Open Source Components for Internet Management by Delegation

F. Strauß, J. Schönwälder
Computer Science Department
Technical University Braunschweig
Bültenweg 74/75
38106 Braunschweig
Germany
{*strauss,schoenw*}*@ibr.cs.tu-bs.de*

J. Quittek
C&C Research Laboratories
NEC Europe Ltd.
Adenauerplatz 6
69115 Heidelberg
Germany
quittek@ccrle.nec.de

Abstract

The joint *Jasmin* project of the Technical University of Braunschweig and NEC C&C Research Laboratories is concerned with the development, implementation and practical evaluation of the management by delegation architecture standardized by the Distributed Management (DISMAN) working group of the IETF. This paper presents the open source software components that have been developed during the past two years within the Jasmin project, namely the Jasmin `DISMAN-SCRIPT-MIB` SNMP agent with its Java and Tcl runtime engines, a Java package for higher level operations on `DISMAN-SCRIPT-MIB` and `DISMAN-SCHEDULE-MIB` agents, a Java GUI application for interacting with `DISMAN-SCRIPT-MIB` agents, a Java AgentX sub-agent toolkit that simplifies the development of scripts exporting new MIB objects, a package that facilitates the development of monitoring scripts, and a `DISMAN-SCHEDULE-MIB` sub-agent. Besides describing the software components and their relations, we also present some project history and the experience we made specific to open source software development.

Keywords

Internet Management, Distributed Management, Management by Delegation, AgentX, Script MIB, Schedule MIB, Jasmin, Open Source

1 Introduction

One approach to build distributed management systems is the dynamic delegation of management functions, also known as the Management by Delegation (MbD) model [1]. The IETF chartered the Distributed Management (DISMAN) working group [2] in 1996 to define a standard which integrates the dynamic delegation model into the Internet management framework [3]. This working group produced among other things the `DISMAN-SCRIPT-MIB`, which was published as a Proposed Standard in 1999 [4].

A joint project called *Jasmin* was started in 1998 between the Technical University of Braunschweig and NEC C&C Research Laboratories to evaluate the management by delegation architecture standardized by the DISMAN working group. This includes the development of a prototype implementation and a practical evaluation in real-world management scenarios. Several software components have been realized in this project and most of them are openly available.

This paper has been written with three goals in mind. The first goal is to give an overview of the various open source software components produced within the Jasmin project. We briefly summarize the technical aspects and provide references where more details can be found. The second goal is to describe the technical and historical relations between these components. Finally, we present some experiences we made in this project that are specific to the development and usage of open source software.

The paper is organized as follows: Section 2 gives an overview of the history of the Jasmin project. This reveals some background information on why and when the individual components have been developed. It also uncovers our development experiences with respect to foreign project feedback, software licensing issues, and moments of fun. Most of these experiences reflect what Eric S. Raymond described in his well known article "The Cathedral and the Bazaar" [5, 6]. Section 3 presents some technical details of the open source components developed in the Jasmin project: the Jasmin `DISMAN-SCRIPT-MIB` agent, the `disman` Java package for high-level MIB access, the Java GUI application `Smurf` for simplified intuitive interaction with the `DISMAN-SCRIPT-MIB` and `DISMAN-SCHEDULE-MIB`, the Java AgentX sub-agent toolkit `JAX`, a simple Java SNMP monitoring package, and a `DISMAN-SCHEDULE-MIB` sub-agent. The summary of some lessons we have learned from our open source project is given in Section 4 before we conclude in Section 5.

2 History of the Jasmin Project

The Jasmin project [7] started back in February 1998 as a joint project between the Technical University of Braunschweig and NEC C&C Research Laboratories. The primary goal at that point in time was to implement a `DISMAN-SCRIPT-MIB` [4] agent that is capable of executing Java "scripts".

Even though the focus was on a Java runtime system, we decided to separate the runtime system from the SNMP agent so that (a) it would be easy to add additional runtime systems in the future and (b) the core agent could be developed at the Technical University of Braunschweig, while the Java runtime engine could be developed independently by NEC. We decided to define a simple line-oriented TCP-based protocol for the communication between the core agent and the runtime engine(s). One design goal was to make the implementation of runtime systems as simple as possible. The protocol was named the Script MIB Extensibility Protocol (SMX) and published as an experimental RFC [8] in May 1999.

