Proceedings of the *IEEE/IFIP Network Operations and Management Symposium (NOMS'98)*,
New Orleans, USA, February 1998

Tool-based Implementation of a Q-Adapter Function for the seamless Integration of SNMP-managed Devices in TMN

Alexander Keller

MNM
TEAM

Munich Network Management Team

Faculty of Computer Science, Munich University of Technology

E-Mail: keller@informatik.uni-muenchen.de

Abstract and Structure of the Presentation

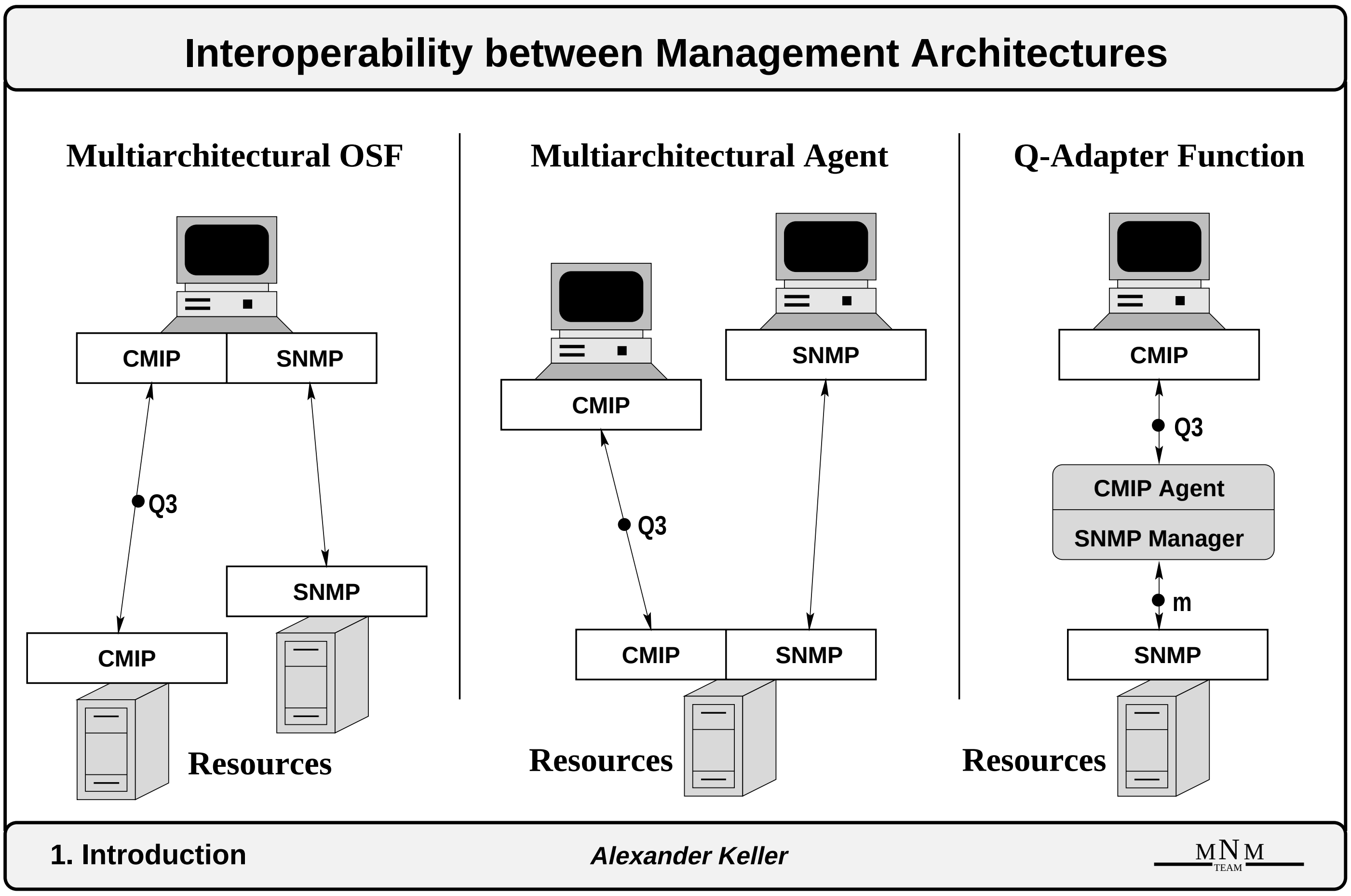
During the recent years, integrated management has become more complicated due to the standardization of several competing management architectures: This adds additional heterogeneity to the already existing amount of heterogeneity of today's distributed systems. The most widespread management architectures are the OSI/TMN and Internet (SNMP-based) management frameworks. With the lack of interoperability between these different management architectures, there is a strong demand for solutions that provide smooth transition paths between them.

This presentation describes the results of a research project dealing with the design and implementation of a **Q Adapter Function (QAF)**, often called *Management Gateway* for the integration of SNMP-managed resources in a TMN environment [16] . QAFs are a mechanism for bringing management information from the pre-TMN installed base of systems into a TMN environment [3]. The reason for choosing a TMN-compliant management platform as the core of our distributed management system is the fact that the OSI/TMN management architecture yields the largest set of management functionality by far. On the other hand, since SNMP agents are in widespread use, an integrated management solution needs to take into account the Internet management framework. Furthermore, if the Internet managed objects appear from the managing system's perspective similar to the other OSI/TMN managed objects, the complete set of OSI/TMN management functionality can be applied to them.

This paper is organized as follows: Section 1 compares three possible ways for bridging the gaps between different management architectures and gives the reasons why we decided to use the QAF approach. Section 2 describes the underlying concepts and the principal steps for designing a QAF for SNMP; it also presents already existing algorithms and methods which have been used for the design of our prototype. Obviously, the development tools have an impact on the structure of the acquired solution; section 3 therefore describes the *IBM TMN Products*, a TMN development toolset that we used for the QAF implementation, and explains the architecture of the prototype. Section 4 discusses several implementation issues and features of the QAF, namely how the *scoping* and *filtering* capabilities are realized and how the capabilities of the QAF can be increased by introducing SNMP MIBs describing new resources. Finally, section 5 concludes the paper and presents issues for further research.

1 Introduction and Motivation

1.1 Approaches for interoperability between management architectures



There are basically three different strategies for achieving a seamless interaction of components located in different architectural domains (see also [15]):

The first approach consists in placing the burden of integrating the different architectures on the managing system. Such a **multiarchitectural Operations System Function (OSF)** supports a set of management protocols which are implemented onto the OSF's communication stack. A conversion between different management protocols is therefore not necessary. The transformation of the management information descriptions is often left to the management applications and not handled by the OSF infrastructure. The experiences with XOM/XMP show that this kind of integration is unsatisfactory: Multiarchitectural OSFs distinguish at the top of the topology hierarchy between the different supported architectures. The application of one architecture's specific features (such as the OSI *scoping* and *filtering* facilities) to managed objects in another architecture is impossible.

An alternative is the integration at the resource level, i.e. the managed systems support more than one management protocol; they are equipped with **multiarchitectural agents**. This is usually unfeasible for the following reasons: SNMP agents, for example, are often used to perform monitoring of simple network devices. They should not consume a large amount of resources and are usually built into the firmware of the device; the implications are that these agents can neither be enhanced to support another management protocol nor should they introduce additional complexity. For an in-depth discussion of this subject, the reader is referred to ([18], [6]).

The third solution is the **Q-adapter function**. It is then possible to manage services, systems and networks in different management architectures from a single point of control. It is even possible to apply the power of the OSI management architecture to any resource in the different architectural domains, as shall be described in section 4.1. In management architectures having no notion of a management functional model such as the Internet framework, the application of management functionality "borrowed" from other architectures is particularly useful. For these reasons, we decided to follow this approach for the design and implementation of our interoperability solution.

