An Integrated Management Architecture for Heterogeneous Networks: INSMware

Martin Knahl ✧, Prof. Dr. Udo Bleimann *, Dr. Holger D. Hofmann†,
Dr. Steven Furnell ✧

e-mail: knahl@vas.fh-darmstadt.de

Keywords: *Network Management, System Management, Componentware, SNMP.*

Abstract: *The Component-based approach to develop distributed software represents a new paradigm in software engineering. This approach is used to implement a new framework for Integrated Network and System Management for heterogeneous networks. Future Management Systems will be derived from a set of pre-fabricated components rather than being developed from scratch. In this paper, we present research in the area of Componentware based Integrated Network and System Management and the research prototype INSMware, which was built using only component-based techniques and which represents a research prototype that was used for investigating such component-based network management techniques. We describe an approach that integrates the software component paradigm with network and systems management. Its integration upon different networking technologies is outlined. We focus on the monitoring of SNMP-capable network elements.*

1 Introduction

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 [1], [4]. The impact and leverage of distributed systems technology is prevailing not only for design and implementation of user applications and services but indeed also for the benefit from deploying management systems. Thus far, management systems have typically been of two categories. Either specialised along one dimension (e.g. vertically, targeting one or a few management aspects such as billing or performance, or horizontally, dedicated to management of a specific layer such as network elements) or have they resembled monolithic "main frames" based to a large extent on proprietary solutions.

Existing management solutions, based on the integrated manager or platform approach, do not meet the requirements and are complex. The development and provisioning of integrated management services proved to be too complex. A simpler solution is required and proposed: Management is Lego™, Management is distributed components - thinner layers, higher reuse potential of existing solutions, improved potential of easy integration of new technologies and therefore less effort to integrate existing and new technologies. Componentware is an enabling technology to meet

✧ Network Research Group, Department of Communication and Electronic Engineering, University of Plymouth, Plymouth, United Kingdom

* Department of Computer Science, University of Applied Sciences Darmstadt, Darmstadt, Germany

† Department of Mathematics and Computing, Cork Institute of Technology, Cork, Ireland

the requirements and architectural principles of the proposed framework and of the prototype implementation (INSMware). The research uses contemporary distributed technology to leverage a modular approach to design management systems, thus facilitating openness and extensibility on one hand and adaptability, i.e. customisation of management services, on the other.

Distributed object-oriented systems [14] represent the logical development of the object model supporting the distribution of objects to physically distinct locations. Distributed object-oriented architectures, also referred to as middleware architectures, such as OMG CORBA or Microsoft DCOM form the basis for the development of distributed object-oriented systems. Providing a mediation layer for distributed object communication can be enhanced by the provision of a framework to support reuse. *Software components* are distributed objects that are designed to be reused. They represent physically immutable pieces of software that can be assembled with other components to form new, more complex components or even entire applications. The latter is called *componentware*.

The development and operation of component-based systems requires an architectural basis in the form of a componentware architecture. The main mechanisms that have to be realised by such an architecture are those for component search and description, for component management, for component composition, and for component configuration.

The use of software components on an enterprise level enables the use of distribution, software reuse, security services and other functionality. Distribution allows software components to communicate with each other over electronic networks (such as local area networks or the Internet). Modelling aspects and the functionality of a distributed, component-based Integrated Network and System Management framework will be discussed to prove the relevance to build a novel management framework using software components, namely INSMware.

The presented approach, leading to the componentware based Management Framework INSMware for Integrated Network and System Management, also enables the monitoring of component-based applications. It is a building block of an enormous effort in research and industry towards the integrated management of all resources in a distributed system. The often-cited ITU-T Management Framework that categorises the management tasks into the five classes Fault, Configuration, Accounting, Performance and Security Management, only gives a functional decomposition of this multi-dimensional problem. However, another important problem dimension is the resources or components within a distributed system upon which the management functions are applied. In the past, mostly the logical communication resources (e.g. communication protocol entities) and the physical network components (e.g. repeaters, bridges, routers) have been addressed by the standardisation bodies and industrial products. Consequently, the term network management is commonly used to cover these aspects. In the meantime, hardware and software resources of the end-systems (e.g. disk space, CPU time of applications) are included in the management view expanding the network management to systems management.