By making the SMX specification publicly available and by simplifying runtime engine implementations as much as possible, we hoped to encourage other people to implement the SMX protocol. It was enjoyable to later hear from some researchers working at the University of the Sinos River's Valley in Brazil who have been developing a Perl runtime engine as part of a project in which the Jasmin agent is used to control Perl scripts for monitoring ATM links [9].

Another important early design decision introduced a second software interface: The SNMP toolkit related part of the agent has been separated from the core Script-MIB implementation by an internal interface. This allowed us to start working on the agent implementation using the commercial `EMANATE` toolkit from SNMP Research, which was the only stable SNMPv3/VACM aware SNMP toolkit available when the project started. Although from the technical point of view, `EMANATE` was a reliable basis for our work on the Jasmin agent, the licensing conditions caused awkward problems: In order to give our Jasmin agent to third parties interested in our work, we had to draw up and sign contracts that reflect all aspects of our licensing agreement with SNMP Research, e.g., with respect to forbidding reverse engineering `EMANATE` code. This has been a really unpleasant situation, since we in fact had no interests in protecting our own work in this regard, but anyhow, we had to fiddle with these legal issues meticulously.

About one year later, the Jasmin agent was ported to use the open source `NET-SNMP` toolkit[1] [10] that then offered sufficiently stable SNMPv3/VACM support.[2] This allowed us later to make the whole Jasmin software openly available under the GNU General Public License (GPL). Since both projects, `NET-SNMP` and Jasmin, were actively going on, it seemed to be better to not just submit the Jasmin agent code to the `NET-SNMP` project, but instead to distribute it as a separate package that is based on `NET-SNMP`. To achieve this, we re-worked the feature of dynamically loadable sub-agent modules, that lingered in an unfinished state at that time, and contributed our patches back to the `NET-SNMP` team.

As our Script-MIB agent stabilized over time we started to look closer at the manager side. Since the management of remote scripts means more than just reading a few isolated agent objects, we decided to develop not only a user interface application for managing remote scripts, but also a reusable manager library in between that supplies a Java API for high-level Script-MIB management applications hiding details of the underlying SNMP operations. The SNMP engine used by this package has not been built from scratch. Instead we used the existing freely available `JMGMT` SNMP class package written by Sven Dörr [11]. A minor bug in this package could be fixed easily and contributed back to the author because this package is also open source.

A GUI application has been built on top of this class package to allow users as well as developers to access and modify agents supporting the `DISMAN-SCRIPT-`

[1]At that point in time, the name of the toolkit was UCD-SNMP.

[2]Contact Sven Mertens <mertens@ibr.cs.tu-bs.de> for details on the porting effort.

MIB in a comfortable and efficient way. Such an application has been missed for a long time. This became obvious, when in August 2000 Ruben Marsman and Ron Sprenkels from the University of Twente released their ScriptMIBControl application [12] just a few days before we managed to integrate our "Script Mib UseR interFace" (Smurf) into our Jasmin distribution. This was our way of learning about the rule "Release early. Release often." [5]. If we or the people at Twente would have released earlier, it would have been easier to coordinate our ideas and work together.

During the practical experience we gained with the DISMAN management by delegation architecture we felt the need for a convenient way to allow scripts to export results in more sophisticated ways than through the smRunResult object provided by the DISMAN-SCRIPT-MIB. We decided to develop a toolkit that allows Java scripts in the Jasmin agent to act as AgentX sub-agents [13], so that they can instantiate arbitrary new MIB objects. Since our concept of Java AgentX sub-agent development is not limited to distributed management environments, we decided to release it as a separate package named JAX, which is also under the GPL [14, 15].

After building a working distributed management infrastructure, i.e., developing an agent and a manager application, we needed to gain more experience with real management tasks that are predestinated to be distributed by delegating management scripts. Accidentally, at the same time we had a labour shortage for system administration at our university institute, so that it made sense to spend some time to develop monitoring scripts that keep track of the health of some key network services and resources. This has been done in a structured way: A reusable package of a few Java classes supplies a basis to register and schedule a set of checks. On top of it several monitor scripts simply register their check classes and start the package's scheduler. Using this approach we evaluated our Jasmin environment from the application point of view and revealed some expected and some new advantages and disadvantages [16].

