An IDE for NETCONF management applications

Paulo Tavares
DETI/IT
Universidade de Aveiro
Aveiro, Portugal
pcct@ua.pt

Pedro Gonçalves
ESTGA/IT
Universidade de Aveiro
Águeda, Portugal
pasg@ua.pt

José Luis Oliveira
DETI/IEETA
Universidade de Aveiro
Aveiro, Portugal
jlo@ua.pt

***Abstract*— The development of network and system management software typically requires data models definition, the creation of specific applications respecting the data model, and yet the implementation of communication interfaces. Skilled professionals usually perform such tasks in a predefined sequence and using different development solutions, but any error or lacks in the data model frequently force to repeat several time-consuming tasks. In this paper we present an integrated development framework that simplifies the construction of NETCONF management applications, from data model specification to deployment and evaluation. The framework is available at http://atnog.av.it.pt/~ptavares/ yangplugin.**

Keywords; component; Network management, NETCONF, YANG, SDK

I. INTRODUCTION

The development of networks and systems management solutions is usually a hard task, carried out by professionals with thorough knowledge of management landscape and the characteristics of the equipment. To help them, several models have been proposed over recent years. One such model is the Network Configuration Protocol (NETCONF) [1].

NETCONF is a network management protocol that was developed inside the IETF and uses W3C technologies such as XML, XML-RPC and SOAP. XML-based technologies have great support in many available XML handling libraries and development tools. Likewise, there is broad support for the development of distributed applications, based on web technologies (e.g. SOAP and REST web services, and Web 2.0 tools). These, associated with a configuration-driven design, are the main characteristics leading NETCONF to replace former solutions such as SNMP, COPS, or WBEM.

In any management framework, data modeling is a key component to formalize the description of data, messages, events and other management entities. In NETCONF, this goal is attained by YANG [RFC6020], a language that allows modeling configuration and state data, remote procedure calls, and notifications. The standardization process allowed mechanisms to be created for data translation into other formats, and the creation of tools that automatically generate documentation and perform automatic code generation [2]. There are also mechanisms that allow information import from other data models to YANG, as is the case of XSD and SMI [2].

This paper presents an integrated development platform for NETCONF based solutions in the form of a plugin for the Eclipse IDE. The work includes the creation of a YANG parser that was integrated within the IDE and enables the complete creation of management applications. It provides services such as importing data from existing Management Information Base (MIBs), editing and validating YANG modules, translating data between different formats and generating the skeleton for the management applications. The paper is organized as follows: section II gives a general overview of NETCONF; section III describes related work by other authors in this area; section IV describes the developed system; section V provides an evaluation of the system, and the paper ends with summarized conclusions.

II. TECHNOLOGY BACKGROUND

The NETCONF protocol [3] was standardized within the IETF as a management protocol that allows upload and retrieval of configuration data for distributed network systems. It is a client-server protocol that implements XML-encoded Remote Procedure Call (XML-RPC) over secure transports such as SSH [4], SOAP [5], BEEP [6], or TLS [7]. The peers use <*rpc*> requests and <*rpc-replay*> responses, regardless of the transport protocol framing. Positive responses include an <*ok*> element in the <*rpc-replay*>, whereas negative responses include an <*rpc-error*> element containing additional information describing the cause of error.

The protocol design was based on a modular architecture composed of four layers: a transport layer that implements the information transport, a RPC layer that implements the XML-RPC remote procedure call, an operations layer that implements NETCONF operations and a content layer containing the configuration data. NETCONF management information can be carried by a set of security enabled transport protocols [4-7]. RFC 4741 requires SSH transport implementation in order to promote interoperability. However, management elements are free to negotiate another secure communication protocol.

The protocol design followed a document-oriented approach, assuming that a device configuration is represented by a document that could be obtained, manipulated and copied. It supports multiple types of configuration datastores: the *startup* that stores the configuration which is delivered to the elements when they start; the *running* configuration representing the operational configuration of each element; and the *candidate* that allows configuration items to be handled in a temporary repository and changes made in that repository to be subsequently committed and applied to the *running* repository.

NETCONF base protocol [3] defines nine operations (Table 1). Additionally RFC 5717 defines a partial lock mechanism

for NETCONF RPC allowing, by means of a filtering XPath expression usage, partial configuration locks to be defined. NETCONF elements may additionally define new operations during the capability exchange process which, if common to both peers, can then be used during the session.

