Integration of SNMP into a CORBA- and Web-based Management Environment

Gerd Aschemann Thomas Mohr Mechthild Ruppert

Department of Computer Science, Darmstadt University of Technology, Germany
{aschemann,mohr,ruppert}@informatik.tu-darmstadt.de

Abstract. The management of networks, distributed systems, and distributed applications is an important and growing field in computer science. Besides the classical architectures like SNMP and OSI management, the introduction of new technologies such as the World Wide Web with Java and the availability of standard middleware architectures like CORBA have brought up new challenges. Future developments in the management domain will probably be at least Web-based. However, in any case they should be "integrated".
We present some ideas to integrate both Web-based and CORBA-based techniques into management platforms and discuss new strategies for the integration of the emerging approaches and the classical architectures. We also describe our implementation of an SNMP-CORBA gateway which covers the translation of the information model as well as the communication model. We have extended our System Configuration Tool (SCOT) by a full CORBA interface. Its Web interface has been extended by a generic generator for HTML pages with integrated, and appropriately parameterised applets, which allow to access the SCOT repository as well as other CORBA-based management entities, such as our CORBA-SNMP gateway.

Keywords: CORBA, integrated management, network management, SNMP, Web-based management.

1 Introduction

The management of distributed systems and applications stems from network management, either the management of Wide Area Networks (WAN), which are classically covered by the Open Systems Interconnection (OSI) standards of the International Organisation for Standardisation (ISO), or Local Area Network (LAN) management, which is typically associated with the term Internet Management or SNMP (despite the fact that the Internet was a WAN from its first days). With the upcoming of distributed systems and distributed applications, new technologies arose, such as the World Wide Web (WWW) or shortly Web, and standardised middleware architectures like the Distributed Computing Environment (DCE) or CORBA. These new technologies are a challenge to the world of classical management since they proclaim to make a user's life easier (Web) or provide more general approaches (middleware) than specific solutions. Even in the field of distributed applications the question arises, why the distributed systems should be maintained with a separate protocol like SNMP instead of using the same middleware that is used for realizing the application. Nevertheless, the old world still exists besides the new world, and technologies for integration are required.

In the future, platforms for the management of networks, distributed systems, and distributed applications may have a more open architecture and may provide a framework for different components, like resources, user interfaces, data storages, management services and integration gateways. For the more complex interactions a standard middleware like CORBA may be used. But for the simpler interactions simpler technologies must be provided. On the one hand these simpler technologies are management protocols, like SNMP, for the management of network elements and simple services. SNMP is well established and it is a must to integrate it into any new platform approach. On the other hand, the Web with HTML/HTTP provides a well established user interface for distributed systems and cannot be ignored. It is well suited for more or less static information like the inventory database of a distributed system or the administrative state in the configuration repository. However, this technology reaches its borders when highly dynamic applications must be implemented, e.g., online monitoring of real resources, or real time manipulation of network elements. This cannot be achieved in a suitable way by static HTML pages or the typical interaction interfaces of the Web, i.e., CGI scripts and applications. Here it is the purpose of the platform to enable the development of integrating applications based upon orthogonal interfaces and services embedded into a middleware. With the upcoming integration of CORBA into standard Web browsers, the "static" HTML pages can be easily extended to dynamic applications, since Java as an application language within Web browsers is also well established. These applications may run with only little modifications as management user interfaces. However, Java enabled management applications will not completely replace HTML pages, due to their simplicity and the broad availability of this technology. We believe that there will not be the "one and only" technology for an application domain like distributed management, as other examples in the past have shown.

We enhance such an approach by implementing a gateway to administrate SNMP-manageable systems through CORBA-managers and propose an open environment for CORBA-based management with an integrated Web interface for simple access. Figure 1 sketches some components of such a platform. The solid lined components are already implemented, while the dotted components are subject to concurrent and future work (see Sect. 5).

