Service Level Agreements based on Business Process Modeling

Holger Schmidt

Munich Network Management Team
University of Munich, Dept. of CS
Oettingenstr. 67, 80538 Munich, Germany
Email: schmidt@informatik.uni-muenchen.de
Phone: +49 89 2178 2165, Fax: +49 89 2178 2262

Abstract

In the evolving service market there is a good chance for companies to update their IT environment or cut their IT costs by outsourcing some parts of their IT processes. This implies meeting the challenge of service quality. For common services this can be handled by changing the service provider. But this is not valid for enterprises that want to outsource rather complex and individually composed services which are part of their business processes. In this situation it is important to implement a contract that allows to manage the service effectively. It must be possible to monitor the service and to manage any fault situation in a constructive and fast manner. For this reason customers want to declare the necessary quality of service in a service level agreement.

Usually service level agreements for complex services are specified by writing down a set of rules. These rules are based mostly on the experience of the provider. This prevents the service from being customer-centric because the contracts emphasize the service implementation instead of its usage. This inadequate view on the service during contract creation leads to a dissatisfactory situation because there often are disagreements on the exact interpretation of the contract. The reason for the difference of opinion is that the customer is forced to work with the implementation view that he often does not understand and in fact is not even interested in.

This paper presents a customer-oriented approach for specifying service level agreements. Our idea is to combine service level agreements with concepts of workflows used in business process modeling. The knowledge on design and management of workflows can be used to specify a service level agreement actively supporting the operation and usage of complex services. The customer-oriented view is ensured by the use of the business processes of the customer as a base for the contract. The combination of these two concepts allows both, specifying non-ambiguous contracts and constructive instructions for the management of services from the customer's point of view.

Keywords

Service Level Agreements, Service Contract, Workflow, Business Process, Outsourcing

1 Introduction

The increasing dependence of a companies success on IT infrastructures requires to face the challenges of service quality. The complexity of the needed IT services is growing. Therefore, the design and operation of services is often outsourced to independent internal departments or to external partners.

To specify the requested service customer and provider sign a service level agreement (SLA). Such SLAs define the functionality of the service and the required service level. The SLA is an important source of information for the provider's service management because the service provider often does not know much about the scenario the service is used in. Thus, the quality of the SLA is one important parameter for the success of outsourcing relationships.

Keeping the service quality on a reasonable level is the challenge of modern services. It is also a task of service level agreements to ensure that the customer has sufficient management facilities to monitor and control the

actual service quality. Therefore, management interactions need to be specified in addition to service level and functionality. Most SLAs focus on technical parameters of the service like bandwidth or availability. But the management interactions must also fulfill quality criteria. In this paper the term management means the service management crossing the domain boundary between customer and provider. Management of the service implementation done by the provider is considered as operation.

The specification of interactions is difficult if it is only based on a relatively informal collection of rules and statements. The temporal aspect of interaction processes cannot be represented adequately and it is hard to master the complexity with unstructured methods from which many SLAs for today's IT services result.

Each interaction can be seen as a process. Production processes or business processes are described by workflow concepts. Therefore, this paper proposes the usage of workflow concepts for designing and writing of high quality service level agreements for IT services. For demonstrating purposes a scenario is presented in section 2. Sections 3 defines the term service level agreement, derives requirements and identifies the elements of SLAs. The relevant concepts of workflow modeling are described in section 4. Section 5 presents our process model for use in service level agreements followed by the classification of processes. Section 6 shows a simplified example process to demonstrate the application of our approach. The last section draws a conclusion and introduces future work.

2 Scenario

To clarify the presentation of our approach and for demonstrating the potential complexity and investments of today's services a scenario is developed (see figure 1). The scenario originates from experiences in several projects with industrial partners. It shows the complexity and high investments a custom–designed IT service can require today.