At this point in time we picked up another MIB developed by the IETF DISMAN working group. The DISMAN-SCHEDULE-MIB [17] which allows — among other actions — to control local scripts on a scheduling basis. This MIB has been implemented as a dynamically loadable sub-agent module similar to our DISMAN-SCRIPT-MIB implementation. During the implementation a few questions and comments on the RFC came up and have been sent back to the editors to clarify and enhance the specification.

Finally, in the end of 2000, a second runtime engine for the Tcl scripting language has been implemented according to the SMX protocol. Since the Tnm Tcl extension for network management [18] is also maintained at TU Braunschweig, we decided to integrate the SMX implementation into the Tnm Tcl package being part of the Scotty distribution [19].

3 Jasmin Software Components

We will now describe some technical aspects of the various open source software components produced within the Jasmin project so far and how they interact with other open source components. A scenario that involves all described components is shown in Figure 1.

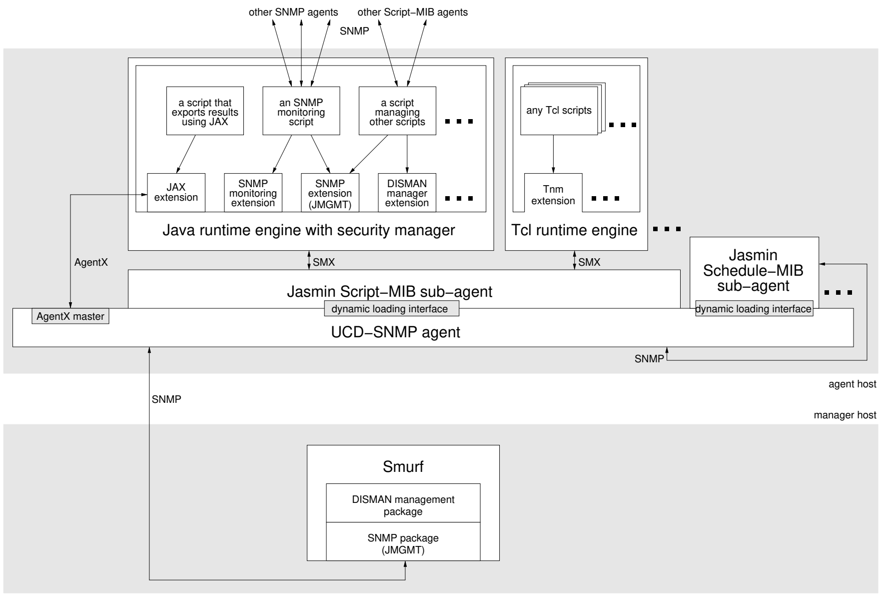

Figure 1: A scenario involving all Jasmin open source components.

The upper half of Figure 1 shows the software components that make up a Jasmin enabled SNMP agent. The Jasmin Script-MIB and Schedule-MIB sub-agents are based on the open source `NET-SNMP` toolkit. We make use of the dynamic loading interface since we did not want to tightly integrate the Jasmin sub-agents into the `NET-SNMP` distribution. The Jasmin Script-MIB sub-agent interacts with the Java runtime engine. Most of the Java management scripts we have developed so far again make use of the `JMGMT` open source SNMP class package. We decided to use `JMGMT` instead of other well known Java SNMP packages since `JMGMT` comes with full source and can be freely distributed and used.

The lower half of Figure 1 shows software components that realize a simple graphical user interface for delegating management scripts. It consists of a high-level Java package named `disman` for interacting with `DISMAN-SCRIPT-MIB` and `DISMAN-SCHEDULE-MIB` agents, which is again based on the `JMGMT` package. On top of the `disman` package sits the `Smurf` graphical user interface.

3.1 The Jasmin Script-MIB Agent

The Jasmin sub-agent implementation of the `DISMAN-SCRIPT-MIB` [4] has the architecture shown in Figure 2. Since the IETF Script-MIB standard supports aribitrary scripting languages, the core of the sub-agent talks to the runtime engines through the SMX protocol [8]. Currently, the Jasmin distribution contains a Java runtime engine. A Tcl engine has been implemented as part of the open source `Tnm` Tcl extension for network management [18]. An engine for Perl scripts has been developed independently at the University of the Sinos River's Valley in Brazil [9].