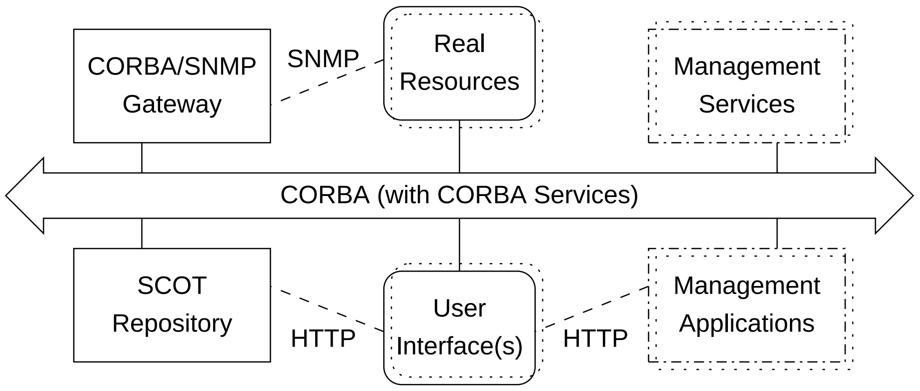


Fig. 1. Open integrated management platform

Structure of the paper

We start with a description of our MIB compiler in section 2 and of our gateway in section 3. Both sections give a short introduction to and a comparison of the appropriate models in SNMP and CORBA (information and communication model). Then the approach to translate one into the other is described. We show an example and give an overview of related work. Section 4 introduces our approach to Web-based management and the integration of the gateway. The last Sect. (5) outlines future work and presents a conclusion.

2 Information Models and the MIB Compiler

An integration of SNMP into a CORBA-based management environment requires two translations: a transformation of the information model and a transformation of the communication model. Our MIB compiler provides a facility to automatically generate a CORBA IDL specification from a given SNMP MIB.

2.1 SNMP Information Model

In SNMP the management information model is described using a subset of the Abstract Syntax Notation One (ASN.1). A generic document determines the structure of management information [19], [23], a so-called Management Information Base (MIB). A MIB is a formal view of the managed objects (MOs) which allow to manage network elements (NEs) as real resources. The description is data type oriented and gives a hierarchical view on a network element by grouping its objects or collections of objects as tables. SNMP allows the definition of traps and notifications as messages which are spontaneously sent from an NE (i.e., its management software, the agent) or another manager to the manager to communicate errors or exceptional behaviour of an object.

Many macros and MIBs are defined by the Internet Engineering Task Force (IETF) working groups, software and hardware vendors, research institutes or other organisations. All MIBs and defined hierarchical structures are embedded in the ISO naming tree [22]. The tree allows to address a particular object by its path through the tree, given by numbers. This path in its numeric form is called Object Identification (OID).

2.2 CORBA Interface Descriptions

The CORBA information model, though not explicitly named by this term, allows for a class-based description of data objects, i.e., their interfaces with attributes and methods are given in a completely object-oriented manner. The so-called Interface Description Language (IDL) has a syntax which is strongly related to C++. A hierarchical naming and scoping of definitions is provided by nested modules.

A CORBA IDL definition is usually translated into (static) stubs and skeletons of a programming language by an IDL compiler. During translation a unique and standardised identification (type identifier) is generated and assigned to the CORBA interfaces and types. This identifier is usually obtained from a certain prefix, a path description of nested modules and interfaces, and a version number. Optionally, all of these parts can be provided by the interface designer (`pragma`-keyword, see below). Together with an interface repository (IR) these unique identifiers allow for dynamic calls of operations (see Sect. 3.2).

2.3 Translation of MIBs into IDL

To translate SNMP MIBs to CORBA IDLs, we have implemented a compiler [20] which has to cope with several aspects. We do not describe all details of the transformation, but only sketch the most important features. Like any other compiler our translator has to cope with issues like different identifiers, access rights, import/export of identifiers, scoping rules, etc. However, we have to handle some special cases:

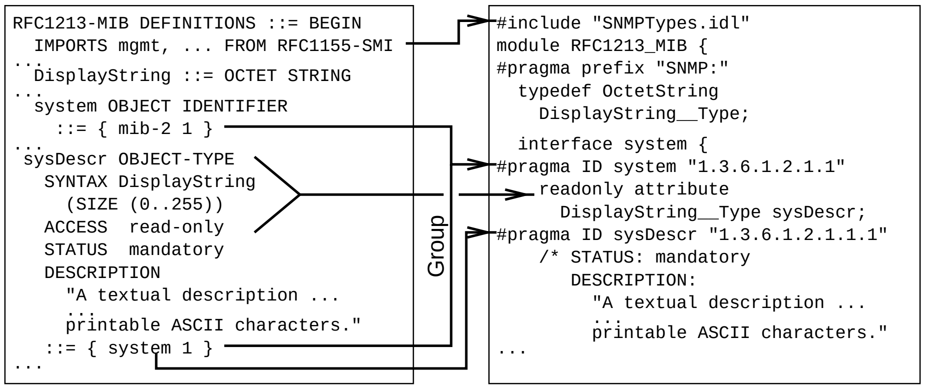
– SNMP macros are hard-coded in the compiler, since they only define syntactic and semantic structure, and analysis of the MIB and cannot be extended.
– In general, SNMP object definitions are translated to CORBA interfaces. We have to distinguish three cases. If the entity is simply an SNMP variable, the generated interface gets an attribute of the appropriate type. If the object belongs to a group (a `SEQUENCE` of SNMP objects), the group becomes an interface and the members become attributes, just as if they were ordinary SNMP variables. If the object belongs to a table, the table contains one special entry object which has the same structure as a group and is translated like a group.
  Additionally, IDL ID-pragmas are generated by concatenating the single numbers of the SNMP OID to allow for the easy mapping of CORBA objects to SNMP Object IDs and vice-versa within the gateway.
– Since it is only possible to define a few restrictions on types and ranges in IDL, e.g., the maximum number of elements within a sequence, we have to drop most of this information.
– In order to use traps and notifications of SNMP, we have defined appropriate generic IDL interfaces. The interfaces provide push and pull operations similar to the CORBA Event Service [13]. However, the methods have common SNMP specific parameters (e.g., community name, trap type, time ticks) and a generic parameter of the IDL-type `any` which holds trap-specific information.
– Parts of MIBs which cannot be mapped to IDL are inserted as comments, e.g., object descriptions or access rights other than readonly.

Figure 2 shows some capabilities of the MIB compiler.

2.4 Related Work

The Joint Inter Domain Management (JIDM) working group of the Open Group (formerly X/Open) has worked out a specification for the translation of the information models of SNMP and OSI/TMN management to CORBA [18]. We picked up some ideas from this work but also made some different design decisions. In most cases we dropped specifications from the JIDM design, since we found them overdone for this tool, e.g., mapping of all ASN.1 types to CORBA (one should keep in mind that JIDM tries to cover both SNMP and OSI management), or mapping of complex SNMP constants to an operation (which must be implemented somehow). One important aspect is that we generate pragma IDs for SNMP OIDs to support dynamic access to SNMP variables through the gateway. Some implementations of the JIDM proposal exist, e.g., [7] [11].

Besides the JIDM approach there exist several tools to translate ASN.1 to other languages. One of them is the Sample Neufeld ASN.1 to C/C++ Compiler (SNACC, [6]) and its derivatives, e.g., for Java [9].



```
RFC1213-MIB DEFINITIONS ::= BEGIN
  IMPORTS mgmt, ... FROM RFC1155-SMI
...
  DisplayString ::= OCTET STRING
...
  system OBJECT IDENTIFIER
    ::= { mib-2 1 }
...
 sysDescr OBJECT-TYPE
   SYNTAX DisplayString
     (SIZE (0..255))
   ACCESS  read-only
   STATUS  mandatory
   DESCRIPTION
     "A textual description ...
     ...
     printable ASCII characters."
   ::= { system 1 }
...
```

```
#include "SNMPTypes.idl"
module RFC1213_MIB {
#pragma prefix "SNMP:"
  typedef OctetString
    DisplayString__Type;

  interface system {
#pragma ID system "1.3.6.1.2.1.1"
    readonly attribute
      DisplayString__Type sysDescr;
#pragma ID sysDescr "1.3.6.1.2.1.1.1"
    /* STATUS: mandatory
       DESCRIPTION:
         "A textual description ...
         ...
         printable ASCII characters."
...
```

a) Fragment of MIB-II b) IDL code generated from MIB-II

Fig. 2. Translation of SNMP MIB to CORBA IDL

3 Gateway

After describing the “specification translation” of a CORBA to SNMP gateway, we now discuss the “interaction translation”.

3.1 Communications in SNMP

In order to make SNMP communications efficient, it was originally built using the User Datagram Protocol (UDP) of the TCP/IP family. So the main character of SNMP communications is that of an unreliable, asynchronous, message based protocol. SNMP knows basically five message types: set request, get request, get next request, response, and trap. Besides some general information like the SNMP version number, message type, request id, and authorisation information, each message contains a list of name-value pairs, a so-called variable binding list. The names are the OIDs of SNMP variables, the values occur only in set requests and in get and get next responses. Get next requests are used to traverse the variables in the order of the OID tree, especially in the case of tables, where the OID of the next variable is not a priori known. The definition of SNMPv2 additionally introduced bulk transfers [24] to minimise the number of protocol messages if, e.g., all values of a table must be transferred.