A company selling its high priced customizable products over an international network of dealers wants to connect those with its own central IT infrastructure. This permits the dealers to customize and order the products online. Additionally, multimedia information material is available online for product presentation. Furthermore, up to date information on the stocking, production date, date of delivery, etc., is available for sales conversation of the dealers with their customers. Each product is individually traceable during the production and delivery processes. Similar facilities are available for after–sales contacts. The applications supporting this functionality are hosted by the company itself.

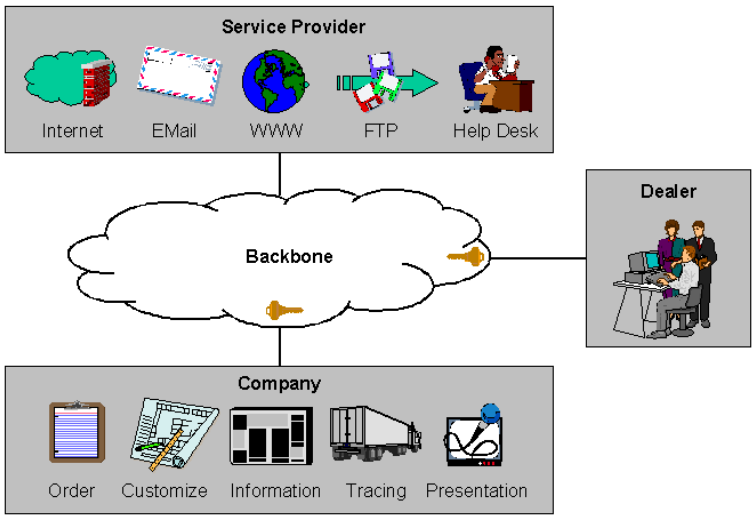

Figure 1: Scenario

The company has a SLA with a service provider for the functionality and necessary management described next. The dealers can also use as additional features Internet connectivity with special support for email, world wide web and file transfer. These services can be used for the communication of dealers with each other, too.

All communication between dealers and the company is encrypted for security reasons. A firewall secures the Internet gateway. The infrastructure is IP based which requires the operation of all supporting services like management of domain names and IP addresses including the necessary infrastructure. The dealer's connectivity to the provider's backbone is realized by leased lines or ISDN switched lines using components at dealers locations owned by the service provider. Finally the usually quite unexperienced IT users at the dealer location need a help desk supporting them for day to day connectivity problems. The service provider insists on a long–term SLA of several years because the provider wants to amortize the investments in design, hardware, software and staff.

3 Service Level Agreement

This section defines service level agreement and identifies requirements and its elements. A *service level agreement* is a contract between customer and provider specifying the service functionality and all management interactions of the customer–provider relationship including the required service level. For complex long–term services these management interactions are as important as the service functionality to achieve a successful outsourcing partnership during the total SLA lifetime.

3.1 Requirements

Common requirements like validity, completeness, consistency or unambiguity must be fulfilled by all types of contracts. But there are additional requirements for service level agreements that specify complex IT services in a long–term customer–provider relationship. A service level agreement must be:

customer–centric: The SLA must be specified in a terminology originating from the customer's usage scenario. The service implementation should be transparent to the customer. The service may not use the terms of the service implementation view usually preferred by the provider. But this must not lead to restricted expressiveness. All necessary details must be expressible while allowing the abstraction of well known facts.

This results in a service–oriented description which does not restrict the implementation of the service by the provider allowing to profit from technological progress. If a SLA is demanding a special kind of implementation a new SLA must be negotiated to make use of possible advantages. Instead it should focus on usage processes.

constructive: Management facilities for the customer are an essential element in a SLA. The management interactions should be used to actively prevent or at least constructively solve problems in cooperation. Operators as well as users should be guided to become active by a service level agreement if the situation requires intervention. The SLA must define the rights and duties of both, customer and provider. All other management interactions like ordering, accounting, maintenance, etc. need to be explicitly specified, too.

As for service functionality there also must be service levels for the management processes. The service level must be specified by concrete quality criteria which are measurable and realistic. The SLA must be understandable by both, user and operator. Therefore, a SLA requires concrete and workable instructions which are supportive for usage and management. This way the service level agreement is present during operation and usage not just in a legal proceeding.