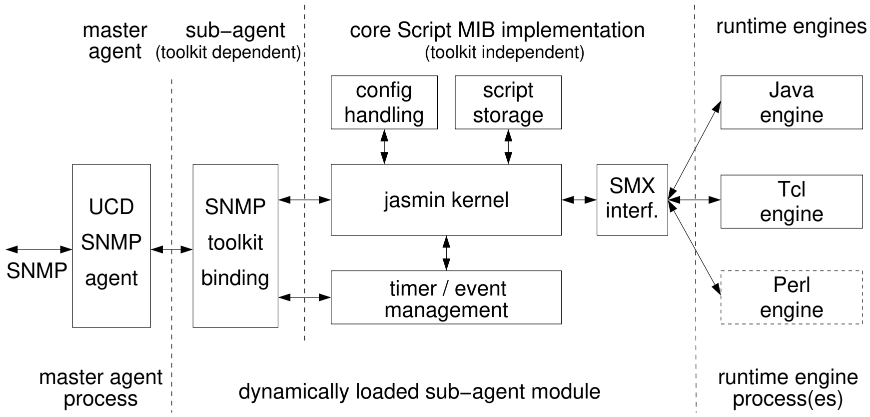

Figure 2: The Jasmin agent architecture.

The current SNMP part of the sub-agent is built with the `NET-SNMP` toolkit. The sub-agent is loaded into the master agent at runtime using the feature of dynamically loadable sub-agent modules. In order to pull scripts from script repositories, we make use of the `libwww` library, which is openly available from the World Wide Web Consortium (W3C) [20].

The Java runtime engine also implements the SMX protocol. It is written as a multi-threaded Java program that contains an adapted class loader and security manager. This allows to run multiple independent scripts in a single Java process and to support runtime security profiles based on the sandbox security model [21]. In a similar fashion, the Tcl runtime engine is capable to run multiple Safe-Tcl interpreters [22] supporting multiple security profiles within a single process.

3.2 The `disman` Java package

Using the `DISMAN-SCRIPT-MIB` facilities in a management application might require some effort. If for example a script to be executed is not already installed at the concerned network node, then a new row in the script table and a new row in the launch table have to be created by the application before the script can be started.

Creating these rows and filling them with appropriate values requires several SNMP requests and for each row a polling phase, until the row is ready to be used [4].

The sequence of actions required for installing a script is almost the same for any script and it should be provided as a function. The `disman` package was designed to provide this and other functions in a reusable Java class package, facilitating the general usage of the `DISMAN-SCRIPT-MIB` and the `DISMAN-SCHEDULE-MIB` in Java programs.

We implemented the package in Java, because we used Java for writing prototype management applications. The design can easily be ported to other object-oriented or procedural languages as far as they support exception handling. For languages without support for exceptions, the handling of errors and timeouts must be redesigned. However, this does not appear to require much effort.

When designing the `disman` package we tried to support as many kinds of management applications as possible. Therefore the package gives access to the MIBs on different levels of detail. At the lowest level, every single element of a MIB table can be accessed and modified (if possible). At a medium level, complete rows of a MIB table can be addressed. On the highest level, all details of the MIBs are hidden. Just schedules and scripts, their execution, their input, and their output are visible.

As mentioned before, the implementation was built on top of the open source SNMP stack by Sven Dörr [11], which also includes BER encoding.

3.3 A GUI Application to Manage Script-MIB Agents

The desire for a graphical user interface arose early in the Jasmin project, particularly when new people joined and when extensive testing was required. However, the Script Mib UseR interFace (`Smurf`) was among the last components to be developed. A screenshot is shown in Figure 3.

`Smurf` can handle multiple agent contexts (IP address, port, community string) at a time. A navigator (left hand side in Figure 3) is used to select a context, and within the context to select a MIB table. The selected table is displayed and changeable elements can be edited as in a spreadsheet application[3].

Higher functions, such as running a script and waiting for the result, or such as deleting a script and all related entries in different tables, are available at the top menu bar and at context menus. Context menus pop up when clicking a row of a table with the right mouse button. `Smurf` is a Java application making extensive use of the `disman` package.

Tests with unexperienced users proved that the GUI is intuitive and easy to use. Helpful additional features are the storage of recent inputs to dialog boxes and a built-in HTTP server which can be used to install scripts from local files[4].

[3]Currently, `Smurf` supports only the `DISMAN-SCRIPT-MIB`. We are working on support for the `DISMAN-SCHEDULE-MIB`.

[4]Our Script-MIB implementation supports pulling scripts from an HTTP server, but not pushing them from a management application via SNMP.