3.2 Communications in CORBA

While in SNMP there are stronger facilities to define data types, especially to restrict them, in CORBA communications are much more complex. Besides asynchronous (unreliable), synchronous, and deferred synchronous (both reliable) communications, CORBA has to cope with many other aspects, e.g., encoding of operation calls, or different parameter semantics.

However, the CORBA standard [16] proposes some facilities that are particularly useful in a bridging application like our gateway. The definition of the Dynamic Invocation Interface (DII) allows for a client to build up a request during runtime by manually packing the request message as an object, containing the reference of the object to be called, the name of the operation, and its parameters. The generic method `invoke` of the request object returns with the results of the called operation which can be unpacked by the client. The dynamic skeleton interface (DSI) allows the same on the server side, the so-called “servant” [21]. With the DSI, object dispatching is still left to the CORBA object adapter (OA), but operation dispatching is provided by the object itself. The object implements a method `invoke` which is called by the object adapter with passing a request object as a parameter.

As part of the DII specification, so-called Context objects were introduced. Context objects are a list of properties (name + string value pairs) which allow to pass additional information with the request to the servant, e.g., about the client environment. They can be seen as dynamic (named) parameters to the servant method, since normally only static parameters defined in the IDL specification are possible.

3.3 Mapping of CORBA Calls to SNMP Requests

Our gateway [12] is currently implemented by five different processes, as shown in Fig. 3. Since we mainly had the integration of SNMP into CORBA in mind, the main access point for CORBA applications is the CORBA Half-bridge (CB). Calls on SNMP objects are performed through the CORBA DII and DSI. Besides the request parameters, encoded in a request object, a context object holds the SNMP agent specific information (agent context), i.e., host, port number, community string etc.

Incoming requests are passed to the SNMP Half-bridge (SB) after mapping the IDL attribute to the corresponding SNMP variable by extracting the SNMP OID from the CORBA IR. Additional parameters for this request are the unchanged agent context and a newly obtained SNMP request context object (invocation context) from the Gateway Information Centre (GIC). This invocation context is later used to associate the SNMP response to the CORBA request. It may also be used to associate errors for this particular SNMP request detected by the Trap Daemon (TD), e.g., authentication errors.

The SB acts as an SNMP entity in manager role and is responsible for encoding and decoding of SNMP messages, i.e., conversion from and to CORBA requests, and communication with SNMP agents. If a request refers to an SNMP table, SB autonomously performs a number of get next requests (or get bulk in SNMPv2) to traverse the table and returns the whole table at once to the caller (by passing it through the CB).

The gateway works as a protocol converter and does not manage any information about potential communication partners, i.e., SNMP agents. To store and provide such information is the duty of a gateway user, who may delegate this task to a particular service or repository, e.g., our SCOT repository (Sect. 4).

It is easily possible to extend the gateway by additional SNMP MIBs, by simply loading them into the CORBA IR. No recompilation or even restart of the gateway is necessary. The gateway can cope with different SNMP versions in a transparent way for the client. In its multi threaded version, the gateway enables concurrent requests from CORBA managers to SNMP managed objects.