Summarizing, to reach these targets a constructive SLA is needed which is customer–centric and service–oriented. Furthermore, it must be concrete, understandable and workable, but without loosing expressiveness. That means all contracting partners know their rights and their duties.

3.2 Elements

The rights and duties are specified by three groups of information shown in figure 2: legal, organizational and technical information. To be considered valid a SLA must comply with all formalities, e.g. names, addresses and signatures of the contracting parties are needed. This *legal information* is important for a contract, but it is regulated by law. Therefore, this paper focuses on organizational and technical aspects of the customer–provider relationship.

The service is described by *technical information* on functionality, capacity, quality and the interfaces for usage and management. Functionality describes what the service supports to do, not how it is done. Capacity and quality criteria specify the service level that must be fulfilled. The usage interface is the service access point. A management interface is needed for monitoring and controlling the service from the customer's domain.

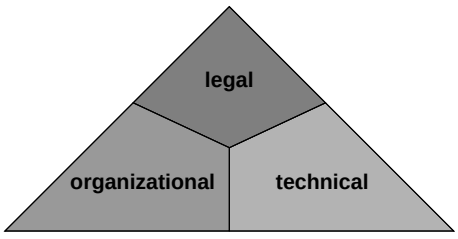

Figure 2: Elements of SLAs

The *organizational information* includes all interactions between customer and provider for the usage and management of the service.

Usage interactions are for example the login process for ordering a product. Interactions like adding a new dealer or problem handling are management interactions.

3.3 Related Work

Most service level agreements for complex services specify a set of rules and statements which are based particularly on experiences of provider and customer. This very informal procedure does not help in fulfilling the raised requirements. It is very difficult to write high quality SLAs, if there is no structure or methodology guiding the design process.

An approach for specifying SLAs in federated environments is presented in [**?**]. A service consists of several components. For each component quality metrics and interdependencies to other components can be specified. This straightforward approach allows to verify the service quality by evaluating relations of measured values provided by each component. The focus lies on service implementation. This complicates the use for complex services because customers are not interested in the numerous measured values of components implementing the service. The implementation view is useful for providers but too difficult to understand for customers which are not familiar with the possible service implementations. Furthermore, this approach does not support management from the customer's domain.

The service model used by the IETF in [**?**] focuses on the service implementation view, too. This RFC defines managed objects which allow the monitoring of service level agreements. The service level is modeled as a set of rules registered in a MIB. The violation of a rule is propagated by SNMP traps. This work focuses on monitoring the service implementation but does not cover the complete management of the service.

Another approach defines a model of electronic services for long–term relationships between customer and provider [**?**]. SLAs are negotiated between customer and provider to guarantee the availability of a service for repeated use in an open service market where one–time service usage dominates. The approach is based on virtual resources which represent the interface between customer and provider. They are mapped to physical resources on service usage. This approach specifies the interface between customer and provider hiding the service implementation but still focusing solely on the service usage. Interactions not concerned with usage like problem handling are not considered.

There also exists some work of the TeleManagement Forum on processes needed to manage telecommunication services [**?**]. The main focus in not the creation of SLAs but the automation of business processes. After completion this standardization effort can simplify the specification of SLAs.

No work for service level agreement specification regarding the interactions which support usage and management of services from the customer's perspective is known. The approach presented in this paper uses workflow concepts to support usage and management of services from the customer's domain and to enable a systematic design of SLAs.

4 Workflow Modeling

The most important standard body in the area of workflow technology is the Workflow Management Coalition (WfMC). This organization has defined a reference model [**?**] and terminology for workflow modeling [**?**]. Based on this work there exists a graphical representation for workflow models [**?**], [**?**].

