_A "end-to-end response time" is a function of the basic QoS parameters $QoS_B(SS_1)$, $QoS_B(SS_2)$ and $QoS_B(SS_3)$.

$$QoS_A = f(QoS_B(SS_1), QoS_B(SS_2), QoS_B(SS_3))$$

$$QoS_A = f(QoS_B(SS_1) \text{ AND } QoS_B(SS_2) \text{ AND } QoS_B(SS_3))$$

The semantics of the AND operator for the calculation of the response time is defined as follows:

$$QoS_1 \text{ AND } QoS_2 := QoS_1 + QoS_2$$

As a result of the previous discussion follows that the calculation language needs to express *service-specific operations*.

2.2 Time Aspect

Another important aspect which has not been addressed yet is the *time aspect*. QoS parameters are observed within a so-called *observation time interval*, denoted with $[t_1, t_2]$. The retrieval of MIB values (i.e. QoD parameters) is done at certain time measurement points t'.

The previous definitions of calculating the aggregated and basic QoS parameters need to be extended as follows. For example, the aggregated QoS parameter availability QoS_A for service S in a time moment t' is the availability of the basic QoS parameter QoS_B for sub-service SS_1 in the time moment t' AND the availability of the basic QoS parameter QoS_B for sub-service SS_2 in the time moment t'. Thus,

$$QoS_A(S, t') = f(\ QoS_B(SS_1, t') \text{ AND } QoS_B(SS_2, t'))$$

The values of aggregated QoS parameters at the measurement point t' are aggregated over a certain time interval as follows

$$QoS_A(S, [t_1, t_n]) = \frac{\sum_{i=1}^{n} QoS_A(S, t'_i)}{n}$$

Our purpose was to describe the semantics of the AND and OR operators on the example of availability. For more details about the calculation of the availability of distributed applications, and the problems which need to be faced, it should be referred to ([DrKa 97]).

2.3 Basic QoS Parameters → QoD Parameters

An example should introduce the discussion of this mapping. Let us assume to calculate the basic QoS parameter *interface_traffic* for an IP service. The goal is to calculate the amount of data which has been transferred over a router interface. To determine the traffic of an interface of a router, it is necessary to add the following two expressions `traffic[ifIndex] = ifInOctets[ifIndex] + ifOutOctets[ifIndex]`. In other words, the traffic for an interface `ifIndex` is determined as the sum of the In and Out octets transferred over that interface. Traffic is an example for a basic QoS parameter whereas `ifInOctets` and `ifOutOctets` are examples of QoD parameters. Another example is the total traffic which was correctly sent or received over an interface of a router, and is expressed as follows: `TotalTraffic[ifIndex] - TrafficErrors[ifIndex]`.

Values of QoD and QoS parameters are often averaged over calculation time intervals. A common term for this is baselining values. For example, to baseline the `traffic`, as specified before, over a `day`, this would be expressed with `baseline(traffic,day)`. In addition to baselining values over time, it is necessary to calculate also *average* values of parameters such as `AverageTraffic` as well as *max, min* values.

As a result of the previous discussion follows that the calculation language needs to express *mathematical operations*.

2.4 Requirements for the Calculation Language

The previous discussion identified the following requirements:

- It is necessary to support a top-down description of aggregated QoS parameters, and their aggregation of basic QoS and QoD parameters.
- Structuring techniques to facilitate the calculation of QoS parameters which are related to several resources need to be supported.
- It must be possible to specify time intervals and resources from which QoD parameters are retrieved.
- It must be possible to use various operators such as mathematical or statistical ones.
- Selection operations to operate on objects (services, service and resource dependencies) are necessary to be supported.
- Extensibility is needed to cater for new QoS parameters, new operators, new functions etc. that may arise in the future with new services.
- The language must be comprehensible, extensible and easy to use.

3 QUAL: The Calculation Language

The goal of QUAL is to address the calculation of QoS parameters, and all service-related operations, as visualized in Fig. 4. Therefore, we will not address the calculation of QoD parameters, as already addressed by tools such as InfoVista ([InfoVista]).

