Network Management By Delegation
From Research Prototypes Towards Standards

J. Schönwälder <schoenw@cs.utwente.nl>

Abstract

This paper gives an overview over current activities to implement the Management by Delegation (MbD) model within the major standard network management frameworks. After a short introduction to the MbD model, a couple of design issues are identified and some research prototypes are described. The status of the work currently going on in the ISO and the IETF standardization processes is presented and related to the design issues.

I. Introduction

Operating todays fast growing multi-protocol networks is a challenging task. Besides the increase in size and complexity, networks have become a critical factor for the success of many organizations. Network management architectures have been developed and standardized over the last decade to provide an open environment which allows to control and optimize network operations in an efficient manner.

Todays network management systems are usually implementations of these architectures. Most of them follow the centralized management paradigm where a central management system controls a potentially large number of managed systems by manipulating management agents. These agents have predefined and fixed capabilities.

The centralized approach to solve the network management problem does not scale to handle todays large networks since management systems usually have to single step agents to correct network problems (micro-management) [33]. This approach becomes even more problematic if a management system uses the managed network itself to communicate with the management agents (in-band management) and the network enters a critical state. Distributed network management systems have been proposed in the last years to improve the scalability and the reliability of the management system itself. Different types of management applications that can benefit from such a distributed network management environment are discussed in [23].

Another problem of existing management standards is that they define a set of predefined management functions that are often difficult to adapt to site specific management needs. Even simple tasks like implementing a routing policy which changes the routing tables based on the time of the day and the current network load in order to take advantage of lower tariffs requires some non-trivial programming because the provided standard functions don't allow to express these kind of policies. This is mainly due to the fact that standard management functions only define a fixed set of parameters which are not powerful enough to express real world management policies.

The Management by Delegation (MbD) model [33] provides a solution to both problems because it allows to delegate management functions dynamically to cooperating managers or agents. Some attempts have been made in the past to build research prototypes based on the MbD model. This paper describes some of these prototypes and compares them with the current activities in the major standardization organizations.

This paper introduces the MbD model in section II followed by a short description of some research prototypes in section III. Section IV identifies some design issues that must be considered during the development of a MbD system. The current activities in the distributed management working group (DISMAN) of the Internet Engineering Task Force (IETF) are presented in section V and a proposal under consideration by the International Organization for Standardization (ISO) is described in section VI. Some conclusions are drawn in section VII.

II. Management By Delegation

The term Management by Delegation was coined by Yemini and Goldszmidt [33, 11, 10]. The fundamental idea behind this model is to add functionality to remote agents or managers by delegating management procedures at runtime. This approach is an application of mobile code technology, where executable code is moved over the network. Building distributed systems based on mobile code technology is fundamentally dif-