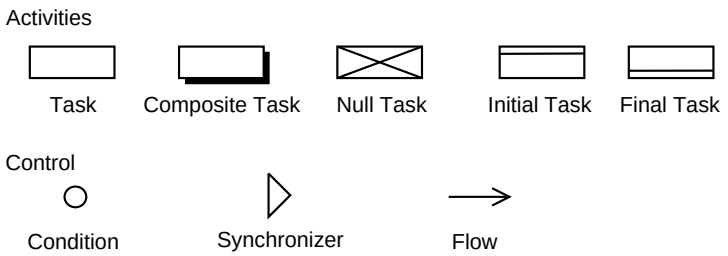

Figure 3: Symbols in Workflow Graphs

Workflow graphs are built of eight symbols depicted in figure 3. *Tasks* represent activities to be executed automatically by an application or manually by a person. Four special forms of tasks exist: composite tasks, null tasks, initial tasks and final tasks. *Composite tasks* represent the recursion in the model: A task can also be a workflow. *Null tasks* do not represent an action but are needed sometimes in a workflow graph for syntactical reasons. A workflow execution starts with an *initial task* and finishes with a *final task*. Such tasks can also be composite or null tasks.

The specification of a task includes the activity, resources executing it and data needed during execution. Resources are for example roles, applications, algorithms or hardware devices. Roles represent persons with mandatory skills and privileges.

The *condition* symbol represents a decision. A question, a resource answering it and the needed data must be specified for each condition. *Synchronizers* are used to join two or more parallel execution paths in a workflow, i.e. a synchronizer blocks until all paths leading to it have finished. A *flow* represents a possible flow of control during the execution of a workflow.

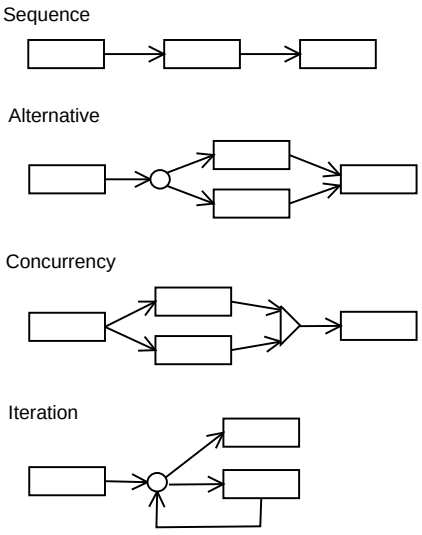

Figure 4: Graphs Constructs

To model a workflow the four constructs sequence, alternative, concurrency and interaction shown in figure 4 are needed. A *sequence* is an ordered list of tasks. After finishing one task the next is executed. The *alternative* construct models decisions. This construct selects one of several execution paths. *Concurrency* allows the representation of parallel activities, i.e. the execution of several paths in parallel. *Interactions* model repeated execution of a series of tasks. A condition included in the iteration path decides each time whether to continue the iteration or not.

5 Combination of Workflows and Service Level Agreements

A service level agreement should describe the usage and management interactions between customer and provider. These interactions, e.g. problem management, can be formalized as processes. Processes like production or business processes are modeled with workflow concepts. Therefore, it is reasonable to use these proven concepts to specify service level agreements, too. We use workflow concepts in the design phase as well as for the actual contents of the service level agreement.

Modeled business processes must be understood by people not familiar with workflow concepts. The same is valid for SLA design. Users and several experts need to be involved in the specification of the technical and organizational parts of the SLA. Workflow models are easy to understand and therefore a good base for communication of people with different knowledge. Furthermore, the resulting workflow graphs are a part of the actual contents of service contracts to support the constructive cooperation in the operation phase.

Each process as well as each activity in a process can have quality criteria which must be guaranteed and therefore defined in the SLA. Thus the quality parameters form an integral part of our process model which is used to model the interactions between customer and provider as well as to write the SLA.

5.1 Process Model

Each interaction between customer and provider consists of one or more processes which are modeled using workflow concepts. Our process model which is compatible to WfMC terms is depicted in figure 5. *Processes* consist of *activities*. To enable hierarchical decomposition a process itself is an activity. An activity needs *resources* to be executed on and *data* to work with. Both are abstract representatives to be mapped later on. Activities can generate *events*, e.g., for inter–process communication, synchronization of concurrent paths or for exception handling. Events are also triggers for other processes.