Beside the integration of established management procedures, the proposed framework requires functionality to be instrumented in order to enable the monitoring of distributed applications - one step in the direction of application management. However, purely isolated component-oriented management is not what the end-user expects. Integration of management solutions with the aim of one single architecture that may cope with all type of resources is demanded. For instance, by means of commercially available network management systems, network operators get a management view upon the network resources and thus helps to answer questions about the reachability and availability status of network devices and end systems or the load on the network segments. Additional knowledge is necessary in order to close the gap between the applications and the network; knowledge like, which application components are running on which end systems, what is the communication behaviour of the applications or how much load is produced by the individual applications.

To gain this knowledge is currently very tedious, as the mapping relations between applications and network resources is often changing, possibly without the knowledge of the network operator. The situation will become even worse when object migration technologies or automatic fault recovery procedures are widely used. What is required is a common and integrated view upon all resources of a distributed system on the network, (operating) system, middleware and application levels.

This paper presents research leading to a componentware-based framework for Integrated Network and System Management (INSMware). 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. There are two versions of INSMware: one using the CORBA [2] component model and a DCOM [3] implementation. This allowed us to study both middleware architectures in detail.

2 The INSMware Project

This section presents the INSMware project. The requirements for integrated management services for heterogeneous networks are identified and outlined. Then, the architecture of the INSMware framework is presented, followed by a discussion of the application domains and sub projects.

2.1 Integrated Management Services

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 (see Figure 1).

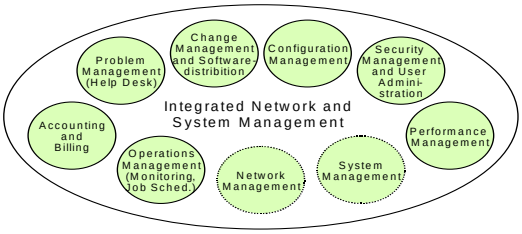

Figure 1: Management Disciplines

Integrated Network and System Management includes the communication network as well as the server and the end-systems in a network. The Telecommunications standardisation sector of the ITU (ITU-T) defined five management categories - Fault, Configuration, Performance, Accounting and Billing, Security Management - that define the different disciplines and requirements for the management of heterogeneous networking environments.

Required management services include configuration of network elements, monitoring and controlling of network elements, software distribution (e.g. software distribution to PCs), user management, security management (e.g. access control), service management (e.g. video) and so on. The complexity of the management services is related to the complexity of the environment to be managed.

The research has shown that management standards, such as SNMP or today's Management-Platforms like Tivoli TME 10 or HP OpenView Network Node Manager, basically cannot handle the end-to-end management of ATM networks [4]. The challenge for the network operators is how to migrate from these restricted systems to an architecture that integrates the different network technologies, i.e. the TCP/IP and the ATM management model, and that will enable them to meet the goal of providing seamless, end-to-end connectivity and management for tens of thousands of users across LANs and WANs. It has been found that this seamless architecture presents problems and raises challenges in just about every area of network management [7].

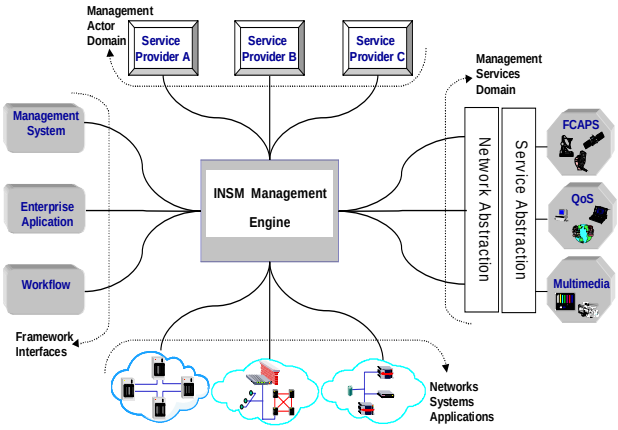

Figure 2: Integrated Management Framework

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 as illustrated in Figure 2. In addition, INSMware is designed to facilitate the easier addition and integration of new networking technologies and management services by employing component-reuse.

