Management of Service Level Agreements using INSMware

M.H.Knahl and S.M.Furnell

Network Research Group, University of Plymouth, Plymouth, United Kingdom
e-mail: knahl@jack.see.plym.ac.uk

Abstract

The paper presents a component-based approach to implement a new framework for Integrated Network and System Management and its application on management services to implement Service Level Agreements (SLAs). Applications are assembled from a set of pre-fabricated components rather than being developed from scratch and are being incorporated into new management domains. In this paper, we describe a project, componentware based Integrated Network and System Management (INSMware), which was built using only component-based techniques. We describe an approach that integrates the software component paradigm with network and systems management and its application upon SLAs. We focus upon the monitoring of SNMP-capable network elements. The requirements for new management services implementing SLAs are outlined and a prototypical implementation is presented.

Keywords

Service Level Agreements, Network Management, System Management, Componentware, SNMP Networks

1. Introduction

The disciplines of Network and System Management encompass all actions taken to enable and guarantee the maintenance and operations of the resources - either hardware or software - in a network. This includes the communication network as well as the server and the end-systems in a network. ITU-T defined five management categories (namely Fault, Configuration, Performance, Accounting and Billing, Security Management) that define the different disciplines and requirements for the management of heterogeneous networking environments.

This paper presents research leading to a component-based framework for Integrated Network and System Management (INSMware) and its application upon the management of Service Level Agreements (SLAs). The transferability of INSMware to other management domains is realised because of a consistent component-based development approach to meet the requirements for integrated management services (Knahl et al. 1999). There are two versions of INSMware: one using the CORBA (OMG, 1998) component model and a Microsoft DCOM (Brown and Kindel, 1996) implementation. This allowed us to study both middleware architectures in detail.

The aims of INSMware are to hide the complexity of the heterogeneous network environment and underlying technologies and to provide a universal framework for the various management services. In addition, INSMware can be applied to several application domains within the INSM area. In this paper, we present the application of INSMware to the management of SLAs.

2. Integrated Management of Service Level Agreements

Limitations and restrictions of existing Network and System Management frameworks, such as distribution of the management services, adoption and integration of new services can be overcome by providing a component-based approach (Knahl et al. 1998; Knahl et al. 1999). Furthermore, the management must be configurable to enable the provisioning and management of Service Level Agreements.

2.1 Service Level Agreements

Service Level Agreements (SLAs) are formal negotiated agreements that help to identify expectations, clarify responsibilities and facilitate communication between a service provider and its customer (Karten, 1998). In a typical customer / provider relationship, a customer demands (and pays for) specified services and for a Quality of Service (QoS) that are defined in the SLAs. To enable (and prove) the fulfilment of those SLAs it is necessary for the provider to have management services that can monitor and control the service status.

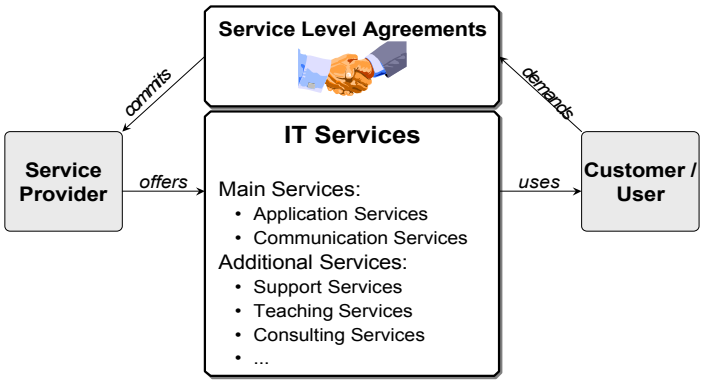

Figure 1: Service Level Agreements

In order to observe the quality of delivered services it is necessary to negotiate QoS parameters as well as modes for the measurement and evaluation of these parameters. QoS parameters must reflect the expectations of the customer and they are part of the SLAs between the customer and the provider. SLAs are for several purposes, e.g. if a service is not delivered with the specified quality the customer may get a discount. Therefore, the SLAs and QoS parameters have to be supervised by the management system of the provider. Further, the provider is obliged by the SLA to report the compliance with agreed QoS parameters. Customer Service Management (CSM) offers a management interface between customer and network provider which enables the customers to monitor and control their subscribed services (Langer et al. 1998).