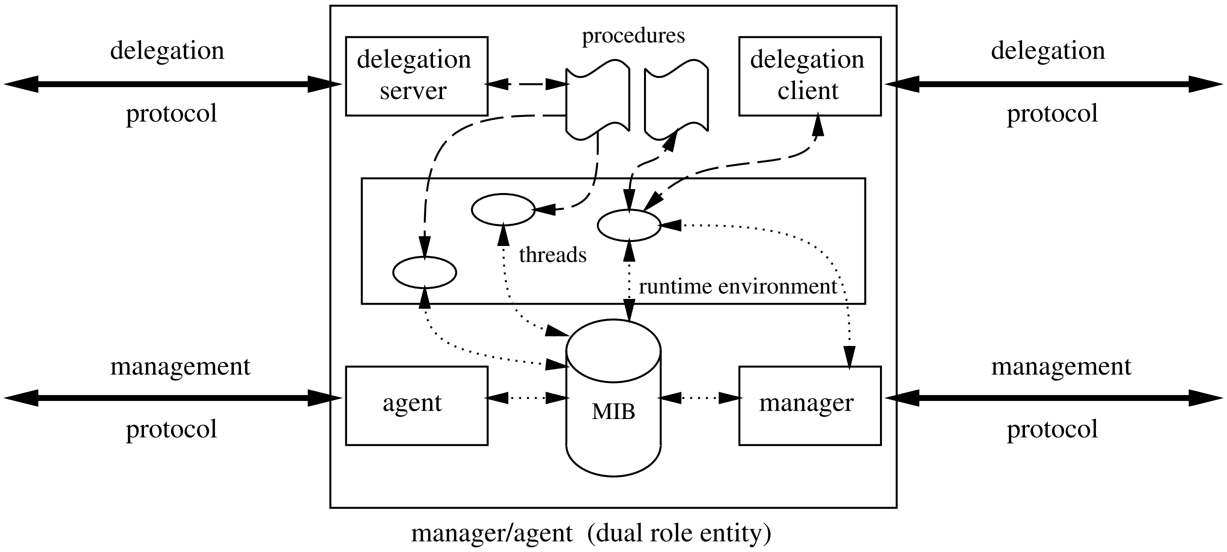

Figure 1: Structure of a symmetric MbD agent.

ferent from the well known client-server paradigm [2].

The MbD model uses code mobility to move management applications to the data they process while traditional approaches move the data to the application. Moving applications is more efficient if the amount of data that needs to be transferred is large or if the medium to transfer the data is a potential source of unreliability. The MbD approach reduces the total amount of network management traffic and allows to detect and repair problems even when the connection to the central management system is not operational (self healing). The MbD model uses the remote evaluation scheme [29] which allows to deploy new services and functionality at remote managers or agents on the fly which increases the flexibility of the whole management system.

Figure 1 shows the structure of a symmetric MbD agent [24, 25]. A delegation server implements services that can be used to create and delete management procedures and to start and control the execution of management procedures by creating so-called management threads. The delegation client allows to invoke services provided by remote delegation servers. The delegation protocol used between a delegation server and a delegation client can be a standard management protocol or a combination of management protocols and general purpose application layer protocols.

Management procedures are written in a language that can be executed by the runtime system. The selection of a management language has some important influences on the extensibility of a management system. Several different languages have been used in various research projects that range from high-level interpreted languages to low-level stack-oriented languages that require a compiler on the manager side in order to provide the required abstraction to write management applications in an acceptable manner [12, 26, 8, 28].

A management thread is executed in a runtime environment which provides all the services available to a management procedure. This usually includes access to the local management information base (MIB) which is exported by the agent module. An embedded manager module can be used to invoke management operations on remote agents. An important requirement for the runtime system is to enforce access control and to control resource usage to protect the system from malicious management procedures. Access control also requires an authentication mechanism to prove the identity of the manager that delegates management functions to a remote manager or agent.

III. Research Prototypes

This sections briefly discusses some research prototypes that are based on the MbD model. This description only focuses on the aspects of each prototype that are different from other approaches.

III.A. Columbia University 1995

The MbD prototype designed by Goldszmidt and Yemini at Columbia University is based on the concept of an elastic server which can be extended at runtime to provide new services [11, 10].

The elastic server loads compiled object code into a threaded runtime system. Local communication between management threads is based on shared-memory. Loading compiled object code allows to use different languages. Even a whole interpreter can be loaded into the elastic server if required.

The cost for this flexibility is limited security. There is no way for the runtime system to enforce resource control if the code loaded at runtime does not behave in a gentle way.

The delegation protocol used by this prototype is a special purpose protocol called the remote delegation protocol (RDP). The runtime system does not include interfaces to standard management protocols like SNMP or CMIP since these protocols can also be loaded at runtime via the RDP protocol.

III.B. SNMP Research 1993

Case and Levi from SNMP Research implemented a version of the MbD model which is entirely based on SNMP [7, 8]. They defined a new language called the SNMP script language (SSL) and a mid-level manager management information base (MLM-MIB) to transfer scripts and to start and control script execution. The SNMP script language provides statements to invoke SNMP operations on remote SNMP agents. The language represents every variable internally as a SNMP varbind which makes it possible to operate on local variables and values returned by SNMP operations in the same way. The language is compiled into an internal representation before a script is executed.