Additionally, the current state must satisfy *pre–conditions* before an activity is executed and the state must satisfy *post–conditions* after finishing the execution. Conditions are used for guaranteeing a consistent state before and after the execution of an activity. Events and data of the process as well as system data like current time can be used in conditions.

The so far defined parts of the process are highly reusable because all information is service-oriented, while the following mapping is customer dependent. The data needed by activities are mapped to variables and

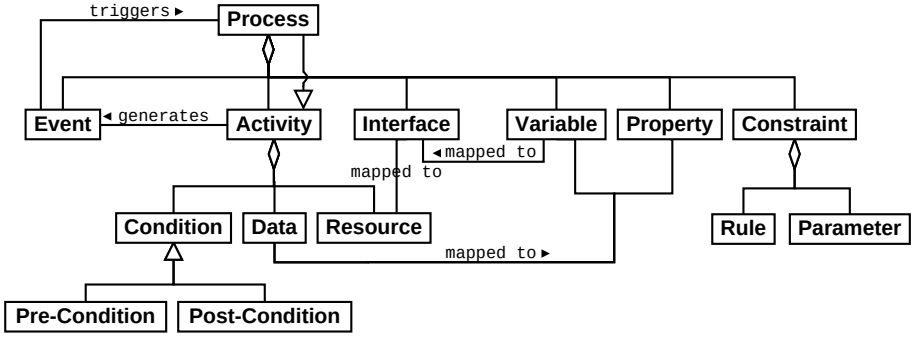

Figure 5: Process Model

properties. *Properties* represent values which are defined in the SLA. *Variables* are highly dynamic values which are exchanged through an *interface* between customer and provider at run time. An example for dynamic values are the user data in the authorization process while the maximum authentication time is a property.

The resources used by activities are mapped to *interfaces*. The definition of interfaces hides the implementation details. A resource representing a role can be mapped to a technical interface, e.g. defined by a telephone number, or to an individual contact person.

An important information in a service level agreement are *constraints*. In combination with properties they specify the service level. Constraints consist of *rules* with *parameters* which are also defined in the service level agreement. Rules can refer to values of variables, properties and system data as well as events. An example is "the overall delivery time of local email may not exceed n minutes during business hours", where "n minutes" and "business hours" are parameters of the constraint.

5.2 Process Classification

The processes can be classified to structure service level agreements. The process classes shown in table 1 have been identified so far in past and current projects. Depending on the concrete service there may be some of the classes missing but the complexer the service the more of them are necessary.

Services must satisfy some quality criteria to reach a service level acceptable to the customer. For example, in the presented scenario the response time must be small enough to query information during the sales conversation. Management processes must reach reasonable service levels, too, e.g. the problem resolution process must fix an error in a certain amount of time.

Class	*Description*
provisioning	installation of the service
usage	normal usage of the service
operation	service management of the provider
maintenance	plannable activities needed for service sustainment
accounting	collection and processing of accounting data
change management	extension, cancellation and change of service elements
problem management	notification, identification and resolution of service malfunction
SLA management	management of SLA contents
customer care	service reviews and user support
termination	deinstallation of the service

Table 1: Process Classes

Before the service can be used it must be installed. The *provisioning* class includes installation, test, acceptance and if necessary migration processes. Timeliness is an important quality parameter for processes in this class. *Usage* is the most common interaction. Service levels define for example the response time of the service.

Operation processes are invisible to the customer most of the time, but quality parameters are needed for them anyway, for example, the amount of time to detect a malfunction of a POP for ISDN calls is a quality criteria of

service operation. As the service is usually at least limited during *maintenance* activities the customer wants to limit e.g. the time slot for maintenance. The method used for *accounting* purposes is an important fact because it influences the price of the service.