Table 1: NETCONF operations

Purpose	Operation	RFC
Information retrieval	<get>	4741
	<get-config>	
Configuration edition	<edit-config>	
Configuration copy	<copy-config>	
Configuration deletion	<delete-config>	
Datastore lock	<lock>	
Datastore unlock	<unlock>	
Session termination	<close-session>	
Session termination	<kill-session>	
Data lock	<partial-lock>	5717
Data unlock	<partial-unlock>	
Subscription creation	<create-subscription>	5177
Notification dispatch	<notification>	

The protocol implements filtering mechanisms in order to facilitate the handling of large configurations, allowing the protocol operations to affect only a part of the configuration. Although NETCONF implementations could support multiple filtering mechanisms, an XML tree-matching template scheme is mandatory. For instance, both *get* and *get-config* support a filter parameter as a means to select the subset of the configuration to be affected by the operation.

NETCONF provides monitoring support [8] implementing an event notification mechanism based on two operations: a) the client subscribes notifications from the agent using a <*create-subscription*> operation; b) when events occur, the NETCONF agent notifies the client using the <*notification*> operation. The subscription operation allows NETCONF clients to define a filter to be applied by the server in order to establish which events should be sent to the client, preventing the server from sending events the client is not interested in, and thus improving the overall scalability.

Basic NETCONF features can be extended. Each peer advertises the set of additional features that implements in the capability exchange process, during session establishment. When a NETCONF session is open, each peer sends a *hello* element that contains the list of capabilities it supports. The client discovers server operations and the operation parameters. RFC 4741 defines a set of optional features that may be announced in the capability process, but does not include the mandatory features of the protocol. The following list describes the capabilities and their identifier and defines operations peers must support; including the examples:

- *Writable-Running*: indicates the device supports direct configuration to the running datastore;
- *Candidate-Configuration*: indicates the device supports a candidate datastore allowing the configuration to be manipulated without direct impact on the device behavior. The capability defines operations to commit the configuration changes to the running datastore and to roll them back;
- *Confirmed-Commit*: allows the configuration changes to be reverted if a confirmed commit operation is not performed in predefined time-out;
- *Rollback-on-Error*: indicates the server supports the *rollback-on-error* option in the *edit-config* operation;
- *Validate*: indicates the server supports candidate configuration validation;
- *Distinct-Startup*: the server supports distinct startup and running configurations, in the sense that operations over the running configuration have no effect on the startup configuration;
- *URL*: the peer supports a URL as the source and target arguments;
- *XPath*: indicates the peer supports XPath expressions in the <*filter*> element.

The *content layer* of NETCONF architecture includes the management data model that is completely independent of the protocol.

A. YANG

YANG is a data modeling language [9] that was standardized by the IETF for NETCONF. The language has an equivalent XML-based format, designated *YANG Independent Notation* (YIN), which facilitates automatic machine processing. The translation between YANG and YIN is lossless, i.e. YANG and YIN formats contain similar information using different syntax, so that a translated model can be translated back to its initial format. The YANG data definition is carried out in modules, which can be subdivided into sub-modules sharing the same XML namespace. The sub-modules are visible only within the respective module and the data definitions made within each module are visible by importing the module. YANG data elements are stored in leaf elements, which are placed inside the XML document tree. The leaf elements can be grouped into arrays of simple elements through the *leaf-list* statement. Lists can also be created with *leaf-lists* and *containers*, an intermediate containing statement. The *grouping* statement creates a new complex type that can contain an arbitrary hierarchy of *containers*, *lists*, *leafs* and *leaf-list* elements. The *augment* statement allows a module to extend an imported module and thereby make extensions to the initial XML configuration tree. This is essential for manufacturers to develop vendor-specific configuration extensions in order to support their equipment-specific details.

The language supports two types of optional constraints that NETCONF servers need to validate: the *must*, defining conditions that must be met by a valid NETCONF

configuration; and the *when*, defining when augmentation can be added to a configuration node element. YANG constraints are expressed in the form of an XPATH expression, which allows the configuration elements to be grouped under a common name. The management stations announce the optional data model components as NETCONF capabilities, when the NETCONF session is initiated. The definition of optional components is conducted by management stations, through the *deviation* statement, after establishing the NETCONF session.