Another view of the research issue is given in Fig. 5. The service graph on the service and resource layer gives the basis for deriving a QoS, respectively QoD, parameter graph. Thus, an aggregated QoS parameter QoS_A for a *service* within an observation time interval *time* is a function of the basic QoS parameters QoS_B of those sub-services that a service depends on within the same time interval. A prerequisite for this is the knowledge of what sub-services a service depends on.

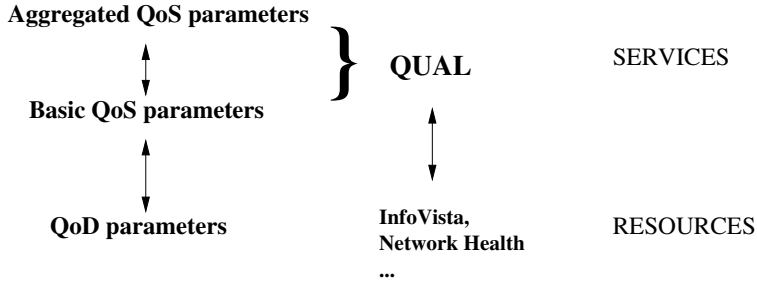

Fig. 4. QUAL: calculation of QoS parameters

Accordingly, a basic QoS parameter QoS_B for a service *service* within the time interval `time` is a function of QoD parameters within the same time interval from resources that a sub-service depends on. A prerequisite for this is the knowledge of the involved resources in the provision of the sub-services.

$$QoS_A(\text{service,time}) = f(QoS_{Bi}(\text{sub-service,time})), \qquad \text{i=1, ..., n}$$
$$QoS_{Bi}(\text{service,time}) = f(QoD_j(\text{time,resource})), \qquad \text{j=1, ..., m}$$

The following discussion identifies the syntax and semantics of the language.

3.1 Service-Specific Operations

The language includes service-specific operations which are necessary on the service layer. We assume that the selection and retrieval operations are performed in discrete time measurement points. The operations defined are as follows:

- **selectSubservices(service).** This operation returns as a result all sub-services of a service `service`.
- **selectService(sub-service).** This operation returns as a result all services where the sub-service `sub-service` is involved in the service provision.
- **selectQoSofAllSubservices(QoS,service).** This operation selects a QoS parameter denoted as `QoS` of all sub-services of service `service`. Example: selectQoS(availability,Web). This operation returns as a result the QoS parameter `availability` of all sub-services of the service `Web`.
- **calculateQoS(QoS,service,time).** This operation calculates the value of a QoS parameter `QoS` of all sub-services of a service `service` within a time interval `time`. Example: calculateQoS(availability,IP,$[t_1, t_2]$). This operation returns as output the availability for the IP service within the time interval

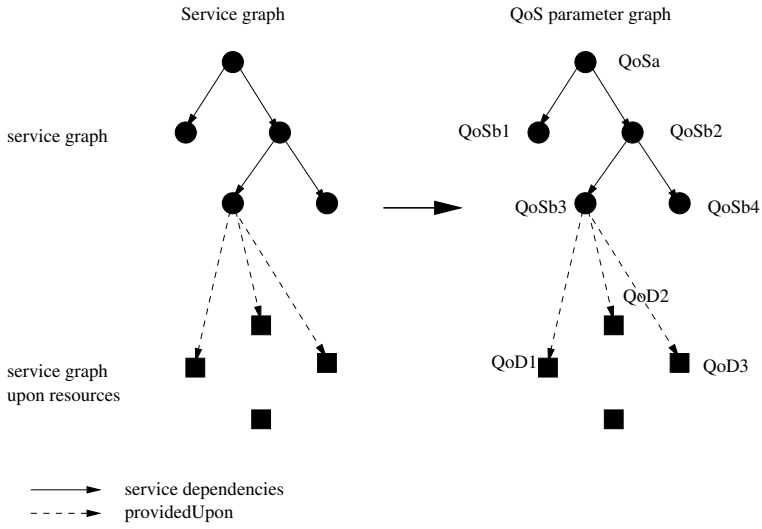

Fig. 5. Service vs. QoS parameter graph

$[t_1, t_2]$. A prerequisite for this operation is to retrieve QoD parameters of resources that are involved in the provision of the IP service. More precisely, this requires the operation **retrieveQoD** as described next.