The MLM-MIB uses SNMP as a delegation protocol. A manager, who wants to delegate a procedure, writes the procedure into the MIB of the receiving agent. This is done line by line since SNMP does not allow to transfer values of arbitrary length. This model is called a "push" model since the server pushes the script to the receiving agent.

The result of a script execution is a SNMP varbind list which is stored in a result table of the agent. The manager has to read the result table once the script has finished its execution. This is called a "pull" model since the manager has to pull the results from the agent.

The work done by Case and Levi has inspired a couple of projects where scripting capabilities were integrated into special purpose agents. Kooijman integrated a Scheme [9] interpreter into a RMON probe in order to post-process and reduce data collected by the probe [21]. Similar work has been done by Williamson and Farrel [32] based on the language Perl [30].

III.C. TU Vienna 1995

A different approach to write management procedures has been proposed by Siegl and Trausmuth with the NMP stack language (NMPSL) [28, 27]. This language differs from many other languages used to describe management procedures because it is a low-level stack language. The language includes besides basic operations primitives to send SNMP requests. The advantage of using a low-level language is that the agent can execute management procedure efficiently. The compilation from a higher-level language can be performed by the management system that delegates a procedure.

The transfer of management procedures is implemented by writing the compiled management procedure into a MIB table via SNMP. Results are stored in a result table, similar to the prototype developed by SNMP Research.

III.D. TU Braunschweig 1996

The design of a MdB prototype which also provides security and mechanisms to control and coordinate MbD agents has been described by the author in [24, 25]. This system is based on an existing language and uses an event-driven runtime environment. It is based on the Internet suite of protocols and supports SNMP version 1 and version 2 as well as the user-based security model (USEC) [31]. The runtime system allows to use multiple interpreters to provide different security profiles.

Management procedures can be transferred by using a "pull" approach where the manager only sends the location of the management procedure to the agent. The agent fetches the procedure once it is needed from the location provided by the manager. This scheme has been implemented by using Uniform Resource Locators (URLs) [4] and the Hypertext Transfer Protocol (HTTP) [3].

IV. Design Issues

The prototypes described above allow to identify a set of design issues that must be taken into account when designing a management system based on the MbD model. Figure 2 summarizes these design issues by structuring the issues in four areas:

1. The delegation protocol can have different characteristics. It can either be based on an

Area	Possible Solutions		
Delegation Protocol	management protocol	⟷	special purpose protocol
	manager "push" (procedures)	⟷	agent "pull" (procedures)
	agent "push" (results)	⟷	manager "pull" (results)
Management Language	interpreted	⟷	compiled
	standard language	⟷	special purpose language
Procedures	MIB object arguments	⟷	arbitrary argument types
	MIB object results	⟷	arbitrary result types
	activation by events/scheduler	⟷	activation by manager
Runtime System	multiple manager	⟷	single manager
	multiple languages	⟷	single language
	multiple threads	⟷	event driven
	error logging	⟷	notifications
	status monitoring objects	⟷	notifications
	multiple security profiles	⟷	no security

Figure 2: Design issues for MbD environments.

existing (management) protocol or it can be a special purpose protocol. It can support a manager push model, an agent pull model or both. The transfer of results can be done in a similar way by using an agent push or a manager pull model.

2. The management language can be derived from an existing language or it can be a special purpose language. The language implementation can either be an interpreter or a compiler with a byte-code or native run-time system. The compiler may either reside on the manager or on the MbD system itself.

3. Management procedures usually take arguments and return results. Arguments and results can either be simple types (much like a command line interface for an operating system) or complex types that are bound to the MIB environment. The activation of management procedures can be scheduled by external and internal events or they are explicitly activated by the manager.

4. The runtime system may support multiple manager, multiple languages and multiple threads. Some implementations however make some restrictions in order to reduce the complexity of the system. The runtime system should provide an instrumentation so that a manager can control the runtime system itself (it can be seen as another managed resource).