Examples for SLAs customer/provider relationships could include:

- Network provider (e.g. Deutsche Telekom, a network operator who is acting as a service provider) and customer (e.g. PanDacom, a system integrator with branches all over Germany who is using the services offered by this or another service provider);
- System Integrator (e.g. PanDacom as a system integrator that is offering remote management services based upon QoS parameters to its customers) and customer with service contract.

One example for such QoS parameters is the response time which implies the connectivity or the availability of a certain service where these parameters have to be measured and valued from the customer point of view when the customer uses a specified service, which requires parts of the management system to be installed at the customers side (e.g. intelligent agents or management gateway that report to the provider's management framework). This allows the

monitoring and measurement of the services from the customers side, e.g. from an end-system to monitor end to end-connectivity.

SLAs will always change in the course of time due to new requirements for services that demand modified or new QoS parameters. It must be possible to extend the management framework with additional functionality, e.g. for the configuration and monitoring of a new QoS parameter. It is, therefore, important that the architecture is flexible and that it enables fast and cheap integration of these new services. For large scale, distributed networks it is also essential that the management system scales well. Furthermore, the variety of the different hardware and software in existing and future networks requires a high-degree of platform independence for the management system.

2.2 Integrated Management Architecture

Distributed systems allow the partitioning of applications into logical and physical self-contained entities: distributed objects. These distributed objects represent a part of the system-global object model. The possibility to use distributed objects for the realisation of different applications makes them into so-called software components. A software component is a piece of software with one or more well-defined interfaces that is configurable, integrable, and immutable (Langer et al. 1998, Hofmann 2000). By configurable we mean being able to set parameters affecting the properties of a component without requiring its modification. The integration of software component means the connection of incoming and outgoing interfaces, i.e., interfaces being used in the client role and in the server role of a client/server component communication while immutability requires a component to be *physically immutable*. Such physically immutable forms of software are, for example, executable files or dynamic link libraries (DLLs).

The most important criterion of our component definition above is immutability since it allows a dissociation from object-oriented concepts such as Classes and Design Patterns. The functionalities of presented management framework consist of the processing, filtering, and analysis of management relevant data, the presentation in a Graphical User Interface (GUI) and user notifications at the occurrence of predefined states that represent important network states. The system allows that one or more users may be notified over varying communication channels.

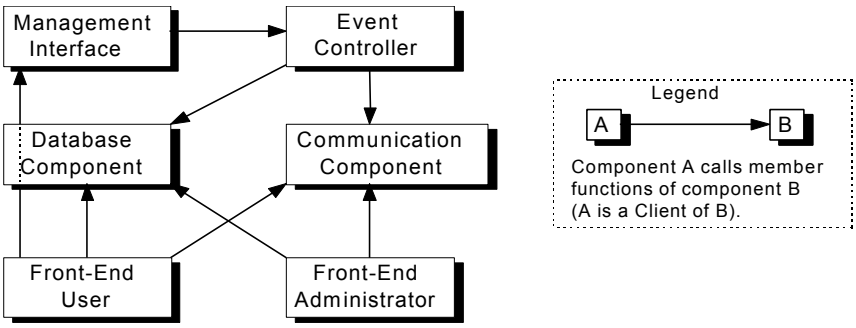

Figure 2: INSMware Components and their Connectivity

The design of the individual INSMware components (see Figure 2) is based on a domain specification which subdivides the entire application domain into subdomains. The data processing system requires a connection to a data source (physically existing system). This is realised by the Management Interface component which exists, similar to device drivers of an operating system, in several different forms and is configurable, as required, for different SLAs. The Management Interface component is installed at the customer side to monitor the specified SLA. The filtered service level data is then forwarded to the SLA provider's

management framework using the services of the underlying middleware technologies. The Management Interface component interprets the received data, filters and analyses it, and notifies the event controller component when particular pre-defined exception states occur. Data storage is accomplished by a call to the database component and user notification is effected over the communication components. It must be emphasised that all information about the users that need to be notified (e.g., access to user, user's role regarding the monitored processes) are stored in the system. The communication component itself consists of a set of several components that again implement subdomains, e.g., sending of faxes, voice mails, e-mails. The user can visualise system states by using the front-end user component and can maintain the system by using the front-end administrator component.