III. RELATED WORK

A. NETCONF implementations

NETCONF-based management technology has been receiving much attention both from industry and academics. YENCA [10] was the first NETCONF implementation and was released in 2004 by LORIA-INRIA laboratories. It consists of an agent implemented in C language and a Java manager. It provides management support of the network interfaces and routing tables and implements communication between the agent and manager through a non-standard SOAP over TCP interface. Meanwhile, *LORIA-INRIA* improved YENCA providing a management platform with the name *EnSuite*, including a python NETCONF agent (*YENCAP*) and a Web based manager, supporting NETCONF over SSH communication. *EnSuite* platform supports YANG modules, but does not send nor is able to receive NETCONF notifications.

Another solution, *Netopeer* [11], is an open-source implementation created as part of *CESNET librouter* project, as a means to perform the configuration of a network probe named *FlowMon*. It was developed in C language and fully implements RFC 4741, performing equipment management based on a NETCONF over SSH communication interface. The manager includes a Web configuration frontend and a command-line interface. *Netopeer* allows probe configuration in both startup and running datastores, performing the complete information transfer before any configuration edition. It follows a RELAX-NG data model, since its development was previous to YANG language standardization by IETF [8].

Yuma tools [2] is a complete Software Development Kit (SDK) developed by the NETCONF Central Inc. that includes a NETCONF daemon (*netconfd*), a command-line client (*yangcli*), several YANG module manipulation tools and very complete documentation. It supports complete solution deployment since it includes tools that translate the MIB files into YANG modules, which generate C language program skeletons and tools that create project documentation.

Yuma generated applications make use of the NETCONF daemon which loads the developed code as a module for manipulation of user-defined objects. Because of its modular design, the client was previously created allowing manipulation of user-defined objects without any recompilation process. The communication is implemented by means of a NETCONF over SSH interface and allows the complete list of NETCONF operations, as defined by the NETCONF RFCs [3, 8].

Netconf4Android [12] is a NETCONF client API for Android that was developed in Java and uses NETCONF over SSH transport. The project is highly integrated with Yuma SDK and supports YANG based modules [9]. Plans to perform automatic skeleton of the distributed application are presented in the initiative web page.

ncclient [13] is a python library that facilitates client-side scripting and application development around the NETCONF protocol. It implements NETCONF over SSH transport and it supports all the capabilities and operations defined in RFC 4741. The API is integrated with a NETCONF manager developed in python, and is distributed as open source under an Apache license.

A number of proprietary implementations of NETCONF extend the open-source management solutions offer described above. Tail-f commercially provides a NETCONF-based management solution that includes an agent and a manager with a dual interface (a Web based and command-line). It also includes a SNMP agent that integrates with the management datastore. Additionally, the NETCONF manager library is available for the purpose of integration with the element management systems. Other examples of NETCONF commercial solutions include *embeddedMIND* from *Silicon&Software Systems*, *XML Based Device Management* from *Wipro*, *Applied Informatics* NETCONF support on *POCO framework*, and *XMS (eXtensible Management System)* from *6WIND*.

Beyond commercial implementations, there are also some implementations that are not available to the management community. Furthermore, and without providing their implementations publicly, the leading network equipment manufacturers have been providing NETCONF support for their equipment, examples being the *Juniper JUNOS*, and *Cisco IOS*. According to tests conducted by Iyad Tumar et al. [14] the implementations, despite presenting some bugs, generally meet NETCONF specification from RFC 4741 and RFC 4742.

Table 2 resumes the main characteristics of the open-source NETCONF implementations.

B. On the use of NETCONF

Several authors have developed and evaluated NETCONF solutions in recent years. Y. Zhichao et al. [15] made a comparison between XPATH subtree filtering mechanisms for NETCONF repositories. They implemented both information-filtering mechanisms in their management platform and did some tests. They found that the search time for information filtering based on XPATH grew more rapidly than the subtree filtering, and that the performance difference between the two mechanisms worsened with increased repository size.