   Management of the runtime system includes error and status logging mechanisms which might issue notifications if some conditions are met. Some implementations might choose to avoid notifications by requiring that the manager is responsible to control the status of running management procedures.

   A critical point of every MbD implementation is the security of the runtime system. Simple implementation might provide no security in the runtime system which means that the whole responsibility lies on the manager and the management protocol. Advanced implementations should support multiple security profiles within one execution environment.

V. IETF Standardization

The IETF has chartered a working group (DISMAN) to work on a short-term solution for scalable distributed management based on the Internet network management framework. This working group is responsible to deliver three documents. The first document is a framework for distributed management based on the Internet network management framework. The second document will be an expression and threshold monitoring MIB, which will be a revised and extended version of the Manager-to-Manager MIB [6], which has become historic when the SNMP version 2 standards moved to draft standard. The third document defines a script MIB which allows to transfer management scripts to remote locations and to start and control remote management threads. The script MIB is an approach to deploy the MbD model in the Internet network management framework.

The distributed management framework document [5] extends the Internet management framework with additional services that are needed in a distributed management environment:

- The *Known Systems* service defines a standard MIB to access and configure the set

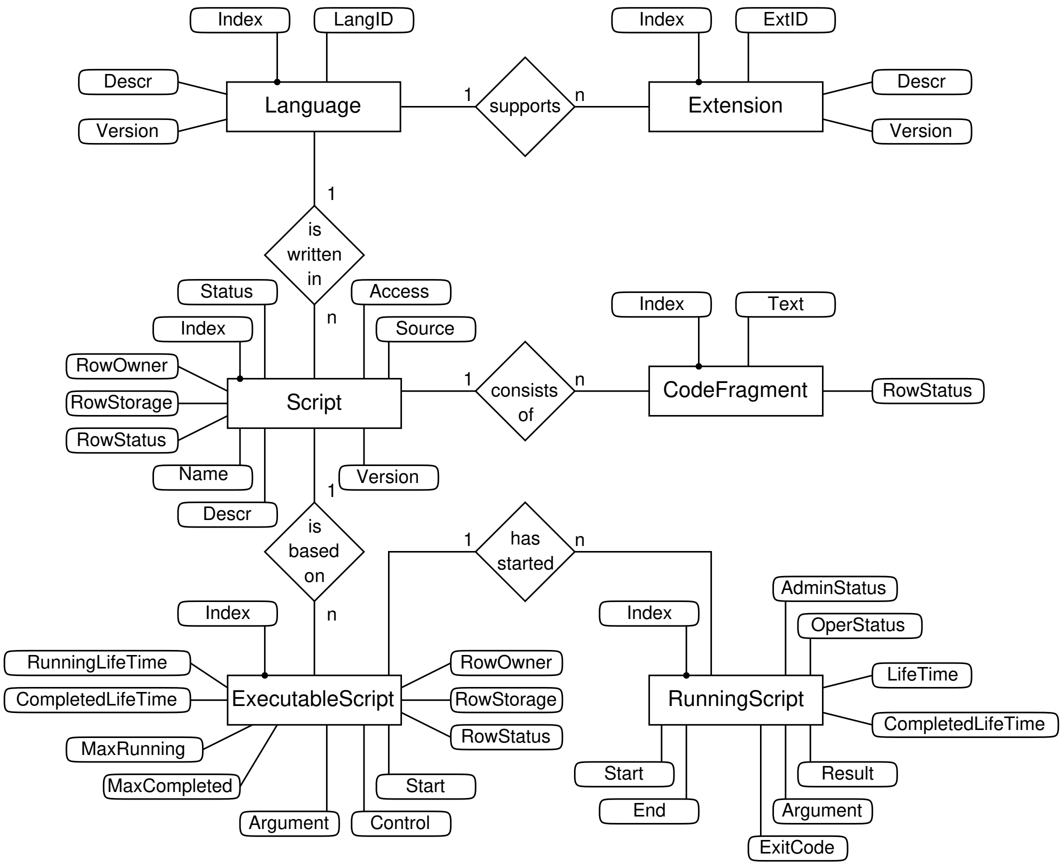

Figure 3: Structure of the script MIB.

of systems that are known to a distributed manager. This set can be populated by a discovery mechanism or a remote manager.