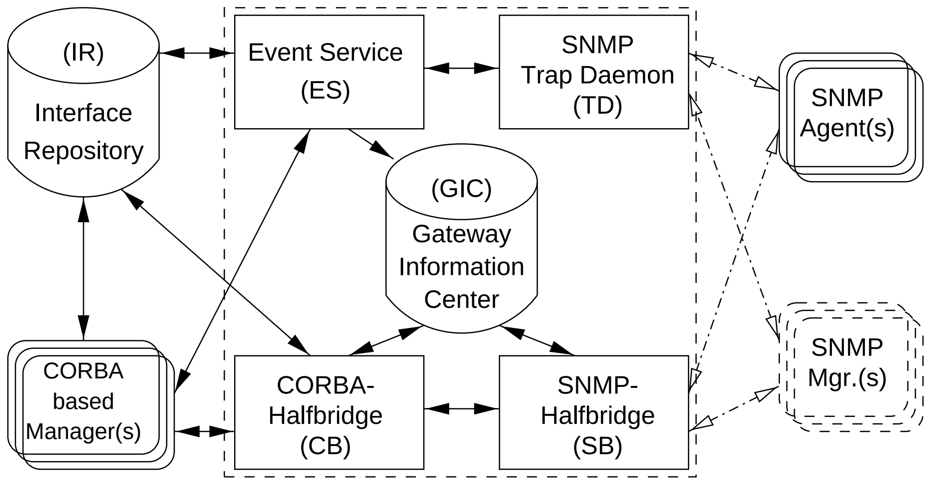


Fig. 3. Components of the gateway

3.4 Mapping of SNMP Traps

The TD receives any trap or notification sent by an SNMP agent. If the message contains an SNMPv2 inform notification, it sends an appropriate confirmation to the originator. Then it forwards the message to the Event Service (ES). The ES unpacks the message and constructs an appropriate CORBA object with the trap type information stored in the IR and forwards this object to the GIC where it might be used to respond to an erroneous client request as described above. Additionally, the event object is stored and can be requested later (polled) by a CORBA client. ES works similar to a specialised CORBA event service which provides forwarding of messages in publisher/subscriber manner and allows for the subscription of additional clients. CORBA manager applications may also register as event consumers with the ES, similar to the standard CORBA event service. Currently it only provides a push interface and untyped events, but can be easily extended by pull interfaces and typed events.

3.5 Implementation and a Client Example

The gateway itself is implemented in C++, currently based on ORBacus [17]. The clients, however, can be implemented in any language which is supported by CORBA and with any inter-operable broker which implements the Internet Inter-ORB Protocol (IIOP). We give a short example in Fig. 4: A context object is acquired from the ORB and the dynamic parameters are stored; a request to get a MIB table is made and afterwards the result can be unpacked.

Table 1 shows some performance measurements of our gateway obtained on a small network of PCs (200 MHz Pentium, Suse Linux 5.1) connected by a shared 10 MBit/s Ethernet. As one would expect, involving the gateway makes the communication much slower. The first row shows the average time needed to complete a call (request + response). In the case of a single value it is slowed down by a factor of nearly 41. Retrieving an SNMP table reduces the overhead and the gateway is only slower by a factor of

```
...
// get the gateway reference somehow
org.omg.CORBA.Object gateway = ...
org.omg.CORBA.Context ctx = orb.get_default_context ();
org.omg.CORBA.Any somedata = orb.create_any();
somedata.insert_string ("1"); ctx.set_one_value ("Version", somedata);
somedata.insert_string ("130.83.14.64");
ctx.set_one_value ("ipaddress", somedata);
org.omg.CORBA.Request request
    = gateway._request ("_get_RFC1213_MIB__snmpInPkts");
request.ctx (ctx);
// the result is always a table, even if it contains only one element
request.set_return_type (tableTagHelper.type());
request.invoke();
org.omg.CORBA.Any res = request.return_value();
tableTag result = tableTagHelper.extract (res);
...
```

Fig. 4. Fragment of CORBA-Java client for the gateway.

3.75. However, the second row shows the real time spent by a command line version of both implementations (using `snmpget` and `snmpwalk` of the UCD SNMP implementation). The response time is much below one second, so it is sufficient for a human user. Only in a few cases of automated tasks, e.g., monitoring of values with a high frequency, it is not appropriate to use the gateway.

Table 1. Performance measurements of requests made through the gateway (Times in ms)

	Single Variable			SNMP Table		
	GW	SNMP	Factor	GW	SNMP	Factor
Call only	29.141 ms	0.717 ms	40.64	61.497 ms	16.479 ms	3.73
Execution time	94.0 ms	38.0 ms	2.474	264.0 ms	59.0 ms	4.475