2.2 INSMware 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. 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 (Black-Box Reuse) since it allows dissociation from object-oriented concepts such as Classes and Design Patterns (White-Box Reuse). The functionality of management framework presented consists of the processing, filtering, and analysis of management relevant data, the presentation in a graphical user interface 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.

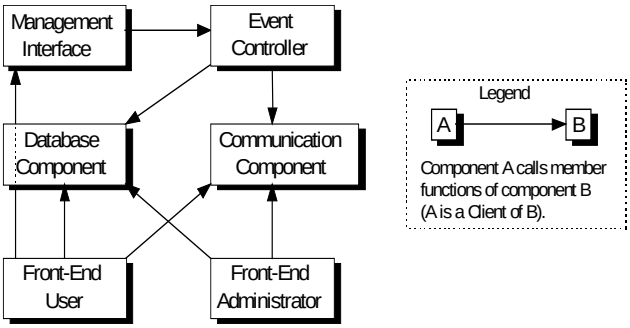

Figure 3: INSMware components and their connectivity

The design of the individual INSMware components (see Figure 3) is based on a domain specification that subdivides the entire application domain into subdomains. First, the data processing system requires a connection to a data source (physically existing system). This is realised by the Management Interface component that exists, similar to device drivers of an operating system, in several different forms and is configurable, as required, for different data sources.

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) is 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 TCP/IP and ATM 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).

To test the workability of the component-based approach and to examine aspects of reuse at component level, INSMware was applied to the application domain of monitoring SNMPv1 and SNMPv2 based managed objects. Two of the components, namely the Management Interface component and the front-end user component, have to be modified to integrate new management services (e.g. SNMP) into the management framework. The database structure has to be adapted to the management relevant information,

while the remaining four components can be reused with no modifications. Adaptation is required in the Management Interface component since the data acquisition mechanism differs between the different physical and logical managed objects with different management protocols. 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 [6].

The graphical data representation supported by the front-end component to visualise the management information has to be adapted to the respective application domain and remain user-friendly and simple.

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 Implementation

This section presents details of the prototype implementation of the INSMware concept.

3.1 SNMP

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 [8]. SNMP was originally developed by the IETF to manage TCP/IP networks and network elements such as routers and hubs [9]. The 'basic' SNMP (SNMP version 1) is now in widespread use and the de-facto industry standard. Virtually all major vendors of end-systems, workstations and network devices such as routers and switches offer SNMP support. The widespread acceptance of SNMP has resulted in adoptions such as the use of SNMP over OSI and other non-TCP/IP protocol suites. In addition, enhancements to the initial SNMP have been pursued in a number of directions (e.g. RMON, SNMP Version 2 and 3, ATM Management).

The SNMP entities are an SNMP Manager, an SNMP Agent and a Management Information Base (MIB). The SNMP Manager is typically network management software that implements the SNMP protocol. 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 and processes SNMP requests from the SNMP Manager and responses from the agent itself. The MIB is the database for management information that is stored in ASN.1 syntax format [10].

SNMP is a polling-based protocol. The manager sends a request for information to the agent (Get, Get Next operations) periodically and the agent responds to that request and the manager can modify and manipulate SNMP values in the agent (Set operation). Beside that, an agent can send an event to indicate an alarm or specified condition (Trap operation) to the manager to mark an important event, such as a critical network error.