Table 2: Open source NETCONF implementations

Product	Language	Standard	Developer	Completeness	Year
Yenca	C	SOAP over TCP	LORIA-INRIA	Java Manager, C Client	2004
EnSuite	Phyton	RFC 4742	LORIA-INRIA	Ensuite SDK includes Yenca-Manager and YENCAP	2005
Yuma	C	RFC 4742 RFC 5277	Netconf Central, Inc.	Complete SDK, YANG tools, application skeleton creation and documentation generation	2009
Netconf4Android	Java	RFC 4742	Giuseppe Palmeri	Client API for Android	2009
ncClient	Python	RFC 4742	Shikhar Bhushan	Client API, and NETCONF manager	2010
Netopeer	C	RFC 4742 RFC 5277	CESNET	API	2004

H. Xu et al. performed a comparison between YANG and XML Schema as management data modeling languages [16]. They defined an evaluation platform consisting of several criteria including interoperability, readability, extensibility and security considerations. Although they took into account features which have since been defined in YANG standard [9], they concluded that YANG would be more appropriate to NETCONF standardization process as a management technology. On the other hand, C. Huiyang et al. [17] performed a data modeling language analysis, where they compared XML Schema, Relax NG and YANG. They compared the three alternatives in terms of the language structure, its expressiveness, readability and interoperability. They also analyzed the languages' efficiency, conducting experiments encoding a common data model and checking their efficiency in terms of number of rows and number of characters in the resulting code. The study concluded that YANG performed globally better than the alternatives.

After YANG language standardization, various YANG editing tools were made available, and several YANG modules were created for a wide range of application scenarios. For instance, *NETCONF Central Inc*. provides a large number of YANG modules on its web page [18], including the translation of most SNMP MIBs published so far. *pyang* is a YANG validator and converter developed in python that conforms to YANG RFCs. It converts YANG documents to YIN and vice-versa, converts YANG to WSDL and XSD and creates UML documentation based on YANG modules. Additionally it includes a plugin framework that performs code generation. *Jyang* [19] is a yang parser developed in java by Emmanuel Nataf from *Madynes Inria* which is distributed as open source. It consists of a command-line application that parses yang modules and sub-modules and performs the translation from YANG to YIN. The distribution includes a documented API that could be used programmatically.

IV. NETCONF MANAGEMENT SDK

Our proposal consisted of a development tool for the creation of NETCONF management applications. The development was carried out as an Eclipse IDE plugin with an integrated set of existing YANG processing tools. This section documents the development process describing the requirements initially defined and the developed software, identifying the tools that were integrated as well as the steps to be followed during the process of management application creation.

A. Overall picture

NETCONF is a network management technology that has been standardized and can replace SNMP technology. SNMP has broad support and there are lots of MIBs for networking equipment that have to be translated into YANG modules. Furthermore, thanks to the attention paid by industry and academia, there are already several tools to process and translate data models to YANG format. Consequently, the requirements defined to the editor were: it would be an integrated environment allowing complete development; it would implement basic editor features, with YANG syntax highlighting, validation and error report; it would import a MIB to a YANG module; it would translate YANG modules to YIN; it would translate YANG module to XSD; and would create the application skeleton.

Eclipse is an IDE developed in java that implements editing facilities of various programming languages as modules. The IDE includes an extensible plug-in system and offers excellent support, easing its extension. Eclipse WTP provides tools for developing standard Java web applications and Java EE applications. Eclipse WTP simplifies the creation of these web artifacts and provides runtime environments in which these artifacts can be deployed, started and debugged.

The activity diagram of Figure 1 illustrates the complete flow of actions that can be performed in our editor. The development process can be started by importing an existing MIB and the corresponding translation in a YANG module, or by developing a YANG module from scratch. The translation

from SMI to YANG is performed by an application named *smidump* that was imbedded in our plugin. *Smidump* is a Unix application that allows dumping of the contents of a single MIB or PIB module or a collection of modules to *stdout* in a selectable output format.