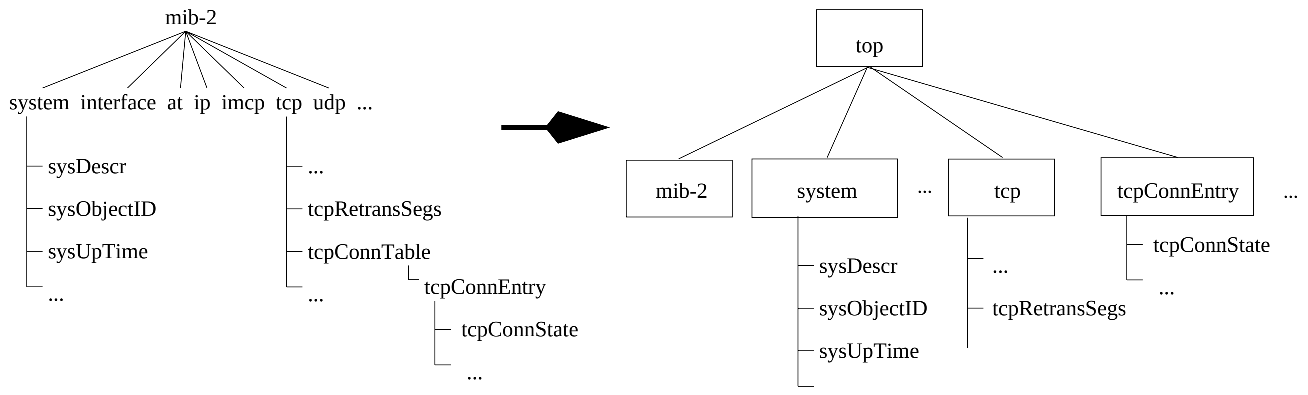
2 Achieving Interoperability

2.1 Transformation Step 1: Mapping the Information Models

Information Model Mapping: The IIMC Algorithm

Transformation of Internet-SMI Elements into new GDMO Class Definitions:

1. Internet-SMI Groups	→	MOCs
2. Internet-SMI Table Rows	→	MOCs
3. Other Internet SMI-Elements	→	MOC Attributes



2. Interoperability *Alexander Keller* MNM TEAM

From the three approaches for achieving interoperability between management architectures, the Q-adapter function approach seems to deliver the most workable solution as the mapping between the different architectures is done at a limited number of points, namely m-reference points at the boundaries of the architectural domains. Furthermore, neither managing systems nor managed systems need to be modified. The price of this flexibility is the complexity of the QAF as shall be described below.

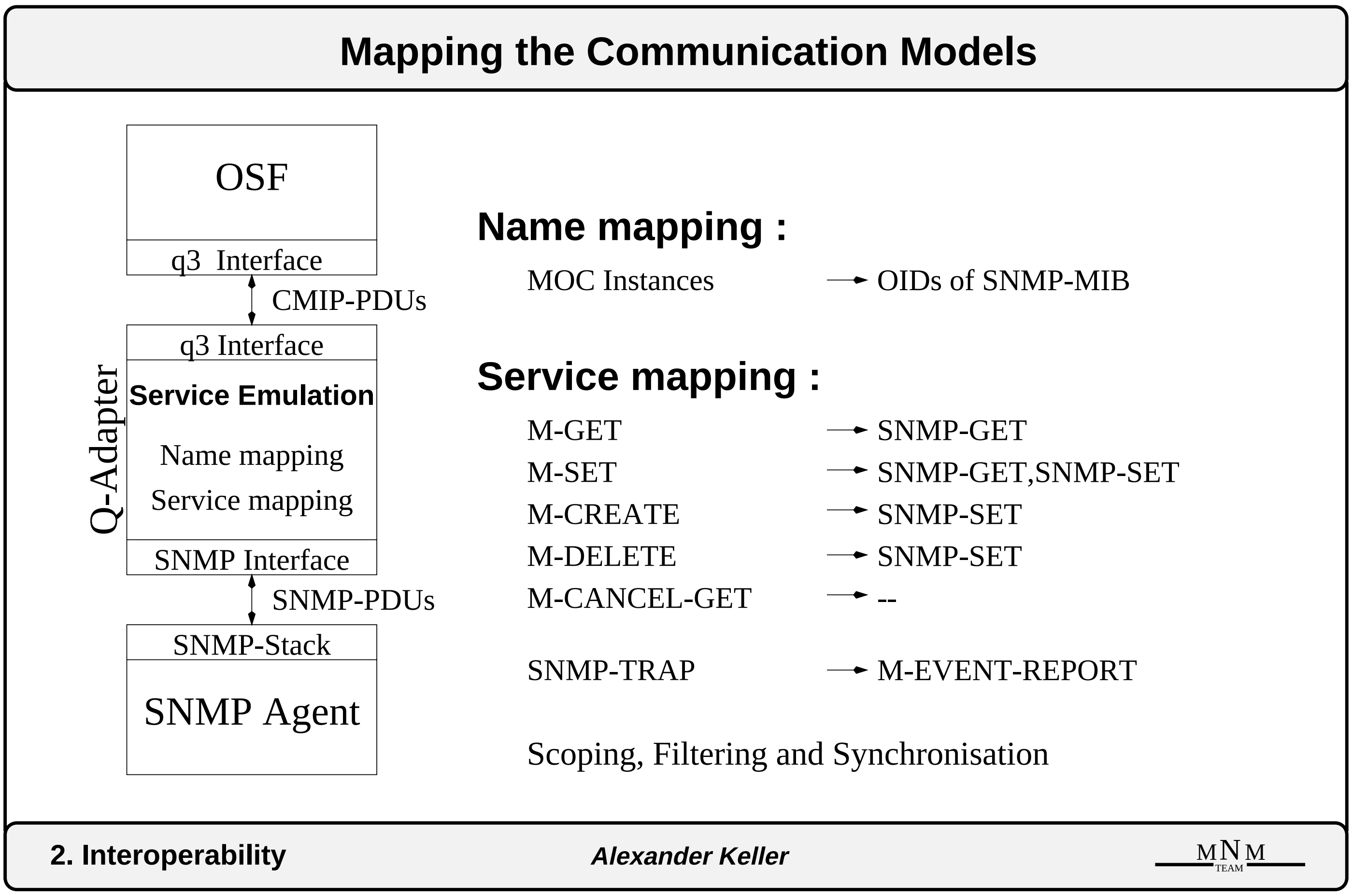
The specification of algorithms used for the implementation of QAFs is the goal of an initiative from X/Open and the Network Management Forum called *ISO-Internet Management Coexistence (IIMC)*. It devides the transformation process in two phases: The first step permits the representation of managed objects defined in Internet SMI (Structure of Management Information) in the OSI/TMN information model [12]. The algorithm is described in [22]; an implementation of this transformation algorithm has been done by the Telecommunications Laboratory of the *Technical Research Centre of Finland* and is available from the Network Management Forum FTP server. The implementation is documented in [26].

The mapping of the Internet-SMI constructs to TMN managed objects is straightforward: Attribute groups and table rows become MOCs, scalar variables are translated into attributes of the appropriate MOC.

The IIMC algorithm handles not only the naming and registration of managed object classes but also the registration of their name bindings that help to build the containment hierarchy. The distinguished names for MOCs, name bindings and naming attributes are built by appending the SNMP-OIDs to a predefined prefix. On the other hand, the uniqueness of the names cannot be guaranteed: The MIB-II [20] contains a group named "system" while the ISO 10165-2 "system" class is defined as the root of the OSI containment hierarchy. While translating MIB-II, "system" has to be renamed to e.g. "internetSystem" (see also section 4.2).