There are two basic approaches to coexistence in a multi-lingual network (several SNMP versions), namely multi-lingual implementations and proxy implementations [11]. Proxy implementations provide a mechanism for translating between SNMP versions using a third party network element, and hence add complexity into the management-services due to the translation services required. The proposed INSMware framework is based on a multi-lingual SNMP implementation. The multi-lingual implementation of the management interface supports the different SNMP versions and enables the seamless integration of the (typically monolingual) 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. The Management Interface component connects the framework to the underlying networking technologies as illustrated in Figure 4.

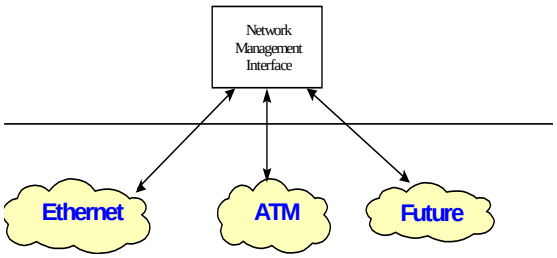

Figure 4: Network Management Interface

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. It became clear after the first stages of the project, that major parts of INSMware within the application domain integrated network and system management will be reusable with no or relative little changes to the components. Modifications to the system to integrate new services and reuse existing components are required on the interface (gateway) to the user (Front-End component for user interaction) and the interface to the managed network (Management Interface component for the communication with the managed objects).

To improve and optimise the reusability of the Management Interface component it was further abstracted – therefore, the functionality of the Management Interface component was split into several smaller components (see Figure 5).

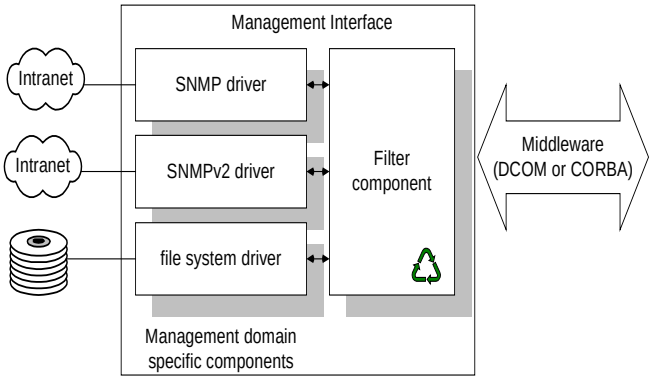

Figure 5: Architecture of Management Interface

The general functions for filtering of management information offer a high potential for reuse. This part analyses the information 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. Therefore (because of this high potential for reuse), the filtering was encapsulated and implemented into an individual component within the Management Interface. Hence the problem occurs of how to bring the data from the various kinds of managed objects into a syntax that can be understood and proceeded from the filtering component. The initialisation and de-initialisation of a connection with a managed object is individual for every kind of managed object with different management functionality (management protocols), e.g. the initialisation of a connection and the access to a SNMP managed object very much from the access to an ODBC database or a file system. The potential for reuse in this domain is relatively small and the component cannot integrate new access technologies through component configuration but through re-implementation and integration of the new access technologies. One exception are data sources that are similar concerning their access, e.g. different ODBC databases where relative little adjustments are required. To be able to integrate different access technologies within the management domain a concept was developed that is similar to the driver concept of operating systems. For each access technology (e.g. SNMP), 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 experienced, 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.

The introduction of an additional abstraction layer within the Management Interface leads to components with a very small level of granularity.

To minimise the associated disadvantages (e.g. loss of performance and stability) no middleware for distribution was introduced but the concept of Dynamic Link Libraries (DLL) was used. Due to the fact that it is an In-Process-Server, a complex binding process is not required and the communication can be performed very fast and without changing tasks.

3.2.2 SNMP-Interface

The analysis and design stage of the SNMP-Interface was based on the layered model of the driver-components of the Management Interface (Figure 1). Dynamic access mechanisms have been implemented to read the configuration from an ODBC database (MS Access is used as database for the prototype). The prototypical implementation of the Management Interface component is written in C++. The SNMP functions 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 [12].