– **retrieveQoD(QoD,resource).** This operation retrieves the QoD parameter `QoD` for a resource `resource`. Example: retrieveQoD(ifInOctets,if1csrwan).

3.2 Calculation Operations

We mention here the required mathematical and statistical operations that are necessary for the calculation of QoS parameters. We assume that the calculations are performed always within a certain time interval although not explicitely expressed in the description of the operations. The relevant operations are:

– **addQoD — addQoS**. Calculates a sum of the specified QoD, respectively QoS, parameters, and is defined as follows

$$\text{addQoS}(QoS_1, QoS_2, \ldots, QoS_n) := \sum_{i=1}^{n} QoS_i$$
$$\text{addQoD}(QoD_1, QoD_2, \ldots, QoD_n) := \sum_{i=1}^{n} QoD_i$$

An example of the `addQoD` operation is to calculate the interface traffic `traffic[ifIndex] = ifInOctets[ifIndex] + ifOutOctets[ifIndex]`. The definition of other operations is similar.

– **subtractQoD — subtractQoS.** Substract QoD and QoS parameters.
– **divideQoD — divideQoS.** Divide QoD and QoS parameters.
– **multiplyQoD — multiplyQoS.** Multiply QoD and QoS parameters.

– **averageQoD — averageQoS.** Calculate an average of values of QoD, respectively QoS, parameters within a certain observation time interval $[t_1, t_2]$ including n measurement points. This operation is defined as follows:

$$\text{averageQoS}(QoS_i,[t_1,t_2]) := \frac{\sum_{i=1}^{n} QoS(t_i')}{n}$$
$$\text{averageQoD}(QoD_i,[t_1,t_2]) := \frac{\sum_{i=1}^{n} QoD(t_i')}{n}$$

– **minQoS — maxQoS.** Specifies the maximal, respectively minimal, QoS parameter (e.g., max response time) within a certain observation time interval.
– **baselineQoS(QoS,time).** This operation enables to average the values of a QoS parameter `QoS` for the observation time interval `time`. Example: baselineQoS(traffic,day).

3.3 Application of QUAL

The objective of the top-down construction of QoS parameters is to approach the description of the calculation of customer-centric QoS parameters. The following example should demonstrate the application of QUAL for the description of the availability of a Web service. Let us assume to specify the *availability* of the Web service (in a simplified way) in a time moment t' as follows:

calculateQoS(availability,Web,t') =
multiplyQoS(calculateQoS(availability,DNS,t'), calculateQoS(availability,IP,t'))

A Web service is available in the time moment t' if the DNS and the IP service are available in the time moment t' as well.

Another example is the availability of the IP service. which would be described in a simplified way as follows:

calculateQoS(availability,IP,t') =
addQoD(retrieveQoD(ifOperStatus,if[Index]router$_1$, ...
retrieveQoD(ifOperStatus,if[Index]router$_n$))

This means that the availability of the IP service is determined with the availability of the interfaces of the backbone routers. The procedure to calculate this availability is as follows: Firstly, it is necessary to recognize what interfaces of a router are in the administrative state. This information is obtained by retrieving the value of the MIB variable `ifAdminStatus.` Secondly, the operational status of those interfaces is retrieved that have been identified as being in the administrative state. This information is obtained by retrieving the values of the MIB variable `ifOperStatus.` At this stage, the operations can be directly mapped to operations of existing device-oriented SLM tools such as InfoVista.