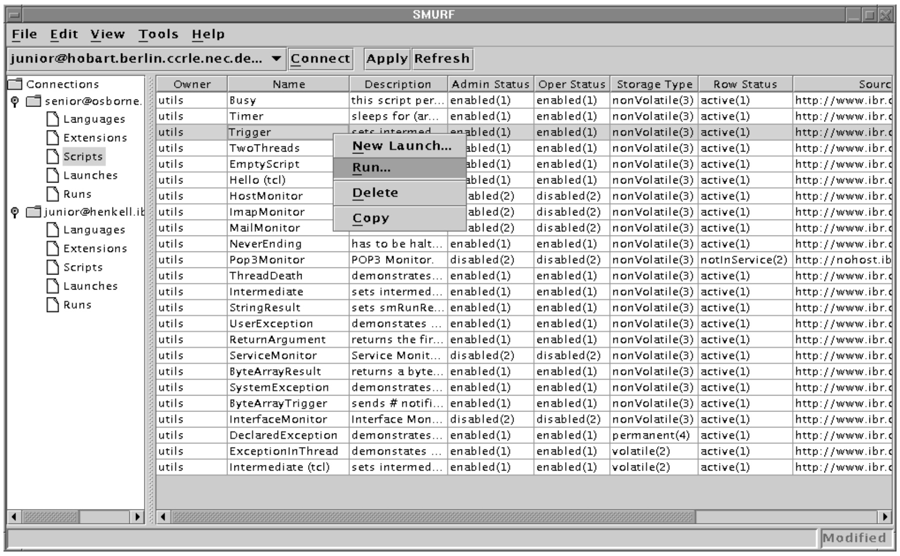

Figure 3: A screenshot of the graphical user interface of Smurf. The left frame shows the available agent contexts and tables. The right frame shows the current script table and a context menu on a selected script.

3.4 A Java Toolkit for AgentX Sub-Agent Development

The Java AgentX sub-agent toolkit `JAX` [14, 15] consists of two parts:

1. a Java package named `jax` that contains

   - classes that represent the building blocks of the AgentX architecture: connections, sessions, and registration,
   - 18 classes that represent the administrative and management processing AgentX PDUs,
   - classes that represent generic MIB information structures namely scalar groups, tables, and notifications,

2. a MIB compiler that generates high-level MIB-specific stub classes derived from the package's generic MIB information classes.

The fact that the SNMP specific parts are handled by the AgentX master agent and the AgentX protocol specific parts of the sub-agent are handled by the `jax` package allows for a high-level sub-agent development. The details of protocol handling are hidden from the programmer, who just has to implement the behavior of MIB objects by filling in method code of compiler-generated classes and to initialize the sub-agent from the instrumented application.

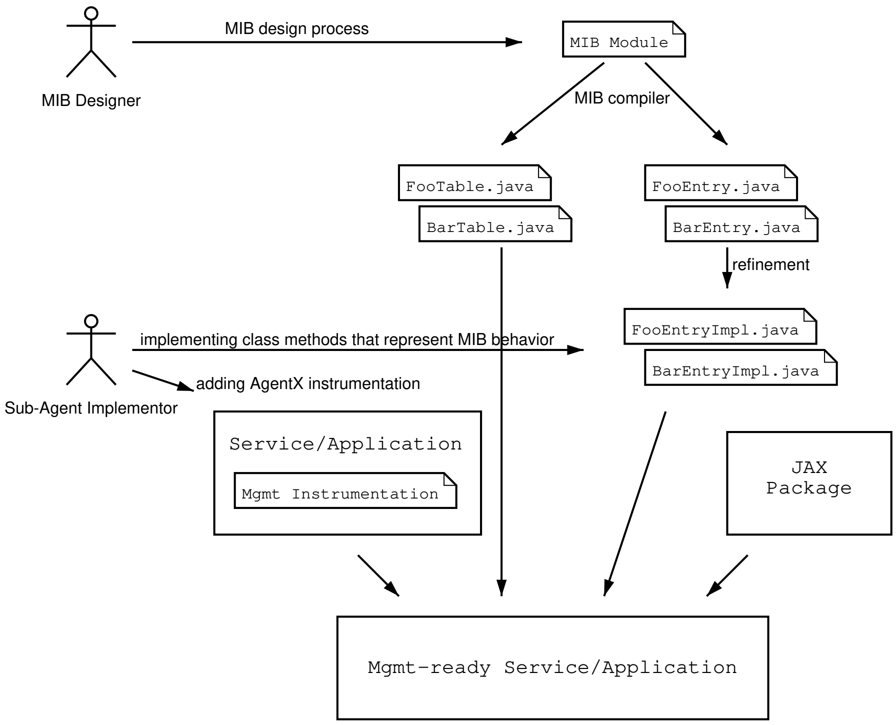

Figure 4: The `JAX` sub-agent development process.