Another advantage of the design that is focused on integration of new management services and managed objects is the easy adoptability of the INSMware management framework for new developments, e.g. for new versions of SNMP. The integration of the SNMP++ framework required only minor changes to the source code of the management interface. New SNMP functionality, such as SNMPv3, start to occur and can be implemented into the SNMP++ framework [13] and, therefore, with minor modifications into the INSMware framework. Figure 6 illustrates the layers of the Management Interface and the integration of the different SNMP frameworks within it.

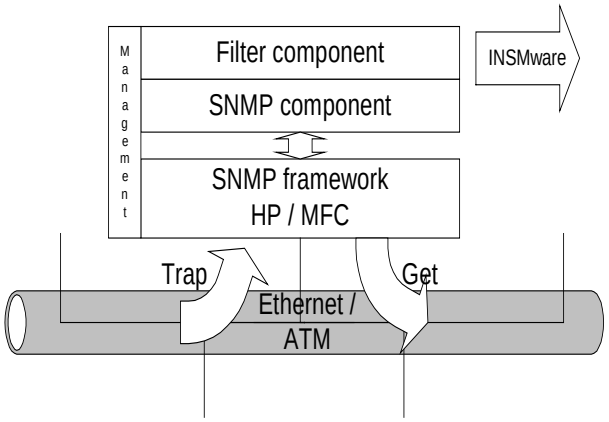

Figure 6: Management Interface layers

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 a 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.

Contrary to the passive management based on traps, where the managed object itself is initiating and sending management data, the Management Interface also enables active monitoring based on SNMP Get and GetNext operations. Here, the Management Interface reads management relevant data from the MIB of the managed objects to monitor and discover status changes or network conditions. The SNMP++ C++ libraries from Hewlett Packard support the different data types (Integer, Unsigned Integer32, Octet, OID, IP-Address, Counter32, Gauge32 and Timeticks) which are defined for SNMP. This MIB management information 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 of the related INSMware user on its GUI or forwards the notification to a manager via an external communication (e.g. 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, Operant) - is fully implemented via the GUI and saved in the management database (MS 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 Front-End

As mentioned before, 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 considered: A pure graphical approach, which uses overview maps to represent the network structure (as known from commercial Network and System Management platforms such as HP OpenView), and a more Microsoft Windows Explorer like view showing the network structure as tree. The first approach visualises the network structure very clearly and several network management tools (e.g. HP OpenView) use such an interface. Therefore, they are typically well known for experienced users of management tools, 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.

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.

The main user interface of INSMware is shown in Figure 7. The network structure is represented by tree at the left-hand side. At 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.

The view of the network is configurable for every single user. Every user can name the nodes in their preferred way and the part of the network, which is shown by the tree, is 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.

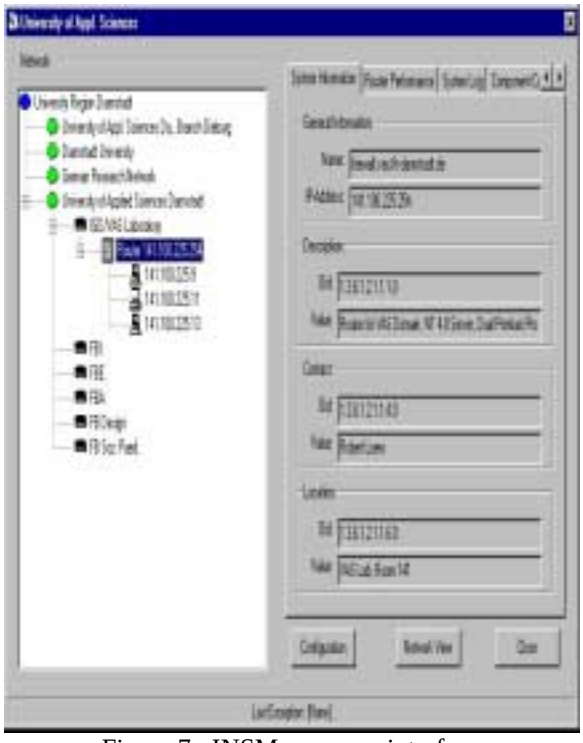

Figure 7 : INSMware user interface

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 at a first sight. 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. The ‘Component configuration’ tab will enable the user to reconfigure the managed object, e.g. to reset a network interface. All the configuration data – including the settings for the notification events – are stored in the system database.