2.3 Application Domains and Sub Projects

INSMware was originally conceived to monitor the various network elements and to provide management services in heterogeneous network environments. With INSMware, a timely user intervention in the running of the network processes is made possible when required (e.g. user notification when a network critical condition occurs).

INSMware was applied to the application domain of monitoring SLAs. SNMPv1 and SNMPv2 based managed objects at the customers side can be monitored and relevant management data is then forwarded to the management framework. Two of the components, namely the Management Interface component and the front-end user component, have to be modified to integrate SLA services (e.g. SNMP) into the management framework and the database structure has to be adapted to the service relevant information while the remaining four components can be reused with no modifications. The application domain implemented by the Management Interface component is actually very limited and a universal Management Interface component was developed which can be adapted to different systems by configuration (Amrhein, 1998). The graphical data representation supported by the front-end component to visualise the management information has to be adapted to the respective service domain and remain user-friendly and simple (Vierow, 1998). By generating source code responsible for inter-component communication, a tremendous reduction in the development time for front-end components was possible. Using a CORBA/ DCOM bridge, a platform-independent connection of partially generated front-end components is achieved.

3. INSMware

3.1 Customer scenario

The scenario provides an insight into a whole range of different Management and SLA requirements (see Figure 1). First of all, each provider must manage its own network. An integral part of this is network element management, which concerns the supervision of the availability, capacity utilisation and fault-free operation of the network elements. Added to this is the functioning of the network as a whole. At the access to a network, the providers aim to offer their customers services with a certain Quality of Service based on an SLA. The constant monitoring of service quality is a management task. The management of the customer / provider interface also includes procedures for fault-reporting and for service adaptation or service provisioning. It is essential that customers have access to specific management information (e.g. service quality, service availability) because this is the information they need if they themselves want to develop added value and other new services based on the network services they are already using. For customers, it is the service related information based on the customer SLA that is generally interesting rather than the 'raw' data from the component management of their providers.

In principle, SLAs should exist for all the services of a provider's service offering and used by customers. The SLA contains an exact description of what is offered in a service and defines

which costs are applied when a customer uses the service. Since providers of networked systems (e.g. of an enterprise network) are just now slowly being accepted as IT service providers, a large number of services in the IT area are still being used without explicit SLAs. Furthermore, customers require these specifications for planning the use of the IT services in their own business processes (Corsten, 1997).

The current framework implements the Simple Network Management Protocol (SNMP) (Case et al. 1990) to monitor and control managed objects for the provisioning of the SLAs. SNMP is a set of network and system management standards that describe the asynchronous requests and responses for the exchange of management data between SNMP management objects (Stallings, 1998). Virtually all major vendors of end-systems, workstations and network devices such as routers and switches offer SNMP support. In addition, enhancements to the initial SNMP have been pursued in a number of directions (e.g. RMON, SNMPv2, SNMPv3).

The SNMP Manager is network management software that implements the SNMP protocol and is represented in our case by the Management Interface component. The SNMP Agent resides in a managed network element, such as a router or switch or in an end-system such as a PC or Unix Workstation. The agent stores management information in the Management Information Base (MIB) and processes SNMP requests from the SNMP Manager and responses from the agent itself. The proposed INSMware framework is based on a multi-lingual SNMP implementation (Levi et al. 1999). The multi-lingual implementation of the management interface supports the different SNMP versions and enables the seamless integration of the (typically mono-lingual) SNMP based managed objects. Besides that, additional protocols or access policies can be integrated into the Management Interface component.

3.2 Management Interface component

This section describes the Management Interface component for the management of SLAs. The Management Interface component can be installed at the customers side to collect and analyse the management information. It then forwards the required management data to the SLA provider's management framework where it is processed and the required management tasks are undertaken.