3.4 Assessment

Obviously, a precondition for QUAL is the existence of a *common service model*, and hence access to service and QoS parameters graphs. The contributions of

QUAL can be summarized as follows: QUAL provides a convenient and novel approach for a high-level specification of calculation metrics of basic and aggregated QoS parameters. Currently, the specification of QoS parameters is limited to the operations of device-oriented SLM tools. Furthermore, the specification of QUAL imposes also requirements for the implementation of the service graphs and the resulting QoS parameter graphs. Areas of further work include the following issues: **Specification of further elements of the language.** The goal of QUAL is to enable the description of calculation metrics of QoS parameters, and thus, the most important aspect of the specification of the necessary operations has been described. What is missing are the specifications of further elements of the language itself (e.g. keywords). **Integration of QUAL with device-oriented SLM tools.** It is necessary to define appropriate mappings of operations defined in QUAL and existing device-oriented SLM tools. The vision is that a provider specifies the calculation of QoS parameters which is directly mapped to operations of device-oriented SLM tools. Obviously, this is far from being simple, as practical experiences with device-oriented SLM tools show, although it is essential for a seamless integration of service-oriented and device-oriented management. **Tool support for QUAL.** As each concept, QUAL needs to be tool supported. An extension of existing device-oriented SLM tools with the defined QUAL language elements and operations could be an alternative.

Some words about the measurement methodology should conclude the assessment discussion. The basic principle of the previous approach is to access QoD parameters, respectively retrieve values of MIB variables, and calculate afterwards the QoS parameters. In such a case, the measurement methodology is *passive* in terms of just retrieving values of MIB variables. Another way of testing, for example the availability of the Web service, is to *actively* call the Web service from the client and recognize whether the service is available or not. In such a case the measurement methodology is consider to be *active*. Regardless of an active or passive measurement methodology, the results can be accessed via MIB variables[3] and represented in terms of reports about service quality.

4 Conclusions

IT service management is a complex and large research topic. Quality management is a challenging and fundamental research topic which has been selected due to addressing the complex mapping between the service and resource layer with respect to quality parameters. Two challenges have to be approached: What is an appropriate approach to specify and obtain QoS parameters, and how to describe the calculation metrics of basic and aggregated QoS parameters? To answer the first question, we have proposed a top-down approach for the specification of QoS parameters. This means to start from customer-centric QoS parameters and afterwards specify how to map them to QoD parameters, which is completely opposite to the current practice.

[3] It should be noted that every format (e.g. log file) can be described in terms of MIB variables.

The analysis of the second question required the development of a specification for the description of calculation metrics of QoS parameters. We developed therefore a specification language QUAL. The goal of QUAL is thus to provide specification means for the description of the calculation metrics of QoS parameters. Furthermore, it enables the specification in such a granularity that a direct mapping to device-oriented SLM tools, dealing with QoD parameters, is possible.

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

[Dreo 02] G. Dreo Rodosek, *A Framework for IT Service Management*, habilitation thesis, University of Munich, June 2002.

[DrKa 97] G. Dreo Rodosek and Th. Kaiser, "Determining the Availability of Distributed Applications", In A. Lazar, R. Saracco and R. Stadler, editors, *Proceedings of the Fifth IFIP/IEEE International Symposium on Integrated Network Management V (IM'97)*, pages 207–218, San Diego, USA, May 1997, Chapman & Hall.

[FeHu 98] 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.

[GHHK 01] M. Garschhammer, R. Hauck, H.-G. Hegering, B. Kempter, I. Radisic, H. Rölle, H. Schmidt, M. Langer and M. Nerb, "Towards Generic Service Management Concepts — A Service Model Based Approach", In [IM 01], pages 719–732.

[GHHK 02] M. Garschhammer, R. Hauck, H.-G. Hegering, B. Kempter, I. Radisic, H. Rölle and H. Schmidt, "A Case-Driven Methodology for Applying the MNM Service Model", In [NOMS 2002], pages 697–710.

[IM 01] 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.

[InfoVista] http://www.infovista.com.

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

[ITU Q.1202] "Intelligent Network - Service Plane Architecture", Recommendation Q.1202, International Telecommunication Union, October 1992.

[Lewi 99] L. Lewis, *Service Level Management for Enterprise Networks*, Artech House Inc., ISBN 1-58053-016-8, 1999.

[NOMS 2002] 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.