As a result of the information model mapping process, the former SNMP MIB has been transformed into GDMO [13] and ASN.1 managed object descriptions.

2.2 Transformation Step 2: Mapping the Communication Models



Mapping the Communication Models

Name mapping :

MOC Instances	→ OIDs of SNMP-MIB

Service mapping :

M-GET	→ SNMP-GET
M-SET	→ SNMP-GET,SNMP-SET
M-CREATE	→ SNMP-SET
M-DELETE	→ SNMP-SET
M-CANCEL-GET	→ --
SNMP-TRAP	→ M-EVENT-REPORT

Scoping, Filtering and Synchronisation

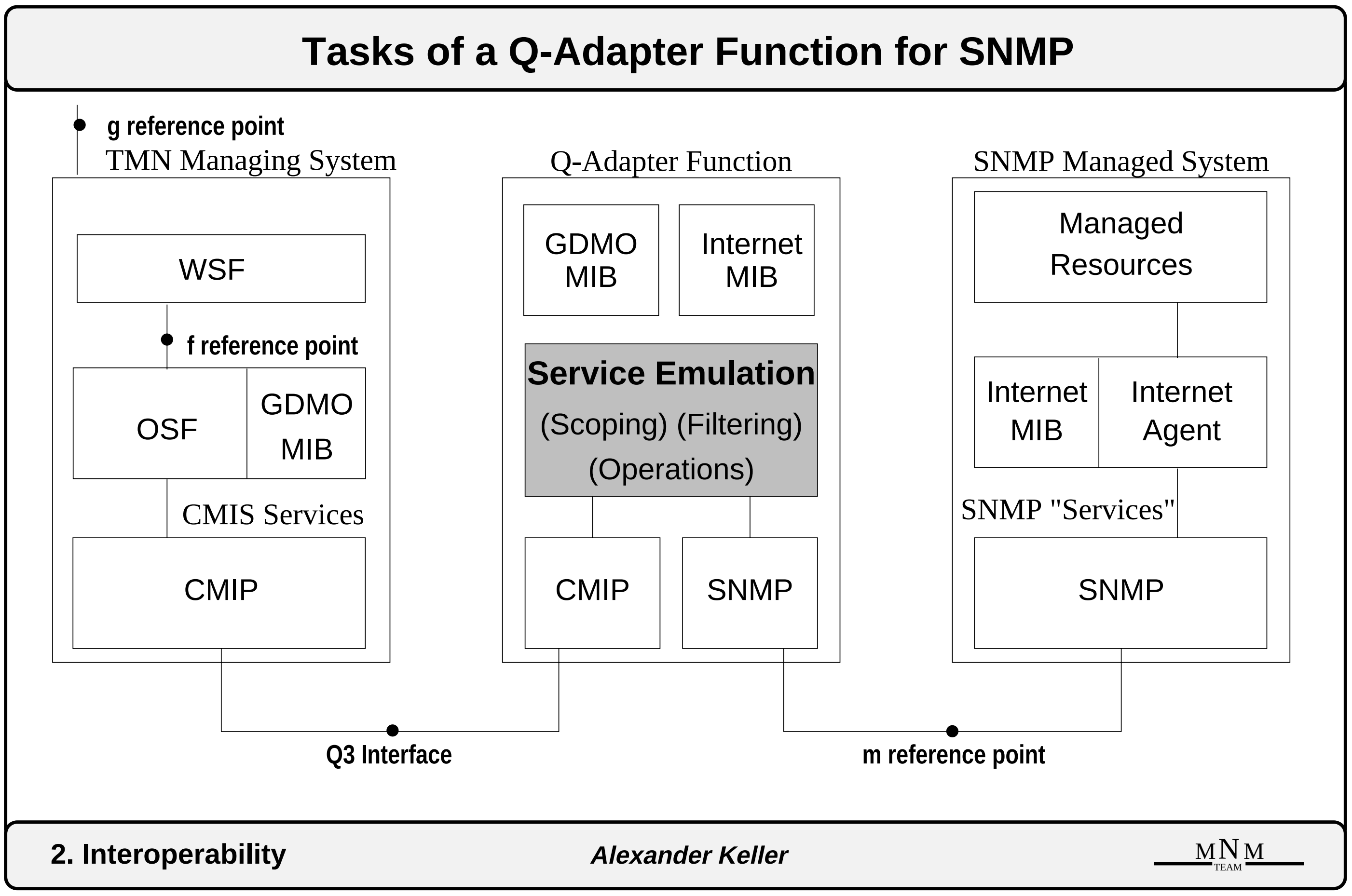
The second step provides a mechanism for the exchange of management data between managed objects in the different domains i.e. the mapping of CMIS requests to SNMP PDUs. The concepts behind this mechanism are described in detail in [23] and will just be sketched out for the sake of brevity. The main aspects of the communication model transformation are the Name Mapping and the Service Mapping.

Name mapping describes the task of computing the corresponding SNMP Object Identifier (OID) from a given name specified in a CMIS request and depending on the containment hierarchy. This is necessary because the OSF has obviously no notion of SNMP OIDs.

Service mapping implies the transformation of CMIS services into adequate SNMP PDUs. First, the managed objects to which an operation applies have to be identified, i.e. the set of instances lying in the scope of the request have to be determined. These instances are then checked whether their attributes meet the filter criteria. The obtained set of MO instances is then subject to the operations according to the figure above.
Incoming SNMP traps are received by the QAF and transformed into CMIP notifications issued by the corresponding object instance. CMIS services like M-CREATE and M-DELETE can be mapped in the case of tables to SNMP-GET requests: This is done by modifying the corresponding *Row Status* variables according to the conventions of the SNMP framework. The M-CANCEL-get service is the only CMIS-operation that cannot be mapped to an appropriate SNMP command.

By now, it should have become clear that a QAF does not only act as a converter of protocol data units; it is particularly necessary to perform a mapping between the heterogeneous management information models. This must be done for achieving coexistence and cooperation between management architectures while the agents remain simple. Q-adapter functions are dual-role entities: From the perspective of a managing system, they act as managed systems; from an agent's point of view, they appear as managing systems.

2.3 Putting it all together: Properties of Q-Adapter Functions; IIMC Proxy Management Model



Q-adapter functions allowing the management of SNMP agents from a TMN-compliant managing system must fulfill the following requirements:

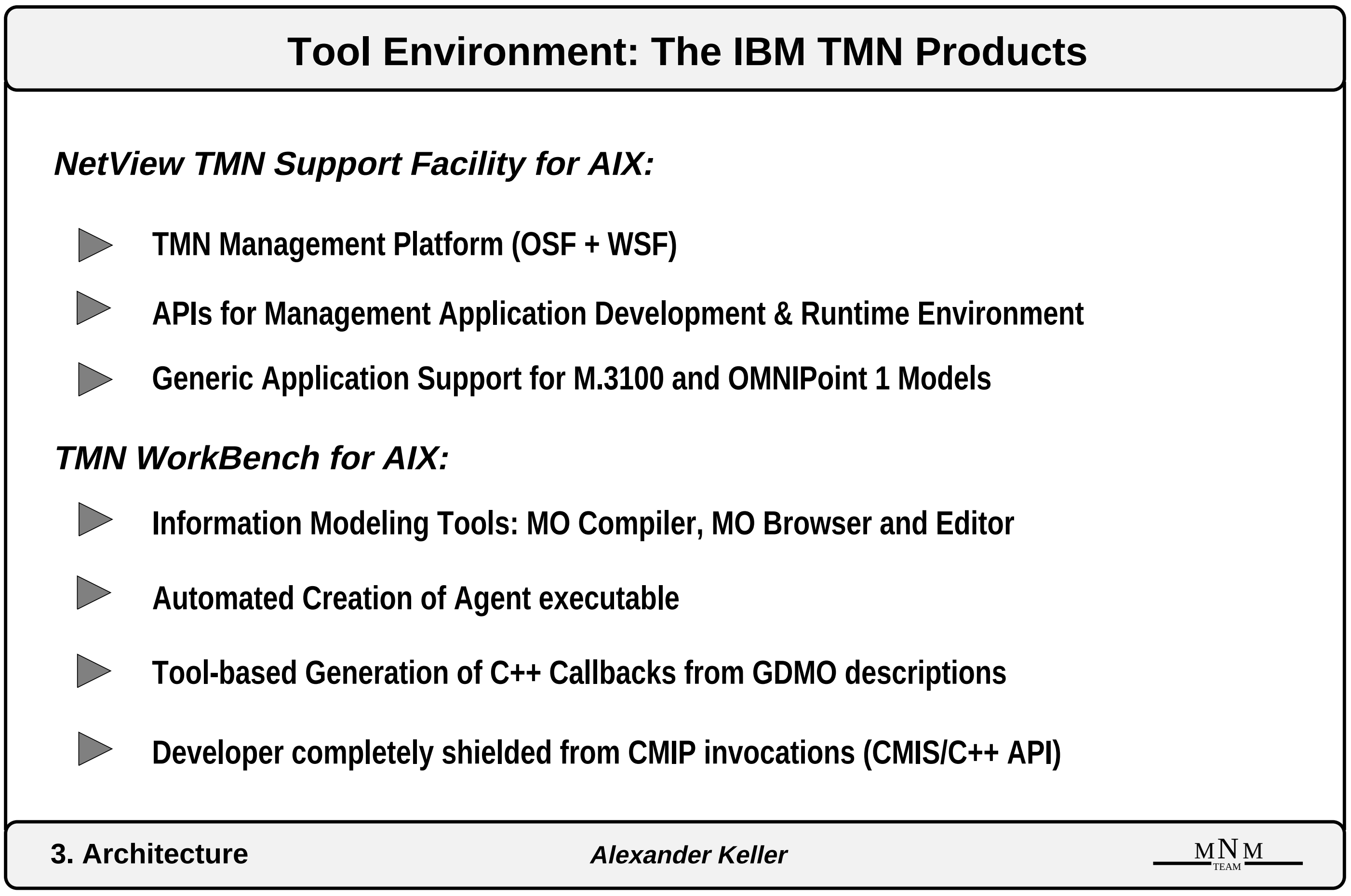
1. The agent's SNMP-MIBs have to be represented in the manager's information model (OSI SMI).
2. No modifications are to be applied to managing or to managed systems, i.e. a *completely transparent* transition between the architectures for all entities must be achieved. The Q-adapter function therefore has the duty of providing TMN-compliant proxy objects for any SNMP agent in the domain surveyed by the QAF.
3. Operations issued by the OSF (Create, Delete, Get, Set) have to be interpreted by the QAF and forwarded to the corresponding agent. On the other hand, SNMP traps issued by the agents have to be transformed into CMIP notifications and then forwarded to the OSF.

One may wonder why the mapping of only two models (namely the information and communication models) is sufficient for our purposes and why it is not necessary to provide a transition between the organizational and functional models also defined in [14]. The reason is that the former model is roughly identical for both architectures; the latter, in contrast, defines a broad range of management functionality for the OSI/TMN domain and has no counterpart in SNMP management.

The goal is to deliver the strengths of the OSI/TMN architecture to the Internet management architecture in order to obtain a comprehensive integrated management solution. This implies that the Internet architecture (having no notion of management functionality) can be used with management services already defined in the OSI/TMN management framework. A typical example for this is the enhancement of the Internet management architecture with management functionality like threshold monitoring or event processing defined by the OSI Systems Management Functions.

3 Tools and Architecture

3.1 Tool Environment



Tool Environment: The IBM TMN Products

NetView TMN Support Facility for AIX:

- TMN Management Platform (OSF + WSF)
- APIs for Management Application Development & Runtime Environment
- Generic Application Support for M.3100 and OMNIPoint 1 Models

TMN WorkBench for AIX:

- Information Modeling Tools: MO Compiler, MO Browser and Editor
- Automated Creation of Agent executable
- Tool-based Generation of C++ Callbacks from GDMO descriptions
- Developer completely shielded from CMIP invocations (CMIS/C++ API)

3. Architecture *Alexander Keller* MNM TEAM

For implementing the QAF according to the principles outlined in section 2, we decided to use the *IBM TMN Products* ([8], [4]) as the development environment. They consist of two main components:

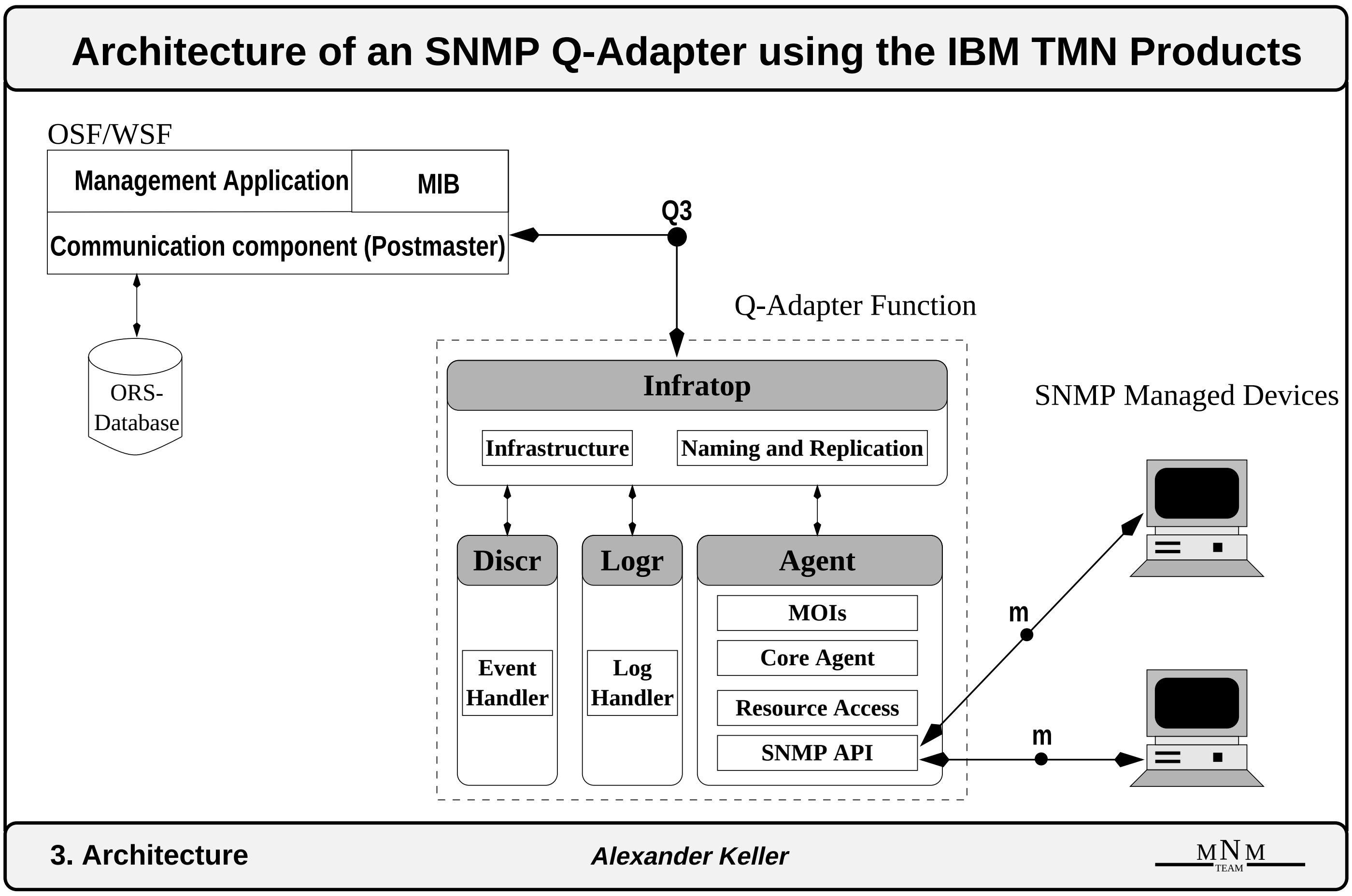
The *NetView TMN Support Facility* [7] encompasses a TMN management platform (OSF and WSF) and APIs for developing TMN applications. The product supports both M.3100 [1] and OMNIPoint [5] object models and provides all the necessary features for managing OSI/TMN compliant management agents.