3.2.1 Architecture of Management Interface component

One advantage of component-oriented software is the relatively easy reuse of individual parts of the system to adapt the framework for different SLAs, e.g. Network Element Monitoring or Service Management (see Figure 3) . Hence, the SLA provider can reuse the existing systems to offer services to different customers. Modifications to the system to integrate new services and reuse existing components are required on the graphical user interface to represent the service states and the interface to the managed network (Management Interface component for the communication with the managed objects).

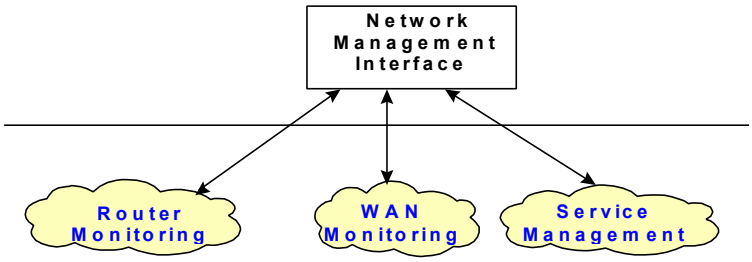

Figure 3: Management Interface Application Domains

To improve and optimise the reusability of the Management Interface component it is split into several smaller components. The filtering functionality analyses the data from the managed objects and decides whether a critical value has been exceeded or whether a critical event has occurred. This evaluation can proceed without further knowledge of the underlying technology (e.g. whether SNMPv1 or SNMPv2) of the managed objects because solely the protocol independent data stream (from the managed objects) has to be analysed. The SNMP component provides the initialisation and de-initialisation of a connection with a managed object. This is individual for every kind of managed object with different management protocols, e.g. differs the initialisation of a connection and the access to a SNMP managed object very much from the access to a file system and has, therefore, be abstracted. For each access technology, a specific component is developed that implements a specific defined interface and which can be used from the general Filter component. This specific component is responsible for the communication with the managed object and transforms/translates the received data into a format that can be used by the Filter component. Whilst developing different management domains it has been discovered that a few parts – particularly in the area of the interface implementation – are similar. For these similarities, source code can be reused. Beside that, the Management Interface offers a high degree of flexibility and provides good reuse for additional and future management domains.

3.2.2 Distribution of Management Interface

The INSMware Architecture allows the distribution of its components over the network. The separate components can also be used for the different tasks involved in SLA Management. The collection and filtering of SLA relevant data at the remote locations reduces the management related traffic over the network. The distribution of management procedures means the realisation of the Client/Server principle on the software level. The different software components of a distributed software system may act as a client or a server, or also as a client and a server if the application requires this.

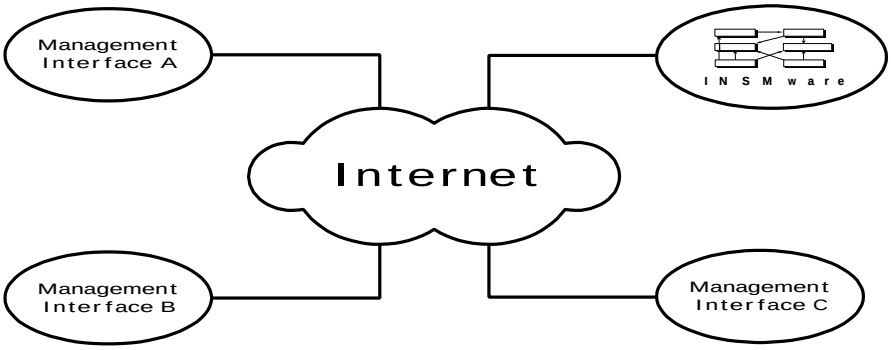

Figure 4: Distribution of Management Interface via Internet

The realisation of distributed software systems is made possible by the increase of low-cost bandwidth on Wide Area Networks (for example the Internet, a new generation of network-enabled desktop operating systems and middleware technologies, such as CORBA and DCOM, that enable the seamless distribution of software components). The possibility to access remote resources (e.g. via the Internet) enables software developers to distribute their software components, even over the boundaries of a corporate network (see Figure 4).