Minor changes of the service are not uncommon in long–term relationships. This could be the extension of resources or addition of further dealers in the presented scenario. Thus a *change management* is necessary. A well working *problem management* is a very important factor in outsourcing. Problems can be solved faster if both partners are working together smoothly using well defined processes.

Often the requirements of the customer change during lifetime of the SLA. If these changes are known but not the point in time it is useful to specify a *SLA management* with processes for announcing the need, negotiating and changing the SLA. Communication on strengths and weaknesses of the service, its quality or future optimizations should be some of the subjects of regular *customer care* contacts. Besides, a help desk or other support offers are usually necessary for complex services.

Termination of processes for a relationship usually need to be specified if equipment of the provider is installed at customer locations or if support of the provider is needed for migration to the following service.

Additionally, it is useful to define some basic processes which are needed in many classes. Such processes are for example documentation, feedback and escalation mechanisms.

6 Example of a Modeled Process

The base of service level agreements design are customer processes which use and manage the service. The operation processes of the provider must fulfill the SLA, but their implementation is not an element of the SLA.

A resulting process model for a simplified example from the presented scenario is shown in figure 6. The example model is a part of the problem management class. It defines the process of fault identification, notification and resolution. The models of the complete process including cooperation during fault identification, documentation, prioritization, exception handling, escalation mechanisms, etc. would go beyond the scope of this paper. During identification of different activities the necessary resources and data must be defined which is also not detailed in this example.

A fault can be detected by both, customer and provider. Therefore, two initial processes exist. Depending on the party that detects the fault the process starts with the *fault detection* on the customer or the provider side.

If the customer detects a fault, the next activity is the *notification* of the provider. If the provider detects a fault the customer is notified by the *customer notification* process. The customer notification is modeled as a composite task because it is a basic process controlled by dynamic values like the message and the receiving role to adapt to the respective situation.

At this point the two paths join. The next activity is *trouble ticket creation*. Then, the concurrency construct models the fault resolution with regular feedback to the customer on the current state. The feedback path is an iteration construct monitoring the trouble ticket state. As long as it is open the *notification* process is repeated. The feedback interval is realized by a post-condition.

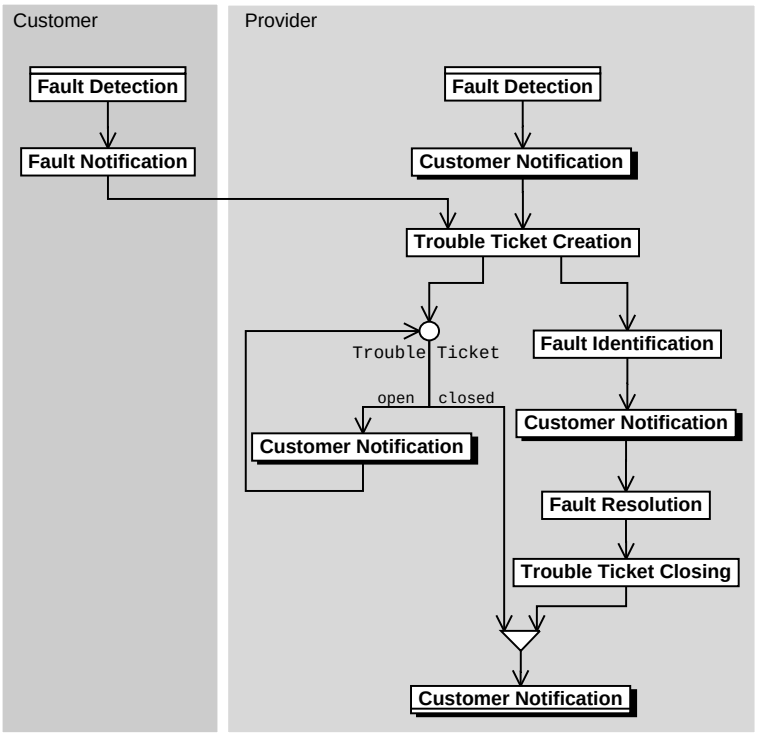

Figure 6: Fault Notification Process

The fault resolution path starts with the *fault identification* activity. This activity is separated from the *fault resolution* because the customer usually wants feedback on the estimated duration and scope of the service malfunction as soon as possible which therefore is modeled explicitly by the *customer notification* process.