- The *Management Domains* service allows to define subsets of the set of known systems which can be used as targets for management operations. Management domains are defined by means of rules that define the properties of the domains.

- The *Scheduling* service allows the execution of distributed management applications to be controlled so that they run at a particular time, periodically, or based on the occurance of another event.

- The *Notifications and Logging* service allows to configure how notifications are processed and logged in the distributed manager.

- The *Delegation Control* service provides functions that limit the resource usage of a distributed management application that has had control delegated to it.

Not all services of the DISMAN framework MIB are specified at the time of this writing due to the complexity of the services listed above and some overlap with other working groups within the IETF that need to be solved.

Work on the script MIB document started by analyzing various proposals that were submitted to the working group. The following set of requirements were identified:

- The script MIB must be independent from concrete languages.

- The script MIB must provide mechanisms that help to avoid new management problems (e.g. script version management problems).

- The script MIB must provide SNMP MIB interfaces to all functions required to delegate management scripts. However, other protocols might be used in addition if they provide a significant improvement in terms of convenience or performance.

The first requirement raised a lot of discussions. Standardizing a management language has the advantage that such a common language will reduce interoperability problems. However, defining a standard management language is a complicated and thus time consuming process. People

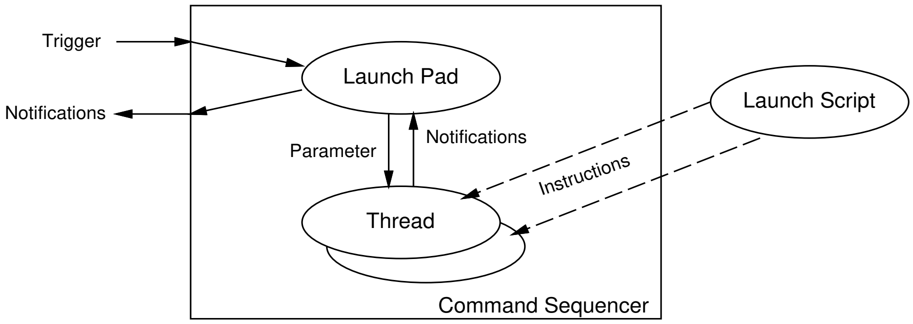

Figure 4: Structure of the command sequencer.

have many different opinions about the elements that make up an effective management language and it might be impossible to find rough consensus.

The script MIB [22] is composed out of six tables (figure 3). The Language table lists all the languages supported by a distributed manager. Language extensions (e.g. class libraries, additional statements or commands) that can be used to turn a general purpose language into a management language are listed in the Extension table.

Management scripts are represented as an entry in the Script table. The columns of this table allow to control whether a script is available, who owns the script, the storage type of the script (e.g. volatile or non-volatile) and how the script source is retrieved. The CodeFragment table can be used to push scripts to a distributed manager by issuing a sequence of SNMP set-requests. The pull method is supported by writing a URL to the Source column of an entry in the Script table.

The RunningScript table describes running scripts or scripts that have recently completed execution. A running script can be controlled by manipulating the administrative status. Entries in the RunningScript table are always created by means of an entry in the ExecutableScript table. An entry in the ExecutableScript table defines default parameters and controls that are applied when script execution is started.