3.2.3 SNMP-Interface

The analysis and design stage of the SNMP-Interface was based on the layered model of the Management Interface. Dynamic access mechanisms have been implemented to read the configuration from an ODBC database. The prototypical implementation of the Management Interface component is written in C++. The SNMP functionalities have been implemented

using two different SNMP frameworks based on C++: the SNMP interface of the Microsoft Foundation Classes (MFC) which only implements SNMPv1 and the SNMP++ framework from Hewlett Packard which offers support for SNMPv1 and SNMPv2c (Mellquist, 1997). Functionalities of the implementation include the Monitoring and collection of SNMP Traps and monitoring of SNMP managed objects using SNMP Get / GetNext. This makes it possible to actively monitor and control changes within the SNMP agents.

New SNMP functionalities such as SNMPv3 start to occur. These can be implemented into the SNMP++ framework and, therefore, with minor modifications into the INSMware framework (Katz, 1999). Figure 5 illustrates the layers of the Management Interface and the integration of the different SNMP frameworks into the Management Interface.

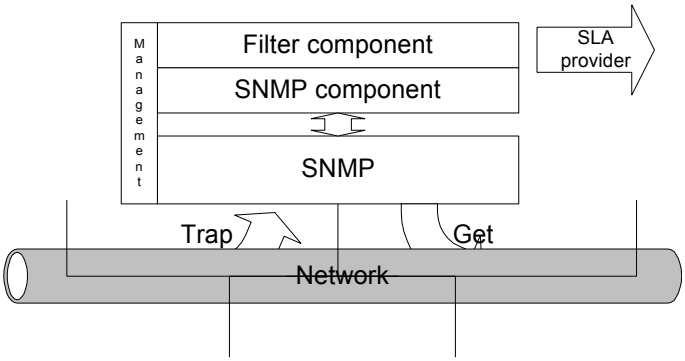

Figure 5: Management Interface layers

The INSMware management framework offers management services for the collection and monitoring of SNMP-Traps and offers services for SNMP Get and GetNext operations. This enables INSMware to act as an SNMP Manager. Traps, which are sent from SNMP managed objects such as routers or even from software in the network, are collected and further processed for the Filter component. The SNMP component then forwards the data to the Filter Layer for further analysis. The Filter component then analyses the data using the configuration parameters from the INSMware Framework. The filtering uses the source address as well as the Object ID (OID) of the trap, e.g. to discover a cold-start of a managed object.

The collected management data is compared against predefined values using different operators (e.g. <, >, =) to discover status or administrational changes such as change of System Administrator or critical network conditions such as high collisions on an Ethernet or the failure of a WAN connection. When the Management Interface discovers - due to an incoming Trap or a MIB value - that a critical event occurred it protocols the event and arranges the notification the related INSMware user on its GUI or forwards the notification to a manager via telephone or E-Mail. The protocolisation and messaging functions are provided by the Event Controller and the Communication Component.

The specification of the relevant data for the managed objects and events - for the configuration of the SNMP managed objects (e.g. IP-Address, SNMP Version and Community) and the related Get/GetNext and Trap events (e.g. OID, Value, Operator) - is fully implemented via the GUI and saved in the management database (implemented in Microsoft Access). The type of notification can be dependent on different parameters (e.g. dependent of day or time) and can be configured for each individual event that occurs. Furthermore, it is possible to set the intervals for the polling and controlling of events according to the requirements and to individually add/delete intervals according to specific management requirements.

3.3 Visualisation of Network infrastructures and service states

As previously mentioned, the second component of INSMware that has to be adapted for new services in the domain of Integrated Network and System Management is the user and administrator front end. Two different design approaches were initially considered: A pure graphical approach that uses overview maps to represent the network structure as known from commercial Network and Sytsem Management platforms such as HP OpenView (see Figure 6) and a more Microsoft Windows Explorer-like view, showing the network structure as a tree (see Figure 7). The first approach visualises the network structure very clearly, particularly for experienced users of management platforms such as HP OpenView. A lot of network management tools use such a map-based interface and for this reason they are well known for experienced users, but they are also very space consuming and become unclear for really large networks, e.g. if an internetworking map consists of 30 routers and 50 networks.