3.6 Related Work

As for the MIB compiler, we were inspired by the ideas of the JIDM group which resulted in an OMG proposal [15]. It defines a large class hierarchy for SNMP objects and relies on the extension of many CORBA services such as naming service, property service, etc. and provides a powerful framework. However, it also introduces some overhead, e.g., to finally access an SNMP MIB variable. JIDM introduces two new services, a specialised SNMP Naming Service and a specialised SNMP Object Interface Repository. Both specifications are derived from the standard CORBA Naming Service and the standard CORBA Interface Repository. The CORBA SNMP Naming service is used to build a naming hierarchy starting with a virtual node, the SNMP-MIB-ROOT. Beyond this virtual node, the particular hosts in the management domain of a particular gateway each have their own subtree (based on the DNS host name or the IP address of the host). Within one subtree the different MIBs which are accessible on a host can be found. The MIBs contain the SNMP variables, groups, and tables with their symbolic SNMP names. To find the corresponding SNMP OID of an SNMP entity, the symbolic names are mapped to their SNMP OID by the SNMP Object Interface Repository. To retrieve the actual OID of an SNMP instance, at first the Naming Service has to be traversed to find the symbolic name via the host subtree. Then the SNMP Interface Repository has to map the symbolic name to the SNMP OID. Both retrievals require some interactions between the gateway and the involved service. We believe that both mappings are orthogonal and should not be combined. In our approach the retrieval of the host information is left to the client, which may already have this information. In the next section we introduce our generic repository for host configuration information which is one example how clients could gain host specific information which might not only be used in the context of SNMP requests. Additionally we use the standard CORBA Interface Repository to map symbolic SNMP names to SNMP OIDs and therefore avoid the introduction of a new service.

[7] introduces proxy objects (shadow objects) for SNMP objects within its gateway, i.e., caches. This has the advantage that some get requests can be directly answered by the gateway without any SNMP communication. On the other hand it introduces a cache consistency problem, especially if the SNMP resources are concurrently accessed from other managers than the gateway. The thesis [7] states that complex methods to minimise (if not to avoid) consistency problems would be necessary but leaves the solution to this problem open.

[4] summarises two other dynamic approaches to make SNMP resources available through CORBA. Both (GOM and Liaison) do not care about making the exact SNMP MIB specifications available by translating the MIB to an IDL description. They only provide access to SNMP variables by a generic mapping of strings to variable names.

4 Integration of SNMP and SCOT

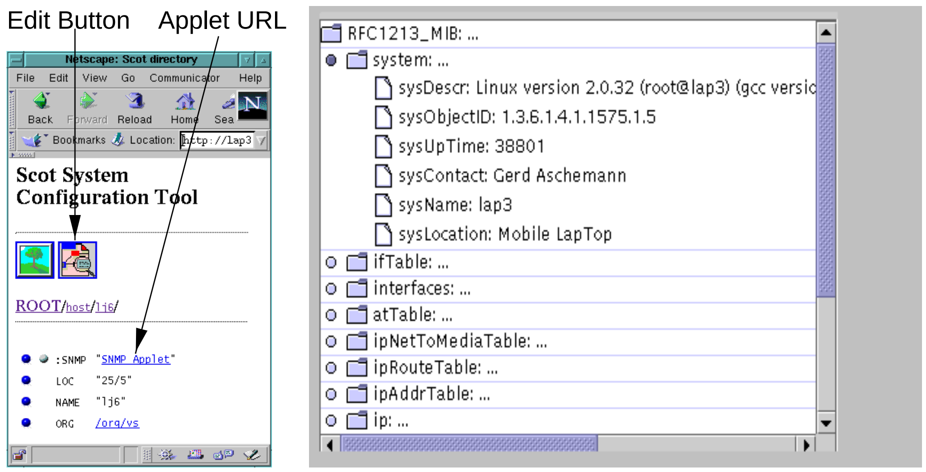
In [1] we introduced our System Configuration Tool (SCOT). It provides an object repository based upon the prototype instance model [10] which allows for value sharing among objects. Managed objects are described within repository objects as a list of slot objects. A slot is a tuple $S = (name, value, attributes)$. It cannot only hold values and references to other objects but also algorithms (Lisp functions and lambda expressions since it is implemented in Lisp) and therefore is able to compute complex object relationships based upon an embedded object-oriented query language. The main purpose is to provide a specification-driven information model for systems and network management. Administrators can directly specify the administrative state of managed objects. In fact, they can even concentrate on the interesting features they want to model. Every object can serve as a prototype for other objects, which extend or specialise its description. Classical information models as in SNMP or OSI management have a large overhead of operational state descriptions which are not necessary for some administrative tasks like configuration management. Since the repository is embedded into a Lisp platform, it is easy to define management policies and constraints as invariants which can be automatically checked by the repository.