VI. ISO Standardization

The International Organization for Standardization (ISO) is working together with the International Telecommunication Union (ITU) on a standard which allows to delegate management scripts in the ISO/ITU management framework. The requirements for this work are defined in [13]:

- User Requirements:
  - Allow the delegation of management activities;
  - Reduce the amount of communication that must occur between manager and agents;
  - Allow delegated systems to operate even when communication between manager and agent systems have been disrupted or are not possible;
  - Provide flexible control of management activities.
- Operational Requirements:
  - Pre-scheduled or delayed execution of a systems management operation;
  - Capabilities for modifying the request for pre-scheduled or delayed execution;
  - Capabilities for initiating, suspending, resuming and terminating systems management operations based on time management actions or the occurrence of events;
  - Capabilities for reporting and recording the outcome of pre-scheduled or delayed execution;
  - The ability to send notifications when state changes occur.

The solution defined in [13] is based on a command sequencer object, a thread object, a launch script object and a launch pad object (figures 4 and 5). The command sequencer is a management support object representing a resource which acts in a manager role as an invoker of operations determined by its launch scripts. It also acts as a notification destination. It contains one or more launch pad objects and acts as a service provider for these launch pad objects.

The thread managed object models command execution and performs commands according to a management script. Each thread object is dedicated to a launch pad object. A thread object

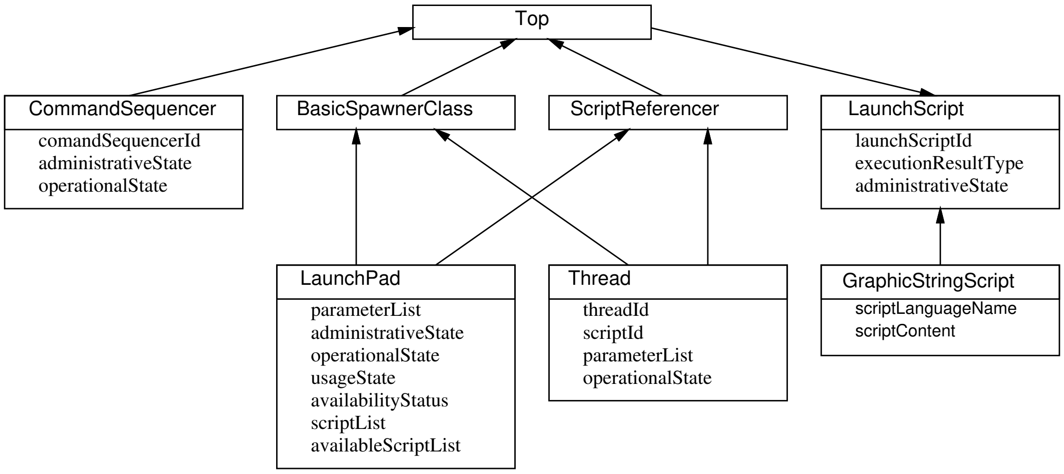

Figure 5: Inheritance structure and attributes of the command sequencer classes.

returns the results of a script execution via a notification.

A launch script object represents a management script, which is a sequence of instructions in a specialized scripting language. An example is the Systems Management Scripting Language (SMSL) which is defined in the annex of [13]. SMSL has a syntax similar to C [20] and converts strings to data types very similar to AWK [1]. Some builtin commands allow to access and modify ISO managed objects which are defined according to the Guidelines for the Definition of Managed Objects (GDMO) [16]. However, SMLS, as defined in [13], has some serious shortcomings. For example, it provides no primitives to invoke management operations on remote managed objects. The specification forbids recursive function calls, limits the maximum number of parameters in a function call to 10 and the maximum number of local variables to 20. Functions can not be nested and arguments are always passed by value.

A launch pad object starts the execution of a launch script object on receiving a trigger. A trigger can be an external event or the result of a scheduling function. A script is executed by a launch pad by means of one or more thread objects. The launch pad object has attributes which list the available launch scripts and predefined parameters.

The command sequencer provides services to create command sequencer, launch pad and launch script instances, to delete launch pad and launch script instances, and to modify and retrieve command sequencer, launch pad and launch script attributes. Additional services support result notifications and triggers to activate a launch pad. Script results are send to the manager by using a notification, according to the "push" model. These communication services are based on the Common Management Information Service (CMIS) [15] and thus on the CMIP [14] management protocol.