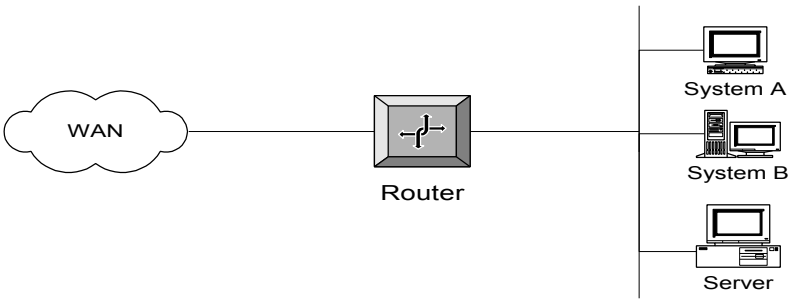

Figure 6: Map representing Network Infrastrucuture

The tree view is less space consuming and, for this reason, more suitable for large networks. A problem of this approach is that the network structure is not always hierarchical (e.g. cross links) and, therefore, the tree representation might not represent the logical and physical network connections as clearly as the established network maps.

To get the best of both worlds the implemented front end is a mixture of both approaches. The main user interface is shown in Figure 7. The network structure is represented by a tree on the left hand side. On the right side, four tabs show information about the node that is currently activated. If desired, the user can demand an overview map by pressing the “Network View” button which then shows a network map for the actual network or domain.

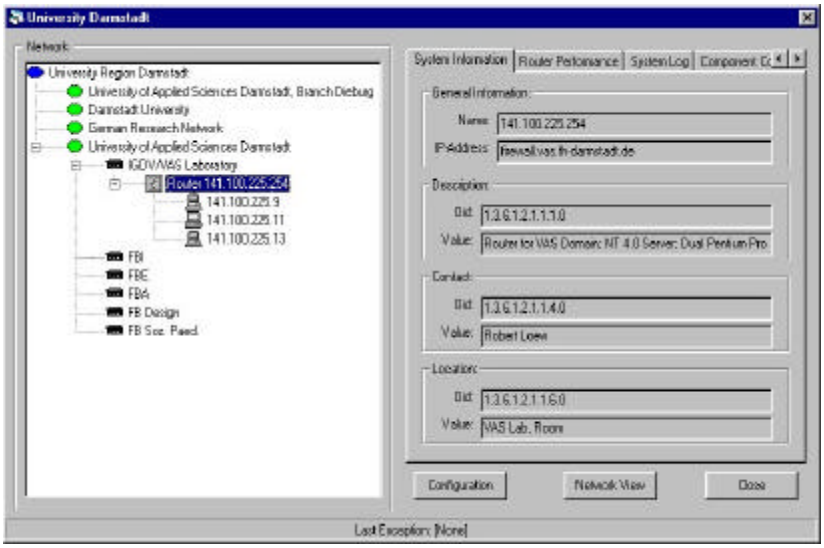

Figure 7: User interface representing network strucures and states

The tree view is configurable for every single user. Every user can name the nodes in his preferred way and it is also intended to make the part of the network that is shown by the tree determinable by the user. In order to enable such flexibility, all the data that is presented by the front-end is stored in the system database. This concerns the entire network structure – represented by the tree – as well as the user settings for the configuration.

The four tabs at the right hand side of the GUI provide quick access to the most important information of every network node. The “Information” tab displays the basic information like node name, description, location and contact person. The second tab visualises the performance of the network node. The network traffic, utilisation and error rates of the whole network or every single port in case of a router or a switch are presented. The “Log” tab displays an event log for every node. Restarts, port failures and breakdowns of the system will be registered with date and time. ‘Component configuration’ allows configuration of managed objects, e.g. to reset a network interface.

The information presented (except Component Configuration) provides the management data and are for monitoring and controlling the managed Service Level Agreements. The configuration facilities enable the user to configure the management services. The selection and monitoring configuration of specific MIB variables specifies the management services for specific managed objects. In addition, SNMP traps can be configured to monitor and configure events reports from the SNMP entities themselves. The user can define notification events by specifying threshold, notification channel (voice, fax, pager, email) and priority. All the configuration data – including the settings for the notification events – are stored in the system database.