The *TMN WorkBench for AIX* [9] provides the whole development environment. It consists of information modeling tools like MOC browsers and editors by which new MOCs can be derived from existing ones by drag-and-drop operations. For this purpose, a large number of standardized MOC catalogues is shipped in machine-readable form. TMN compliant agents are built by compiling the GDMO and ASN.1 definitions of the required MOCs into C++ class headers. The developer then only has the duty of implementing the behaviour of the managed objects by filling the provided *callbacks* with program code.

The main advantage of the IBM TMN development environment lies in a new C++/CMIP-API ([21], [2]) which is being standardized by the NM Forum: The developer is completely shielded from the complexities of the management protocol and is able to automatically generate C++ interfaces from GDMO managed object descriptions. It is even possible to automatically generate an agent executable directly from GDMO and ASN.1 documents which already contains the standardized generic behaviour descriptions. The developer has therefore only to cope with the implementation of the specific behaviour. In [19] a similar approach to a QAF for SNMP based on the OSIMIS management platform [24] is described.

The following sections will explain the influence of the IBM TMN development environment on the design and implementation of our solution. Our experiences with the IBM TMN products and off-the-shelf CORBA toolkits have shown that the development process of TMN agents is very similar to the way CORBA management agents are implemented.

3.2 The Architecture of the Q-Adapter Function



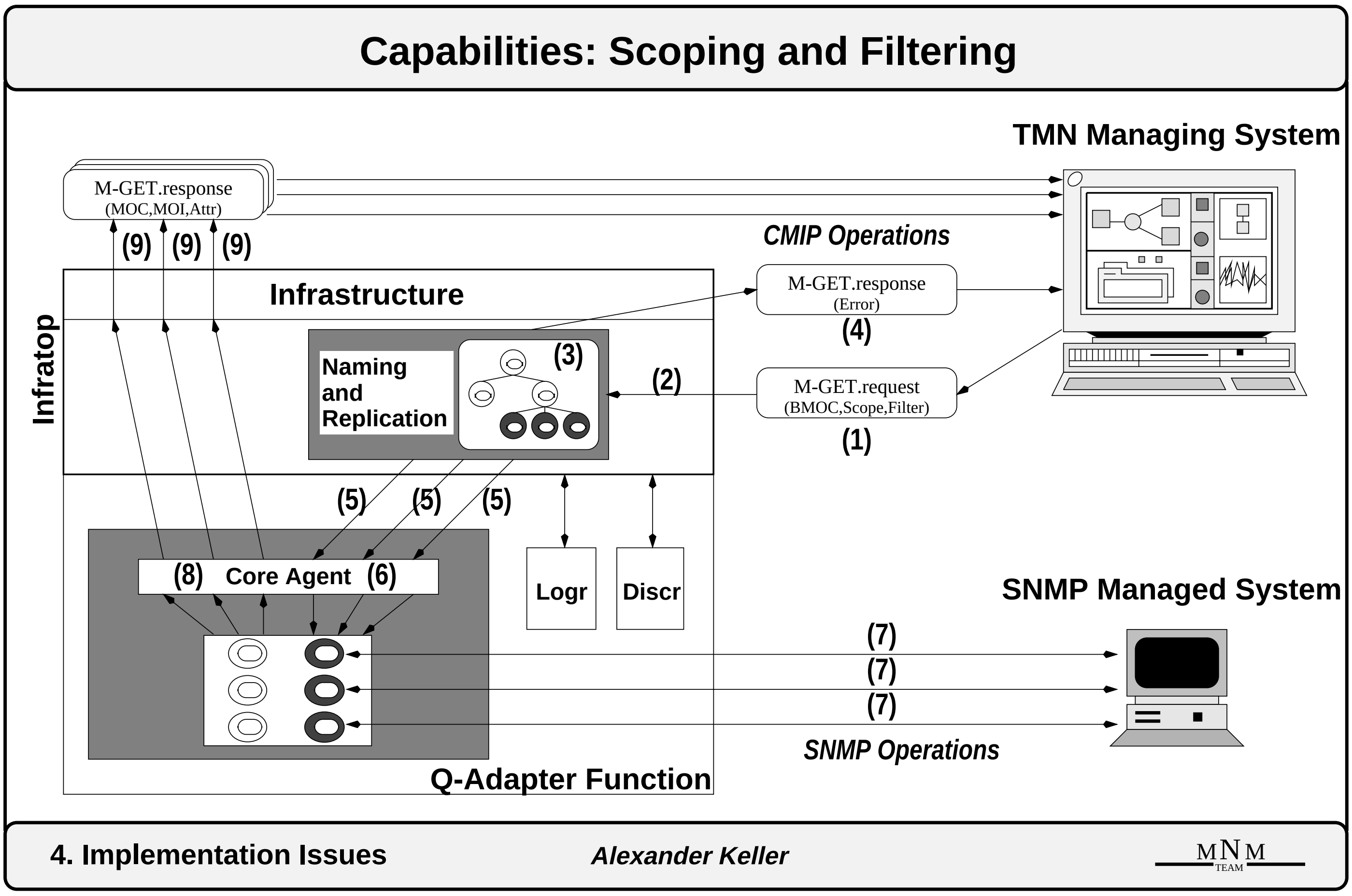
By introducing a QAF into the management environment, an additional hierarchical level is inserted between the managing system and the managed system. Recall from section 1.1 that from the perspective of an OSF, a QAF is considered as an agent. In fact, this is the reason why we were able to use a TMN agent development toolkit (see section 3.1) for the implementation of the Q-adapter. The OSF interacts with the QAF by means of the Q3-interface; the QAF itself is perceived as managing system by the SNMP managed devices and communicates with them through the m-interface (here: SNMP). As obviously no SNMP-API is shipped with the TMN development environment, we had to encapsulate an already existing C-API for SNMP (the ucd-snmp API) in order to trigger SNMP-PDUs in the C++ callbacks.

During runtime the agent executable consists of four distinct processes (shaded areas in the above figure):

- The **Infratop** process consists of the components *Infrastructure* and *Naming and Replication*. The process is crucial for the function of the QAF since it implements CMISE and ACSE, maintains the containment hierarchy and checks the name binding restrictions in case of M-CREATE and M-DELETE requests. Furthermore, it provides the scoping and filtering functionality by replicating the requests to the concerned MOIs.
- **Discr** is the event handling component which is compliant to the *Event Report Management Function* [10].
- **Logr** maintains the log records according to the *Log Control Function* [11].
- The **Agent**, finally, contains the managed object instances (MOI), the core agent and facilities for resource access. It may be either single- or multithreaded.