The process of instrumenting an application is shown in Figure 4.

The MIB compiler has been built as an additional output driver of the `smidump` tool, which is part of the `libsmi` SMI C library distribution. Like Jasmin, `libsmi` is an ongoing open source software project which started in 1998 at the Technical University Braunschweig and was first released in June 1999 [23, 24]. It is currently being used by several other packages such as GxSNMP, Tcpdump, and in some commercial products.

3.5 Java Monitoring Scripts

A common use case of distributed scripts is monitoring. Monitoring tasks can be applied to different domains, hence multiple monitors have to be developed. To give all monitor scripts a unified behavior with respect to argument parsing, scheduling, and their handling of monitored SNMP targets, and to increase the scale of reusable monitor code, we developed a simple `snmpmonitor` class package. A UML class diagram of this package is shown in Figure 5.

Monitors that use this package work as follows: First, they instantiate an `Snmp-`

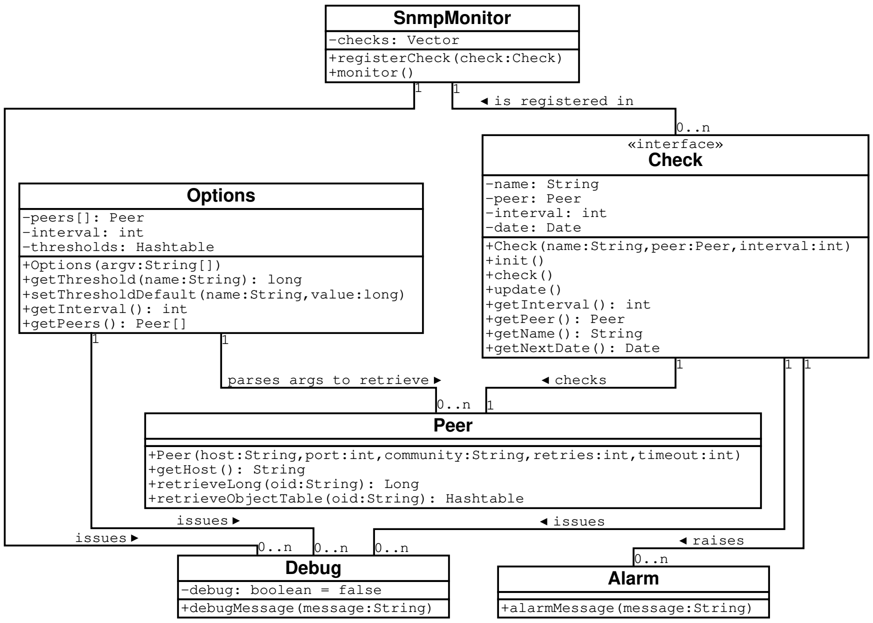

Figure 5: UML class diagram of the SNMP monitoring package.

Monitor object. Then with the help of the Options class they walk through the monitoring targets that have been passed to the monitor as arguments in order to register a set of check objects within the monitor that implement the Check interface. When all checks are registered, the monitor() method has to be called in order to start the scheduler. Each time when a check is called by the scheduler and it detects an anomaly, an alarm message is sent as a notification.

Based on this package we developed

- an interface monitor that detects changes of interface states, high interface utilization, and high error rates,
- a host monitor that detects high numbers of processes and user sessions on hosts, rebooted hosts, and full file-systems,
- a mail server monitor that detects high volumes of messages stored in the mail server queue and high rates of incoming messages,
- a remote service monitor that detects unavailable SMTP, POP3, FTP, HTTP, and NNTP servers, and
- a TCP service monitor that detects anomalies in the number of open RPC, FTP, SMTP, NNTP, and NFS connections.