4. Summary and Outlook

Our future INSMware research aims at the extension of INSMware Management Services to provide a unified, integrated management framework. INSMware's current shaping focuses on the management of hardware components. However, this represents only a reduced view of real world information systems, since there is a correlation between managed hardware components and software components operated on the basis of that hardware. Hence, future versions of INSMware will also integrate the management of software components and thus provide fully integrated management facilities.

Furthermore, the development of extended and new forms of user interaction, including the integration of Web-based management services and the integration of handheld devices such as the Palm Pilot to realise ubiquitous computing facilities will be considered. As no component models currently exist for such handheld systems and their operating systems, one task will be to develop appropriate integration mechanisms. The first prototype of the front-end component is realised as a Visual Basic program, but there are thoughts to produce a web-based front end in order to allow management and configuration from everywhere without installing client software. Another effort in this area will be the integration of speech input and output with INSMware, which is to facilitate system operation in scenarios in which no computing device is available. It also provides a more intuitive way of INSMware utilisation to the user.

5. References

Amrhein, M. (1998), *Wiederverwendung von Softwarekomponenten — dargestellt am Beispiel eines Überwachungssystems mit Sprachausgabe (Reuse of software components - described by way of example of a monitoring system with speech output)*. Diploma Thesis, University of Applied Sciences Darmstadt/Germany.

Brown, N. and Kindel, C. (1996), *Distributed Component Object Model Protocol – DCOM/1.0*. Microsoft Corporation, Network Working Group.

Case, J. D. , Darwin, C., Fedor, M. , Schoffstall, M. L. (1990), *A Simple Network Management Protocol (SNMP)*. Request for comments (Standard) RFC 1157. Internet Engineering Task Force. May 1990.

Corsten, H. (1997), *Management von Geschäftsprozessen: Theoretische Ansätze – Praktische Besispiele*. W. Kohlhammer GmbH. Stuttgart, 1997.

Hofman, H.D. (2000), *Software Component Reuse by Adaptation*. PhD Thesis. Institute Of Technology, Cork, Ireland. March 2000.

Karten, N. (1998), "How to Establish Service Level Agreements". Karten Associates. Randolph, MA, USA, 1998.Katz, J. (1999), "SNMPv3 Support for SNMP++". *The Simple Times*. Volume 7, Number 1. March 1999.

Knahl, M. , Bleimann, U. , Furnell, S. M. , Sanders, P. W. (1998), "Integration of ATM management procedures into native integrated network and systems management architectures". *Proceedings of the International Network Conference 1998*, Plymouth, UK, July 1998, pp91-97.

Knahl, M. Hofmann, H. D. and Phippen, A. (1999), "A Distributed Component Framework for Integrated Network and System Management". *Information Management and Computer Security*, Vol. 7, No. 5, pp254-260.

Langer, M., Loidl, S. and Nerb, M. , (1998), "Customer Service Management : A More Transparent View to your subscribed services". *Proceedings of the 9th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM 98)*. Newark, USA. October 1998.

Frye, R. , Levi, D. , Routhier, S. , Wijnen, B. (2000). "Coexistence between Version 1, Version 2 and Version 3 of the Internet Standard Network Management Framework". Internet Engineering Task Force, SNMPv3 Working Group, RFC 2576. March 2000.

Mellquist, P.E. (1997), *SNMP++: An Object-Oriented Approach to Developing Network Management Applications*. Prentice Hall, 1997.

OMG. (1998), *The Common Object Request Broker: Architecture and Specification, Revision 2.2*. OMG Document 98-07-01, Object Management Group, Inc.

Stallings, W. (1998), "SNMP and SNMPv2 : The Infrastructure for Network Management". *IEEE Communications Magazine*. March 1998.

Vierow, T. (1998), *Entwicklung eines Generatorwerkzeuges zur Unterstuetzung der Gestaltung von graphischen Benutzeroberflaechen (Development of a generator tool for supporting the design of graphical user interfaces)*. Diploma Thesis, University of Applied Sciences Darmstadt/Germany.