4.1 CORBA Interface

SCOT originally provided only a very simple CORBA interface which allowed to access objects via Lisp expressions as string parameters to a CORBA operation. We have extended the implementation of repository objects and slots by making them directly available as CORBA objects through the CORBA compliant ILU package (Inter Language Unification, [27]). Objects and slots now can be directly accessed by appropriate operations (get and set functions, methods to insert, delete, etc. objects and slots). CORBA management applications can thereby use and manipulate the repository, e.g., to get the context information of a network element for SNMP access through the SNMP CORBA gateway (see above).

4.2 Web Interface

SCOT originally allowed to browse the repository through a Web interface. The repository generated HTML code representing the state of the requested object. With the CORBA extension it became possible to directly launch management applications as Java applets from the object specific HTML pages. Such applications can be common utilities like an object editor for the administrative state of the objects or a generic SNMP MIB browser or more dedicated applications like performance monitoring tools, event browsers, or user interfaces of other CORBA management applications (see Sect. 5).


Edit Button
Applet URL
Netscape: Scot directory
File Edit View Go Communicator Help
Back Forward Reload Home Search
Bookmarks Location: http://lap3

Scot System Configuration Tool

ROOT/host/lj6/

```
:SNMP  "SNMP Applet"
LOC    "25/5"
NAME   "lj6"
ORG    /org/vs
```

```
RFC1213_MIB: ...
  system: ...
    sysDescr: Linux version 2.0.32 (root@lap3) (gcc versic
    sysObjectID: 1.3.6.1.4.1.1575.1.5
    sysUpTime: 38801
    sysContact: Gerd Aschemann
    sysName: lap3
    sysLocation: Mobile LapTop
  ifTable: ...
  interfaces: ...
  atTable: ...
  ipNetToMediaTable: ...
  ipRouteTable: ...
  ipAddrTable: ...
  ip: ...
```

Fig. 5. SCOT standard HTML page and SNMP browser applet

Figure 5 shows an example of a repository object and the launch points for two different applets and a generic SNMP MIB browser as an example for such an applet. The browser can be started by clicking on the embedded URL in the `:SNMP`-slot of the shown object. Such applets must be parameterised by object specific values. The parameters are, e.g., its stringified inter-operable object reference (IOR) or a reference to the object in a name-server, the available MIBs on the real resource, etc. We have implemented a generic CGI script to start such an parameterised applet with appropriate HTML code.

4.3 Related Work

With the growing popularity of the WWW several approaches for the integration of Distributed Management into Web-based user interfaces have been proposed (see appropriate conferences, e.g., NOMS and IM). Only few of them provide a generic architecture to cover or at least integrate the whole spectrum of management issues.

In general it can be said that there are two directions for current development. Some approaches, e.g., Web-based Enterprise Management (WBEM, [26]) and the Java Management API (JMAPI, [25]), try to introduce completely new models and service architectures: WBEM with the Common Information Model (CIM), and the so-called CIM Object Manager (CIMOM), JMAPI with the JMAPI Object Model, the so-called Appliances, and communications based on the Java Remote Method Invocation (RMI). Both architectures try to integrate other architectures, mostly SNMP, by internal gateways. Others try to provide a seamless integration between the different management worlds by a translation or gateway approach, e.g., Liaison [2] (see also Sect. 3), Integrated Web-Based Management Architecture [8], or the Web-based TMN integration proposed in [3]. Our architecture belongs to the second group and tries to combine the established Web technologies for simple tasks as browsing through resources and visualisation with automatic translation of protocols on a CORBA basis. We try to rely on HTML/HTTP as much as possible and only use Java applets if the HTML/HTTP-based access is not sufficient.

5 Future Work and Conclusions

We are currently investigating the implementation and integration of additional services to provide an open integrated platform for the management of networks, distributed systems, and distributed applications as required in Sect. 1 and outlined in Fig. 1.

Besides the currently implemented repository, the Web interface, and the CORBA SNMP gateway which is an example for management gateways in general, we are going to integrate a service for management scripts [5] in the near future. Other components will be a CORBA-based agent on the managed resources and a stand-alone user interface (Web-based user interfacing is sometimes too restricted for some purposes). Besides these generic components we are developing more complex management applications like a migration tool for services which may rely on highly sophisticated CORBA services like the Object Transaction Service [14].