The identification and resolution activities by itself are of no interest to the customer, but the customer wants to define quality criteria on this activities. Therefore, they must be modeled. After the problem is solved the *trouble ticket* is *closed* which ends the feedback loop and the customer is notified of the resolution in the final *customer notification* process.

The resulting process model including the data and resource definitions is part of the service level agreement. In the next steps data and resources are mapped to properties, variables and interfaces. The necessary interfaces are easy to find because each domain boundary crossing is visible in the workflow graph. The only new interface added by this process is the fault notification interface which could be mapped to a telephone interface.

All requirements for the service level not expressed by now must be specified as constraints. For each combination of activities it is necessary to consider if there are important constraints which should be represented in the SLA. In the presented example process a constraint on the total time from trouble ticket creation to trouble ticket closing should be covered by a constraint. Additionally, to limit the amount of service failures, e.g. a constraint for the number of trouble tickets open simultaneously and in certain intervals is needed. Several other constraints are thinkable, too.

7 Conclusion and Future Work

This paper proposes the application of workflow concepts for the specification of SLAs concerning IT services. The main points of the approach discussed are that it supports the identified requirements for long–term service level agreements which need to be customer–centric and constructive. Supplementary, this approach generates some additional advantages, like the explicit statement of resources, interfaces, competence and information flows supporting automation efforts and optimization of resource consumption. The most attractive benefit of the workflow concepts is the active support of cooperation for usage and operation resulting from the instructive nature of workflows.

The approach is currently verified in an industry cooperation with a major provider of telecommunication services. The necessary effort sets the main focus especially to extensive, custom designed services. But we are researching ways to simplify the reuse of this effort.

Further research is concerned with designing an application methodology and a methodology for extraction of characteristic capacity and quality parameters. The target is the systematic derivation of customer–oriented but measurable quality parameters.

Acknowledgment

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 directed by Prof. Dr. Heinz-Gerd Hegering is a group of researchers of the University of Munich, the Munich University of Technology, and the Leibniz Supercomputing Center of the Bavarian Academy of Sciences. Its Webserver is located at `http://wwwmnmteam.informatik.uni-muenchen.de`.

References

[1] P. Bhoj, S. Singhal, and S. Chutani. SLA Management in Federated Environments. In M. Sloman, S. Mazumdar, and E. Lupo, editors, *Integrated Network Management VI (IM'99)*, Boston, MA, May 1999. IEEE Publishing.

[2] H.-G. Hegering, S. Abeck, and B. Neumair. *Integrated Management of Networked Systems – Concepts, Architectures and their Operational Application*. Morgan Kaufmann Publishers, ISBN 1-55860-571-1, 1999. 651 p.

[3] T. Preuß and H. König. Service Supplier Relations for the Outsourcing of Information Processing Services. In *Proceedings of the IEEE Enterprise Networking and Computing Conference (ENCOM 98)*, Atlanta, GA, USA, 1998.

[4] W. Sadiq and M. Orlowska. Modeling and Verification of Workflow Graphs. Technical Report 386, University of Queensland, November 1996.

[5] W. Sadiq and M. Orlowska. On Capturing Process Requirements of Workflow Based Business Information Systems. In *Proceedings of the 3rd International Conference on Business Information Systems (BIS99)*, Poznan, Poland, April 1999.

[6] SMART TMN Telecom Operations Map. Evaluation Version 1.1 GB910, TeleManagement Forum, April 1999.

[7] The Workflow Reference Model. TC Document 00-1003, Workflow Management Coalition, January 1995.

[8] Terminology & Glossary. TC Document 1011, Workflow Management Coalition, February 1999.

[9] K. White. RFC 2758: Definitions of Managed Objects for Service Level Agreements Performance Monitoring. RFC, IETF, February 2000.