3.6 The Schedule-MIB Agent

The Jasmin `DISMAN-SCHEDULE-MIB` [17] sub-agent has been implemented as a dynamically loadable module for the `NET-SNMP` agent, just as the Script-MIB sub-agent. This allows us to load both sub-agents into the same SNMP agent and to schedule script invocations based on calendar time. The Set requests issued as actions of the Schedule-MIB are passed to the local agent through usual SNMP protocol messages, instead of using an internal interface of the `NET-SNMP` agent. This decision has been made to keep the interfaces as standards compliant as possible and to keep the implementation simple by leaving the required access control to the agent.

4 Open Source Experiences

Our experiences from the Jasmin project during the past two years reflect in parts what various proponents of the open source community preach. These aspects are not limited to projects in the area of network and systems management:

- Releasing our own work helped us to receive more feedback of technical relevance, because qualified people have the chance to examine our code.

- The fact that a lot of other software packages are freely available helped us significantly. As an example, we could build on a powerful SNMP agent toolkit and we could re-use components for script retrieval via HTTP and FTP, and for SNMP and BER processing in Java.

- A lot of high quality development tools that we used are open source products, e.g., GCC, GNU make, automake, and autoconf.

- Using licensed software as part of your own work can be a pain if you intend to make your own work available.

- Early releases — or at least announced intentions — help to coordinate work and ideas and to reduce doubled work.

- Disclosing the own source code and specifications increases the developer's ambition to do things in a clean and proved way in the first place. This saves you work and pain in the long run.

- Not only program source code should be freely available. Open specifications are important as well to help getting a technology on track and to assure implementations' compatibility.

- A reasonably modularized software design makes components of your project reusable to other people and increases the advantages of open source software with respect to at least these components.

- These aspects make working on and with open source software projects more fun — or at least less pain.

In contrast to these advantageous facets of opening projects to the community we do not want to conceal some two edged aspects:

- Projects on emerging popular topics that are openly announced at a very early stage and that are not explicitly directed to follow certain goals and technologies can receive even too much input causing more controversial subjects than productive and goal-oriented work. However, this is more likely to happen in specification processes and larger open projects.

- Building on existent open source software, like we did with the `NET-SNMP` toolkit, the `JMGMT` class package and the `libwww` library, increases your project's dependency on other people. To ensure a clean setup the coordination with those projects has to work properly. In case of the Jasmin project, our `NET-SNMP` patches have been integrated with the current `NET-SNMP` source code which is publicly available through an anonymous CVS repository. Our patch for the `JMGMT` class package has also been contributed back to the author but not yet released. That's why we had to integrate this patch with our Jasmin release.

- A successful open project can cause a huge amount of traffic on accompanying mailing lists or suitable newsgroups. Much of this traffic can be categorized as "free online support" instead of technically valuable feedback. Serving these demands adequately can take lots of time. To reduce this effect, valuable documentation should be made available. However, from the user's point of view this efficient support is an encouraging argument for using software of living open source projects. And hopefully, some of these users will in turn improve the project.

- How lively and fruitful the discussions on different channels get varies. In case of the Jasmin project there is rather low traffic on the mailing list, although the number of more than 100 downloads of the Jasmin software during the first three months is quite a lot for such a specific project. In fact, besides the mailing list, we receive valuable feedback also from discussions on conferences and from private email.

5 Conclusions

This paper presented an overview over the open source components for distributed Internet management by delegation (MbD) that were developed within the Jasmin project. Some project history has been described to explain how the open source idea influenced the overall project planning. We also summarized some experiences we made that are specific to open source projects.

In summary, we do encourage people to experiment with open source software components and to make their own work openly available. In fact, we believe that a proposed technology which is backed up by an open source implementation has much more impact than a proposed technology which is “only” well documented, but not widely accessible.

Acknowledgments

Many people have contributed to the Jasmin project. We would like to thank Matthias Bolz and Sven Mertens for the implementation of the initial Jasmin agent. Sven Mertens later ported the initial Jasmin agent to the `NET-SNMP` toolkit. Sven Brandenburg and Sven Mertens worked on the implementation of the `disman` Java package. Sven Mertens also contributed major parts to the `JAX` implementation. The `Smurf` implementation which uses the `disman` Java package was done by Thiemo Schwarz and Raghuveer Singh. Cornelia Kappler and Torsten Klie worked on a test suite to evaluate the `disman` package and Script-MIB agents. Torsten Klie worked as well on the Schedule-MIB implementation and test suite. Finally, there are many more people who contributed to the project by providing valuable comments on specifications and our implementation. This feedback generally encouraged us to improve the software and specifications produced in the Jasmin project.