The ISO approach to implement the MbD model in the OSI management framework makes use of existing systems management functions (object management function [18], state management function [19], alarm reporting function [17], event report management function [17]). The OSI specification does not specify security mechanisms for the proposed script language so that the command sequencer has to trust the manager who downloads scripts.

VII. Conclusions

The Management by Delegation model has been first published in 1991 [33]. Five years later, two major network management standardization organizations have started to work on documents that can bring this technology into wide-spread use.

This paper started by identifying some design issues that can be derived from research prototypes in this area. Afterwards, the current activities within the IETF and the ISO to support the management by delegation paradigm as part of their management frameworks have been presented.

Both approaches have many similarities. Both designs support multiple languages, both approaches separate scheduling functions from script execution and both approaches allow to execute a script several times simultaneously. There are however some fundamental differences. The ISO approach uses notifications to "push" results to the manager while the IETF approach requires that the manager "pulls" the results by reading the RunningScript table. The IETF working group decided to use the "pull" method in order to support off-line operations, where result histories are collected at the distributed manager.

The ISO document defines a concrete script language (SMSL) in an annex which gives a clear hint to implementors which language to choose. The IETF does not define such a language and it is up to the market to decide which language will be accepted. The IETF and the ISO approaches also differ in the way arguments are passed to a script. The IETF proposal uses an opaque sequence of bytes as a script argument and leave the interpretation to the script, while the ISO document passes a list of MIB objects. The same distinction is true for the result of a script execution.

Additional differences are due to the fact that the ISO framework already defines a rich set of systems management functions that can be reused. The IETF management framework still lacks generic event forwarding or scheduling functions. Some of these missing functions are currently defined as part of the DISMAN activities. However, design decisions must be taken carefully in order to keep the IETF distributed management framework implementable while making it robust enough to scale well.

References

[1] A. V. Aho, B. W. Kernighan, and P. J. Weinberger. *The AWK Programming Language.* Addison Wesley, 1988.

[2] M. Baldi, S. Gai, and G. P. Picco. Exploiting code mobility in decentralized and flexible network management. In *Proc. of the First International Conference on Mobile Agents 97 (MA'97)*, Berlin, April 1997.

[3] T. Berners-Lee, R. Fielding, and H. Frystyk. Hypertext Transfer Protocol – HTTP/1.0. RFC 1945, MIT/LCS, UC Irvine, May 1996.

[4] T. Berners-Lee, L. Masinter, and M. McCahill. Uniform Resource Locators (URL). RFC 1738, CERN, Xerox Corporation, University of Minnesota, December 1994.

[5] A. Bierman, M. Greene, B. Stewart, and S. Waldbusser. Distributed Management Framework. Internet Draft <draft-ietf-disman-framework-00.txt>, Cisco Systems, Ascom Nexion, Cisco Systems, International Network Services, December 1996.

[6] J. Case, K. McCloghrie, M. Rose, and S. Waldbusser. Manager-to-Manager Management Information Base. RFC 1451, SNMP Research Inc., Hughes LAN Systems, Dover Beach Consulting Inc., Carnegie Mellon University, April 1993.

[7] J. D. Case and D. B. Levi. SNMP Mid-Level-Manager MIB. Internet Draft 00, SNMP Research, Inc., October 1993.

[8] J. D. Case and D. B. Levi. SNMP Script Language. Internet Draft 00, SNMP Research, Inc., October 1993.

[9] W. Clinger and J. Rees. Revised^4 report on the algorithmic language scheme. Technical Report AI Memo 848b, MIT, November 1991.

[10] G. Goldszmidt. *Distributed Management by Delegation.* PhD thesis, Columbia University, New York, December 1995.

[11] G. Goldszmidt and Y. Yemini. Distributed Management by Delegation. In *Proc. International Conference on Distributed Computing*, June 1995.