4 Implementation Issues

4.1 Performing scoped and filtered M-GET operations

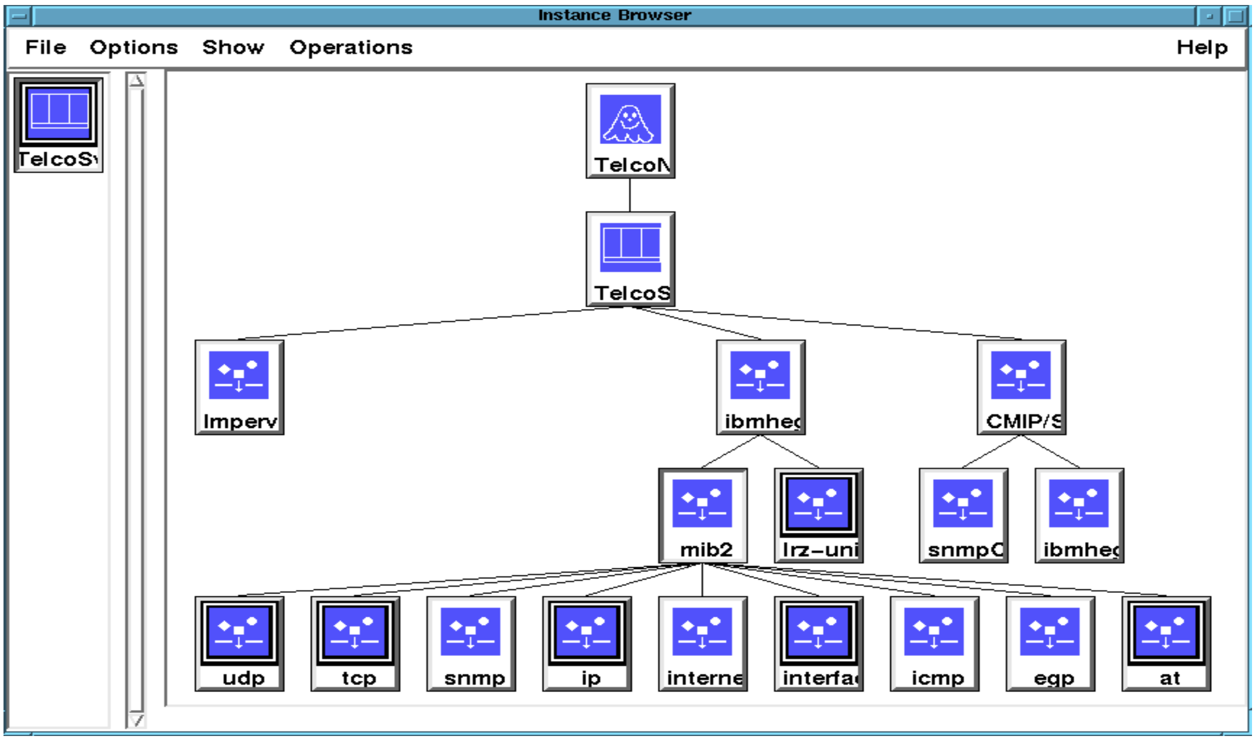


In order to clarify the cooperation of the distributed management environment, we will describe now how a scoped and filtered M-GET.request is handled by the QAF.

1. The OSF issues through its CMIS/ACSE stack a scoped and filtered M-GET.request on a *Base Managed Object Class (BMOC)* of the QAF.
2. The *Infrastructure* receives an M-GET.indication and passes the parameters to the *Naming and Replication* component.
3. The instances contained in the scope are determined.
4. If the base MOI of the request is not present in the containment hierarchy, an M-GET.response containing the *noSuchInstance* message is returned to the OSF.
5. If the scope can be determined, every concerned MOI in the core agent receives a replica of the request.
6. The core agent locates the MOI and executes the request on the MOI.
7. This triggers the callback method of the addressed attribute. The callback method determines the OID and invokes an SNMP GET request on the managed resource through the SNMP interface. The SNMP GET response returns the value of the addressed SNMP MIB variable.
8. The *Core Agent* receives the result and forwards it to the *Infrastructure*.
9. The *Infrastructure* generates an M-GET.response message with linked replies. The OSF receives these replies (M-GET.confirm) through its CMIS/ACSE protocol stack.

4.2 Containment Hierarchy

MIB-II: The Containment Hierarchy



4. Implementation Issues *Alexander Keller*

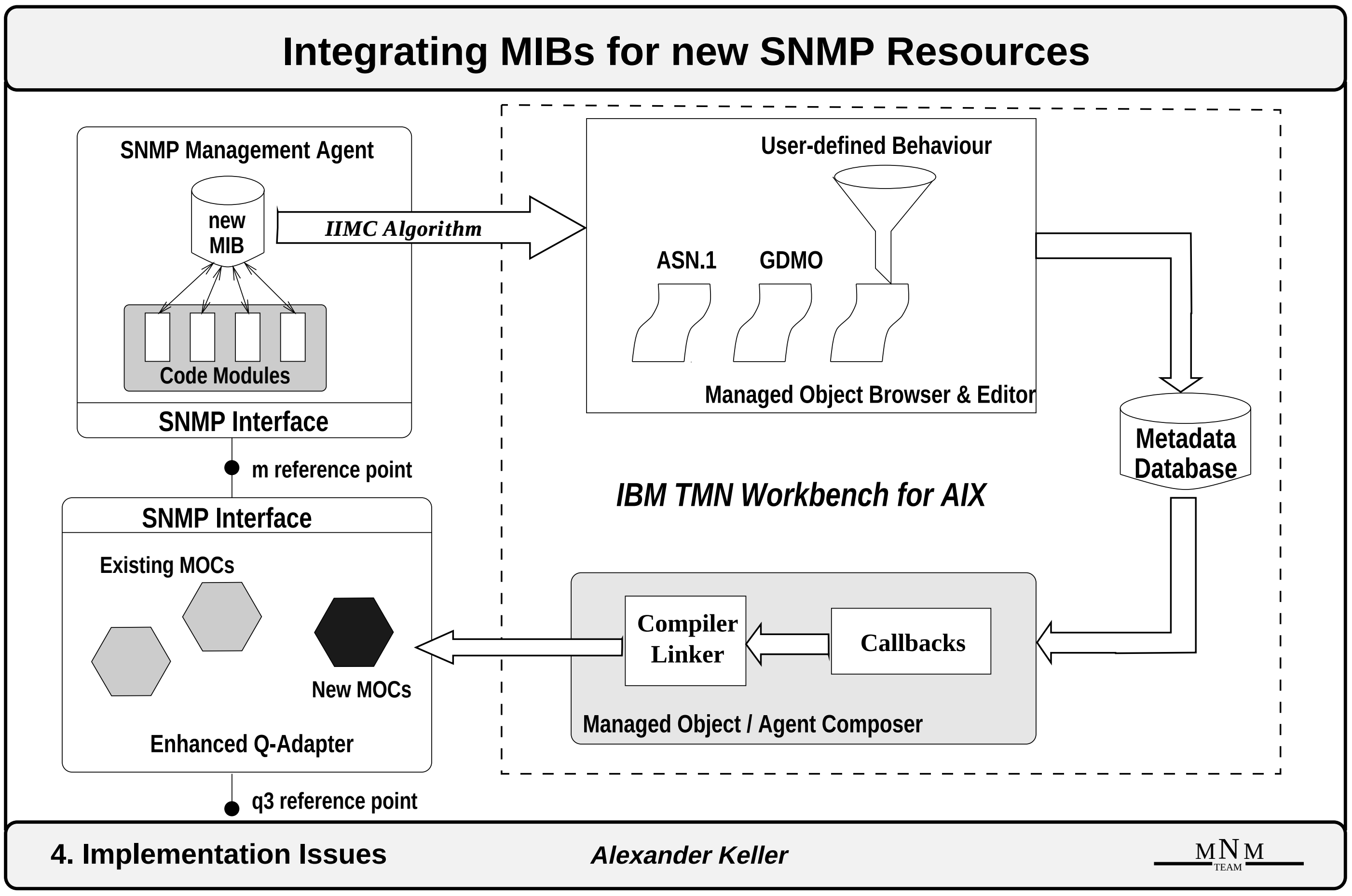
We will now describe how a user interacts with the QAF and the characteristics of our implementation.