References

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

[2] IETF, http://www.ietf.org/html.charters/disman-charter.html. *DISMAN - The Distributed Management Working Group*.

[3] J. Schönwälder. Network Management by Delegation - From Research Prototypes Towards Standards. *Computer Networks and ISDN Systems*, 29(15):1843–1852, November 1997.

[4] D. Levi and J. Schönwälder. Definitions of Managed Objects for the Delegation of Management Scripts. RFC 2592, May 1999.

[5] E. S. Raymond. *The Cathedral and the Bazaar*. http://www.tuxedo.org/ esr/writings/cathedral-bazaar/, 1997-2000.

[6] E. S. Raymond. *The Cathedral and the Bazaar - Musings on Linux and Open Source by an Accidental Revolutionary*. O’Reilly, 1 edition, October 1999.

[7] TU Braunschweig, NEC C&C Research Laboratories, http://www.ibr.cs.tu-bs.de/projects/jasmin/. *Jasmin - A Script MIB Implementation*, 1999.

[8] J. Schönwälder and J. Quittek. Script MIB Extensibility Protocol Version 1.0. RFC 2593, May 1999.

[9] L. F. Balbinot and L. P. Gaspary. Towards Configuration Management of CoralReef-based Traffic Measurement Stations through the IETF Script MIB. In *Proc. 1st IEEE Workshop on IP-oriented Operations and Management (IPOM'2000)*, pages 129–136, September 2000.

[10] Wes Hardaker. *The NET-SNMP Project*. Sourceforge, http://net-snmp.sourceforge.net/.

[11] S. Dörr. *JMGMT: Free Java SNMP Stack with Servlet and Agent Framework*. http://i31www.ira.uka.de/ sd/mgmt/, 1999.

[12] R. Marsman and R. Sprenkels. *ScriptMIBControl*. University of Twente, http://www.simpleweb.org/software/packages/ScriptMibControl/, 2000.

[13] M. Daniele, B. Wijnen, M. Ellison, and D. Francisco. Agent Extensibility (AgentX) Protocol Version 1. RFC 2741, January 2000.

[14] F. Strauß. JAX - A Java AgentX Subagent Toolkit. Informatik Bericht 2000-06, Technical University Braunschweig, http://www.ibr.cs.tu-bs.de/vs/papers/jax-ib-2000-06.ps.gz, July 2000.

[15] F. Strauß, J. Schönwälder, and S. Mertens. JAX - A Java AgentX Subagent Toolkit. In *Proc. 1st IEEE Workshop on IP-oriented Operations & Management*, Cracow, September 2000.

[16] F. Strauß. Advantages and Disadvantages of the Script MIB Infrastructure. Informatik Bericht 2000-07, Technical University Braunschweig, http://www.ibr.cs.tu-bs.de/vs/papers/jasmin-eval-ib-2000-07.ps.gz, October 2000.

[17] D. Levi and J. Schönwälder. Definitions of Managed Objects for Scheduling Management Operations. RFC 2591, May 1999.

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

[19] J. Schönwälder. *Scotty - Tcl Extensions for Network Management Applications*. TU Braunschweig, 2000. http://wwwhome.cs.utwente.nl/ schoenw/scotty/.

[20] World Wide Web Consortium (W3C), http://www.w3.org/Library/. *Libwww - the W3C Protocol Library*, 2000.

[21] J. Schönwälder and J. Quittek. Secure Management By Delegation within the Internet Management Framework. In *Proc. 6th IFIP/IEEE International Symposium on Integrated Network Management*, pages 687–700, Boston, May 1999.

[22] J. Y. Levy, L. Demailly, J. K. Ousterhout, and B. Welch. The Safe-Tcl Security Model. In *Proc. USENIX Annual Technical Conference*, June 1998.

[23] TU Braunschweig, http://www.ibr.cs.tu-bs.de/projects/libsmi/. *Libsmi - A Library to Access SMI MIB Information*, 1999.

[24] J. Schönwälder and F. Strauß. Next Generation Structure of Management Information for the Internet. In *Proc. 10th IFIP/IEEE Workshop on Distributed Systems: Operations and Management*, pages 93–106. Springer Verlag, October 1999.