We have presented the design and implementation of an gateway for the access of SNMP resources by CORBA managers. It has minimal overhead in comparison to other approaches while allowing dynamic access and runtime extension. It can be run stand-alone to integrate SNMP with CORBA management. Furthermore, we have extended our SCOT repository by a full CORBA interface. In combination with SCOT's Web interface and the CORBA-based SNMP gateway this allows for the complete integration of SNMP management into Web-based management without introducing completely new protocols and services.

References

1. G. Aschemann and R. Kehr. Towards a Requirements-based Information Model for Configuration Management. In *Proceedings of 4th International Conference on Configurable Distributed Systems*, pp. 181–189. IEEE Computer Society Press, May 1998.
2. F. Barillaud, et al. Network Management using Internet Technologies. In *International Symposium on Integrated Management (IM)*. 1997.
3. Z. Canela, et al. Integrating Web-Based User Interfaces in TMN Systems. In *Network Operations and Management Symposium (NOMS)*. 1998.
4. L. Deri and B. Ban. Static vs. Dynamic CMIP/SNMP Network Management Using CORBA. In *4th International Conference on Intelligence in Services and Networks (IS&N)*. 1997.
5. R. Fröhning. *Entwurf und Implementierung eines CORBA-basierten Servers für Skripte und autonome Applikationen zum Management verteilter Systeme und Netze*. Diploma's thesis, FG Verteilte Systeme, FB Informatik, TU Darmstadt, Aug 1998.
6. GMD Fokus. SNACC Homepage. `http://www.fokus.gmd.de/ovma/freeware/snacc/entry.html`.
7. T. Höller. *Entwurf und Realisierung eines CORBA/SNMP Gateways*. Diploma's thesis, TU München, Aug 1996.
8. J. W.-K. Hong, et al. Web-based Intranet Services and Network Management. *IEEE Communications Magazine*, pp. 100 – 110, Oct 1997.
9. IBM. SNACC for Java. `http://www.alphaworks.ibm.com/formula/snaccforjava`.
10. H. Lieberman. Using Prototypical Objects to Implement Shared Behavior in Object-Oriented Systems. In *Proc. of the OOPSLA '86*, pp. 214–223. Oct 1986.
11. S. Mazumdar, et al. Web based management: Corba/snmp gateway approach. `http://nsm.research.bell-labs.com/~mazum/CorbaSnmp/`, 1998.
12. T. Mohr. *SNMP-Management mit CORBA — Transformation des Kommunikationsmodells*. Diploma's thesis, FG Verteilte Systeme, FB Informatik, TU Darmstadt, Apr 1998.
13. Object Management Group. CORBA Event Service Specification, Mar 1995.
14. Object Management Group. CORBA Object Transaction Service Specification, Nov 1997.
15. Object Management Group. CORBA/TMN Interworking - SNMP Part, Feb 1998.
16. Object Management Group. The Common Object Request Broker: Architecture and Specification, Revision 2.2, Feb 1998.
17. Object Oriented Concepts, Inc. *ORBacus for C++ and Java*, 1998.
18. Open Group. Inter-Domain Management: Specification Translation, 1997.
19. M. Rose and K. McCloghrie. RFC 1212: Concise MIB definitions, Mar 1991.
20. M. Ruppert. *SNMP-Management mit CORBA — Transformationen im Informationsmodell*. Diploma's thesis, FG Verteilte Systeme, FB Informatik, TU Darmstadt, Apr 1998.
21. D. C. Schmidt and S. Vinoski. Object Adapters: Concepts and Terminology. *SIGS C++ Magazine*, 9(11), NovDec 1997.
22. M. Sloman, ed. *Network and Distributed Systems Management*. Addison-Wesley Publishing Company, 1994.
23. SNMPv2 Working Group, et al. RFC 1902: Structure of Management Information for Version 2 of the Simple Network Management Protocol (SNMPv2), Jan 1996.
24. SNMPv2 Working Group, et al. RFC 1905: Protocol Operations for Version 2 of the Simple Network Management Protocol (SNMPv2), Jan 1996.
25. Sun Microsystems, Inc. Java Management API.
26. J. P. Thompson. Web-Based Enterprise Management Architecture. *IEEE Communications Magazine*, pp. 80 – 86, Mar 1998.
27. Xerox Palo Alto Research Center (PARC). Inter Language Unification. `ftp://ftp.parc.xerox.com/pub/ilu/ilu.html`.