At the current stage, our QAF prototype is able to handle two kinds of SNMP MIBs: Obviously, a QAF for SNMP must support MIB-II [20], the common MIB of all SNMP-capable resources. The second MIB ("lrz-unix") has been developed by our research group and covers the typical administration and maintenance tasks of UNIX workstations: It allows the definition of users and user groups and the assignment of quotas for disk space and printing facilities. Furthermore, different kinds of devices (disk partitions, filesystems, disks, tapes etc.) can be managed; it is also possible to create and delete processes. The emphasis lies in having not only monitoring capabilities, but also facilities for issuing administrative commands via SNMP.

The representation of these two MIBs in the QAF sums up to roughly 60 MOCs; these MOCs contain together about 600 attributes. For the sake of brevity, the screendump shown above depicts only the MIB-II part of a managed system (here: Relative Distinguished Name *ibmhegering1*; in the center of the figure) as it is perceived by the user on the WSF console. Operations on these MOIs are issued by dragging the icon of the desired MOI on an icon representing the desired operation (M-GET, M-SET, etc.) and by entering the appropriate parameters into a pop-up menu (defining the scope and filters and the type of synchronization). This is equivalent to the mechanism how TMN objects are managed and represented on the WSF, therefore providing seamless integration of SNMP managed objects into the TMN environment.

The second part of interest is the area which contains 3 MOIs ("CMIP/SNMP" and below; on the right side of the figure) that allow the configuration of the QAF itself during runtime through the same mechanism as described above. Management of the QAF is necessary because our QAF implementation performs several tasks itself such as caching of frequently requested MIB attributes. Through these MOCs, a user has, among others, the possibility of configuring the caching lifetime of attributes and their polling intervals. Furthermore, it is also possible to retrieve the values of several counters giving information on protocol errors.

4.3 Extending the Q-Adapter Function



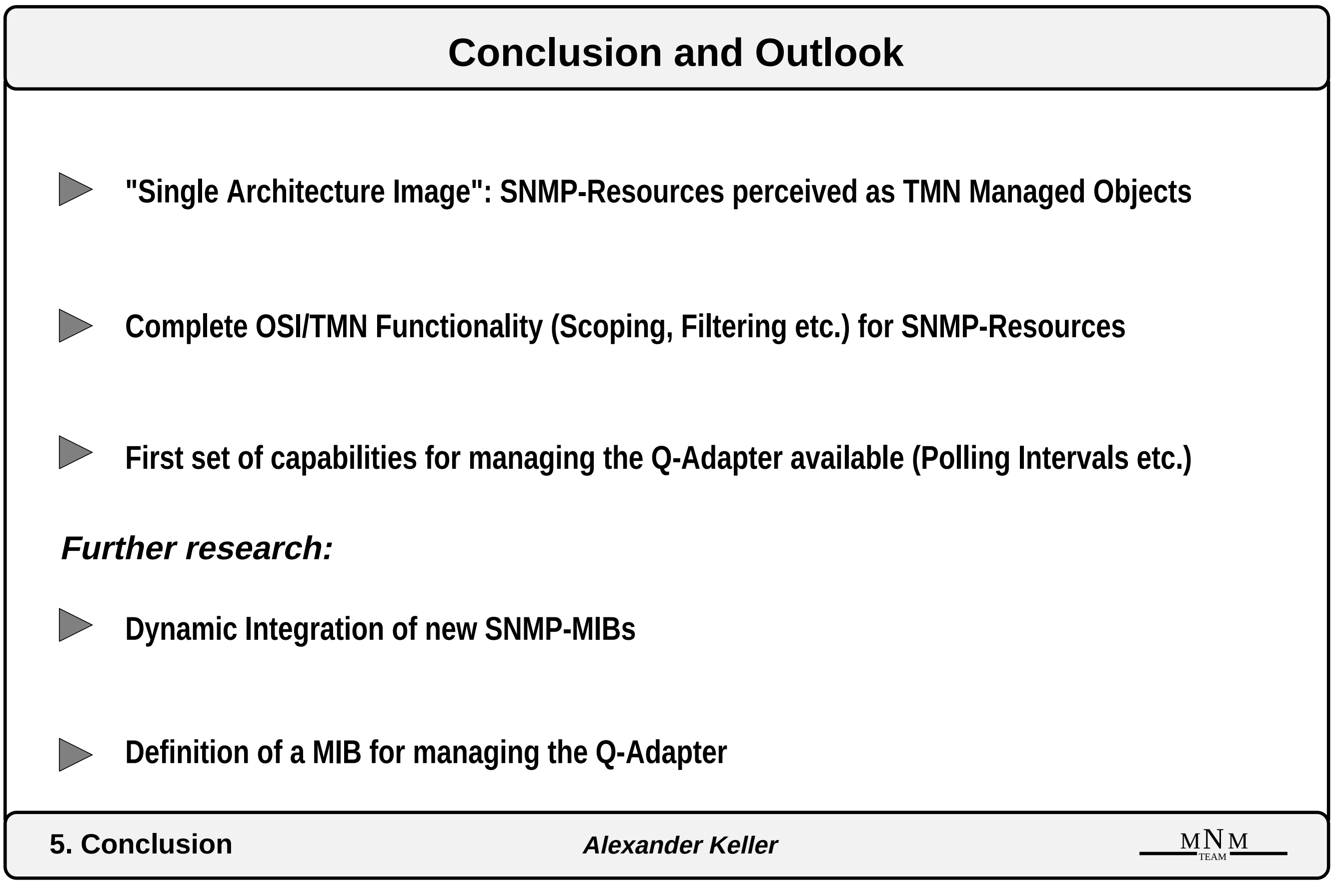
Given the large number of different SNMP MIBs that are being defined (e.g. Host resources MIB, HTTP MIB, Manager-to-Manager MIB etc.) and the amount of vendor- and device-specific MIBs, the support of only two MIBs of course is not sufficient: It is therefore necessary to have the opportunity of extending the QAF easily by new managed object descriptions.

This is done as follows: The new MIB has to be translated by means of the compiler according to the IIMC algorithm into GDMO templates and ASN.1 descriptions; eventually, the created name bindings and OIDs have to be adapted manually: After the translation, all the generated name bindings being necessary for the creation of the containment hierarchy have their superior object class set to “system”. For MOCs having a different superior MOC (the vast majority in our case), this must be changed manually to the appropriate values in order satisfy the name binding constraints for instatiating the MOCs. This requires knowledge of the underlying information models. If needed, user-defined behaviour can be added. These files are then entered into the IBM TMN Workbench by processing them with the *Managed Object Browser and Editor* which enters the new descriptions into the *Metadata Database*, a common and architecture independent repository for MOC descriptions.

After this has been done, the new MOCs are added to the already existing *ensemble* by means of the *Managed Object/Agent Composer*. This tool generates then the headers of the callback methods which have to be filled with the implementation code. In the case of the QAF, the calls to the SNMP API are entered. Finally, the generated code has to be compiled and linked with the already existing QAF code. The QAF is then extended to handle also the new MIB.

The flexibility for integrating different kinds of SNMP capable resources and the achieved ease of use for applying complex and powerful operations have a certain price in terms of manpower and system resources needed: It took about one man-year to design and implement the prototype. The executable code size is roughly five Megabytes if UNIX shared libraries are used; the stand-alone version of the QAF has a size of 40 Megabytes. The compile time for the QAF is about 60 minutes on an IBM RS/6000 workstation model 3BT (7030) equipped with 128MB of RAM. The development environment requires 250 MB of disk space.