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 – using HTML and ASP – in order to allow management and configuration from everywhere without installing client software.

4 Summary and Outlook

This paper has presented an integrated approach to Network and System Management and its prototypical implementation. The use of componentware technology provides flexibility and enables distribution of the management system. The usage of component technology allows the reuse of existing management, messaging and control functionality to integrate new management services.

Our future INSMware research aims at the following goals:

(i) development of extended and new forms of user interaction;

(ii) extension of INSMware to provide unified, integrative end-to-end management services for the different High-Speed Networking technologies and distributed applications.

Goal (i) includes the integration of Web-based management services and the integration of handheld devices, such as the Palm Pilot, to realise ubiquitous computing facilities. As no component models currently exist for such handheld systems, one task will be to develop appropriate integration mechanisms. 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. Another area for future research is a Web based GUI.

In relation to goal (ii), INSMware's current shaping focuses on the management of hardware components. However, this represents only a reduced view to real world information systems, since a correlation exists between managed hardware components (e.g. server systems) 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 integrative management facilities. To overcome architecture-inherent limitation of componentware, we integrate the Component Adapter approach to INSMware, which allows the integration of even semantically incompatible software components.

References

[1] Knahl, Martin; Hofmann, Holger D.; Phippen, Andy. A Distributed Component Framework for Integrated Network and System Management. Information Management and Computer Security, 7(5), pp. 254-260, MCB University Press, Bradford/UK, 1999.

[2] OMG. The Common Object Request Broker: Architecture and Specification, Revision 2.2. OMG Document 98-07-01, Object Management Group, Inc., 1998.

[3] Brown, N.; Kindel, C. Distributed Component Object Model Protocol - DCOM/1.0. Microsoft Corporation, Network Working Group, 1996.

[4] Knahl, M. et al. Integration of ATM management procedures into native integrated network and systems management architectures. Proceedings of the International Network Conference 1998, Plymouth, pp. 91-97. July 1998.

[5] Hofmann, Holger D., Stynes, J. Implementation Reuse and Inheritance in Distributed Component Systems. 22nd Annual International Computer Software and Applications Conference (COMPSAC'98), pp. 496-501, Vienna/Austria, IEEE Computer Society, 1998.

[6] Amrhein, Matthias. 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). BSc thesis, University of Applied Sciences Darmstadt/Germany, 1998.

[7] CEK-97, Cekro, Z. (1997). Using the SunNet Manager Platform to monitor European ATM activities. 2nd ATM Symposium, Brussels, November 21, 1997.

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

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

[10] ISO-87, ISO (1987). ISO 8824-1987: Information Processing Systems – Open Systems Interconnection – Specification of the Abstract Syntax Notation One (ASN.1). 1987.

[11] LEV-99, Levi, D. et al. (1999). 'Coexistence between SNMP Versions'. SNMPv3 Working Group. Internet Draft, 21 May 1999.

[12] Mellquist, Peter E., SNMP++: An Object-Oriented Approach to Developing Network Management Applications, Prentice Hall, 1997.

[13] Katz, Jochen. SNMPv3 Support for SNMP++. The Simple Times. Volume 7, Number 1. March 1999.

[14] Zitterbart, Martina (ed.). Kommunikation in Verteilten Systemen (Communication in distributed systems). Informatik aktuell, Springer, Berlin, Heidelberg, New York, 1997.