[12] D. Holden. Predictive Languages for Management. In *Proc. International Symposium on Integrated Network Management*, pages 585–596, May 1989.

[13] ISO. Information technology — Open Systems Interconnection — Systems Management — Command Sequencer. Draft International Standard ISO 10164-21, ISO.

[14] ISO. Information technology – Open System Interconnection – Common management information protocol specification. International Standard ISO 9596, ISO, 1990.

[15] ISO. Information technology – Open System Interconnection – Common management information service definition. International Standard ISO 9595, ISO, 1991.

[16] ISO. Information technology — Open Systems Interconnection — Structure of Management Information — Part 4: Guidelines for the Definition of Managed Objects. International Standard ISO 10165-4, ISO, 1992.

[17] ISO. Information technology — Open Systems Interconnection — Systems Management — Alarm Reporting Function. International Standard ISO 10164-4, ISO, 1992.

[18] ISO. Information technology — Open Systems Interconnection — Systems Management — Object Management Function. International Standard ISO 10164-1, ISO, 1993.

[19] ISO. Information technology — Open Systems Interconnection — Systems Management — State Management Function. International Standard ISO 10164-2, ISO, 1993.

[20] B. W. Kernighan and D. M. Ritchie. *The C Programming Language.* Prentice Hall, 2 edition, 1988.

[21] R. Kooijman. Divide and conquer in network management using event-driven network area agents. Technical Report, TU Delft, March 1995.

[22] D. B. Levi and J. Schönwälder. Definitions of Managed Objects for the Delegation of Management Scripts. Internet Draft <draft-ietf-disman-script-mib-01.txt>, SNMP Research, University of Twente, March 1997.

[23] K. Meyer, M. Erlinger, J. Betser, C. Sunshine, G. Goldszmidt, and Y. Yemini. Decentralizing Control and Intelligence in Network Management. In *Proc. International Symposium on Integrated Network Management*, May 1995.

[24] J. Schönwälder. *Netzwerkmanagement mit programmierbaren, kooperierenden Agenten.* PhD thesis, Institut für Betriebssysteme und Rechnerverbund, TU Braunschweig, 1996.

[25] J. Schönwälder. Using Multicast-SNMP to Coordinate Distributed Management Agents. In *Proc. IEEE Workshop on Systems Management*, Toronto, July 1996.

[26] J. Schönwälder and H. Langendörfer. Tcl Extensions for Network Management Applications. In *Proc. 3rd Tcl/Tk Workshop*, pages 279–288, Toronto (Canada), July 1995.

[27] M. R. Siegl. *Design and Realisation of a Mid-Level Management System.* PhD thesis, TU Wien, Vienna, November 1996.

[28] M. R. Siegl and G. Trausmuth. Hierarchical Network Management – A Concept and its Prototype in SNMPv2. In *Proc. 6th Joint European Networking Conference*, Tel Aviv (Israel), May 1995.

[29] J. W. Stamos and D. K. Gifford. Implementing Remote Evaluation. *IEEE Transactions on Software Engineering*, 16(7):710–722, July 1990.

[30] L. Wall and R. Schwartz. *Programming Perl.* O'Reilly, 1991.

[31] G. Waters. User-based Security Model for SNMPv2. RFC 1910, Bell-Northern Research Ltd., February 1996.

[32] B. Williamson and C. Farrell. Distributed Management using the Remote Monitoring Management Information Base (RMON MIB) and extensible agents. Technical Report, Curtin University of Technology, September 1996.

[33] Y. Yemini, G. Goldszmidt, and S. Yemini. Network Management by Delegation. In *Proc. International Symposium on Integrated Network Management*, pages 95–107, April 1991.

Author Information

Jürgen Schönwälder received his Ph.D. in computer science in 1996 from the Technical University of Braunschweig, Germany. He joined the network management group at the University of Twente in spring 1996.

He is an active member of the IETF and has co-authored drafts for the distributed management working group (DISMAN) and the application management working group (APPLMIB). He is also the designer and main author of the 'Scotty' network management package.