5 Conclusion and Future Work


Conclusion and Outlook

- "Single Architecture Image": SNMP-Resources perceived as TMN Managed Objects
- Complete OSI/TMN Functionality (Scoping, Filtering etc.) for SNMP-Resources
- First set of capabilities for managing the Q-Adapter available (Polling Intervals etc.)

Further research:

- Dynamic Integration of new SNMP-MIBs
- Definition of a MIB for managing the Q-Adapter

5. Conclusion *Alexander Keller* MNM TEAM

This presentation has described a research project dealing with the design and implementation of a Q-adapter function for the integration of existing SNMP resources into a TMN environment. From the perspective of the managing system, these resources appear as OSI/TMN managed objects; therefore, the whole range of OSI/TMN functionality and features may be applied to them. This includes *scoping*, *filtering* and *synchronization*; it is also possible to delegate event handling and logging tasks to the QAF by defining *Event Forwarding Discriminators* and *Log Records*. Furthermore, we implemented a first set of managed objects allowing to manage the QAF itself from the OSF: This includes, among other, the setting of resource-specific polling profiles. Although we were able to base our development on a sound fundament (namely the IIMC concepts and the powerful IBM TMN development environment that shields completely the complexity of CMIP), we experienced that the design and implementation of a Q-adapter function for SNMP are far from being trivial:

The compiler based on the IIMC algorithm (described in section 2.1) does a very good translation job but can, at its current stage, only handle SNMPv1 managed object descriptions. The incorrect generation of name bindings and object identifiers (sketched out in section 4.3) requires some additional work.

Our experiences with the IBM TMN development toolset have been described in section 4 and can be summarized as follows: Apart from the large compile times and the considerable code size of the QAF, the experiences with this development environment were good although some deep understanding of OSI SMI is required.

Steps for further research include the enhancement of the QAF in a way that its set of MOCs can be extended during runtime; currently, the QAF needs to be recompiled if a new SNMP MIB is to be added. Another issue is the manageability of the QAF itself; we are working on the definition and implementation of a Q-adapter MIB allowing to lookup and modify the properties of the QAF. The experiences obtained in this project are being successfully reused in a follow-on research project dealing with the integration of SNMP resources in CORBA. This prototype is currently under development.

Acknowledgements

The author wishes to thank the members of the Munich Network Management (MNM) Team for helpful discussions and valuable comments on previous versions of the paper. The MNM Team directed by Prof. Dr. Heinz-Gerd Hegering is a group of researchers of the University of Munich, the Munich University of Technology, and the Leibniz Supercomputing Center of the Bavarian Academy of Sciences.

References

[1] Generic Network Information Model. Recommendation M.3100, CCITT, June 1992.

[2] Tom R. Chatt, Michael A. Curry, Juha Seppä, and Ulf Hollberg. TMN/C++: An object-oriented API for GDMO, CMIS, and ASN.1. In Lazar et al. [17], pages 177–191.

[3] Roberta Cohen. *The Telecommunications Management Network (TMN)*, chapter 9, pages 217–242. In Sloman [27], June 1994.

[4] M. Feridun, L. Heusler, and R. Nielsen. Implementing OSI Agents for TMN. IBM Research Report RZ 2759, IBM Research Division, Zurich Research Laboratory, 1995.

[5] Network Management Forum, editor. *Discovering OMNIPoint - A Common Approach to the Integrated Management of Networked Information Systems*. Prentice Hall, 1993.

[6] S. Heilbronner, A. Keller, and B. Neumair. Integriertes Netz- und Systemmanagement mit modularen Agenten. In C. Cap, editor, *Proceedings of SIWORK'96: Workstations und ihre Anwendungen*, pages 275–286, Zurich, Switzerland, May 1996. vdf Hochschulverlag AG an der ETH Zürich. ISBN 3-7281-2342-0.

[7] IBM Corporation, Research Triangle Park, NC 27709-2195. *IBM NetView TMN Support Facility for AIX Release 2: User's Guide*, March 1996. Order Number: SC31-8017-01.

[8] IBM Corporation, Research Triangle Park, NC 27709-2195. *IBM TMN Products for AIX Release 2: General Information*, release 2 edition, March 1996. Order Number: GC31-8016-01.

[9] IBM Corporation, Research Triangle Park, NC 27709-2195. *IBM TMN WorkBench for AIX Release 2: Managed Object/Agent Composer User's and Programmer's Guide*, March 1996. Order Number: SC31-8006-01.

[10] Information Technology – Open Systems Interconnection – Systems Management – Part 5: Event Report Management Function. IS 10164-5, ISO/IEC, June 1993.

[11] Information Technology – Open Systems Interconnection – Systems Management – Part 6: Log Control Function. IS 10164-6, ISO/IEC, June 1991.

[12] Information Technology – Open Systems Interconnection – Structure of Management Information – Part 2: Definition of Management Information. IS 10165-2, ISO/IEC, September 1991.

[13] Information Technology – Open Systems Interconnection – Structure of Management Information – Part 4: Guidelines for the Definition of Managed Objects. IS 10165-4, ISO/IEC, August 1991.

[14] Information Processing Systems – Open Systems Interconnection – Basic Reference Model – Part 4: Management Framework. IS 7498-4, ISO/IEC, November 1989.

[15] Pramod Kalyanasundaram and Adarshpal Sethi. Interoperability Issues in Heterogeneous Network Management. In Manu Malek, editor, *Journal of Network and Systems Management*, volume 2, pages 169 – 193. Plenum Publishing Corporation, June 1994.

[16] M. Langer. Entwurf und Implementierung eines CMIP/SNMP Gateways. Master's thesis, Technische Universität München, November 1996.

[17] Aurel A. Lazar, Roberto Saracco, and Rolf Stadler, editors. *Proceedings of the 5th IFIP/IEEE International Symposium on Integrated Network Management, San Diego*. IFIP, Chapman and Hall, May 1997.

[18] S. Mazumdar, S. Brady, and D. Levine. Design of Protocol Independent Management Agent to Support SNMP and CMIP Queries. In *Proceedings of the 3rd IFIP/IEEE International Symposium on Integrated Network Management*. North-Holland, April 1993.

[19] Kevin McCarthy, George Pavlou, Saleem N. Bhatti, and Jose Neuman de Souza. Exploiting the power of OSI Management for the control of SNMP-capable resources using generic application level gateways. In Raynaud et al. [25], pages 440–453.

[20] Keith McCloghrie and Marshall T. Rose. Management Information Base for Network Management of TCP/IP-based Internets: MIB-II. RFC 1213, IAB, March 1991.

[21] TMN C++ Application Programming Interface. Issue 1.0, draft 5 - for public comment, Network Management Forum, January 1996.

[22] Translation of Internet MIBs to OSI/CCITT GDMO MIBs. NMF 026, Network Management Forum, 1993.

[23] ISO/CCITT to Internet Management Proxy. NMF 028, Network Management Forum, 1993.

[24] George Pavlou, Kevin McCarthy, Saleem N. Bhatti, and Graham Knight. The OSIMIS Platform: Making OSI Management Simple. In Raynaud et al. [25].

[25] Yves Raynaud, Adarshpal Sethi, and Fabienne Faure-Vincent, editors. *Proceedings of the 4th International Symposium on Integrated Network Management, Santa Barbara*. IFIP, Chapman and Hall, May 1995.

[26] J. Reilly. VTT IIMC Notes - Modification to SMIC SNMP MIB Compiler to produce GDMO MIB Definitons. Technical report, Technical Research Centre of Finland, Telecommunications Laboratory (VTT/TEL), May 1993.

[27] Morris Sloman, editor. *Network and Distributed Systems Management*. Addison-Wesley, June 1994.