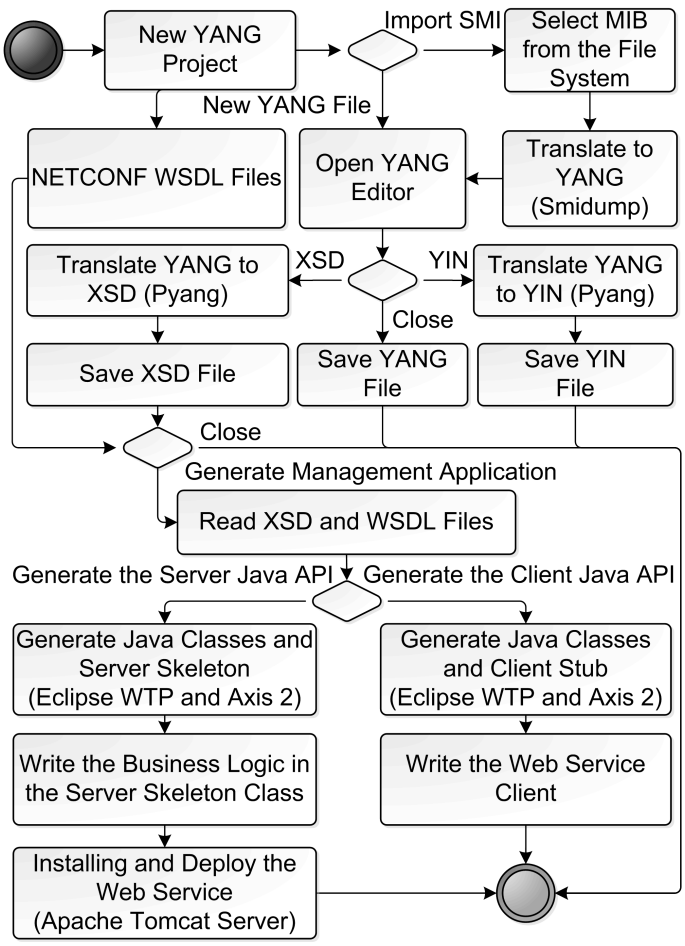

Figure 1: Development Activity Diagram

Our editor allows the exportation of YANG modules, since the modules were syntactically validated. It supports the transformation of YANG modules into XSD, and the translation of YANG models to YIN. To do that, we embedded the *pyang* application [20] in our YANG plugin. The editing process can be interrupted and resumed later by saving the developed data model in any of the supported formats.

The editor also allows generation of the management applications skeleton, creating the SOAP communication stubs, following the process described in [21]. The implementation uses Axis2, a popular open source Web service framework. Axis2 uses a Stax based XML parsing and internally uses the Axiom object model to represent the XML message. Based on this object model Axis2 provides an adapter, which is JAX-WS compliant. The application uses the description of the web service in the WSDL excerpt, from the RFC 4743, as a way to know the messages that have to be created in the web service. Additionally, it uses the YANG module description from the XSD in order to obtain the data model to use in management applications and NETCONF operations.

The editor allows the creation of both the NETCONF server application and the client application: in the case of the NETCONF server creation, after creating the classes that implement the stubs using Eclipse WTP and Axis2, the web service is installed and deployed via Apache tomcat server [22]; in the client code generation process the RPC stub classes are generated and then the Web service client created. The Axis library acts as a SOAP API, implementing communication between NETCONF client and server.

B. *YANG edition facilities*

The YANG edition support was based on Xtext [23], a language development framework that allows the creation of compilers and interpreters and fully integrates with eclipse, allowing the sophisticated IDE features like syntax highlighting and code completion.

The main component of an Xtext project is a file named grammar, where the language is described using expressions with a syntax similar to *Extended Backus-Naur* format. Figure 2 illustrates the grammar specified for the YANG editor. The code defines that each type statement begins with the keyword 'type' followed by one of the YANG built-in types or a type that can be defined by a *Typedef* statement. The statement then can end with a ';' or block of the statements that are defined in the *TypeSubStatement* Rule.

A unique feature of Xtext is the ability to declare crosslinks in the grammar. In traditional compiler construction the crosslinks are not established during the parsing process, but in a later linking phase. The same happens in the Xtext, but is allowed to specify crosslink information in the grammar and then the linker uses this information. The cross-references are specified by square brackets, as illustrated in the type statement rule, which allows the editor to trigger a search mechanism to search for a specific element in the local file or even a file in the workspace that has been imported.

```
TypeStatement:
    'type' (type=BuiltInType
    | (pre=STRINGARG':')? importtype=[TypedefStatement])
    (';'
    |'{' (typesubstatements+=TypeSubStatement)* '}');
TypeSubStatement:
    (BitStatement
    |FractionDigitsStatement
    |DefaultStatement
    |BaseStatement
    |EnumStatement
    |LengthStatement
    |PathStatement
    |PatternStatement
    |RangeStatement
    |RequireInstanceStatement
    |TypeStatement
    |UnknownStatement)
    ;
```

Figure 2: Grammar code for the YANG Type statement

Besides the grammar, it is possible to write constraints for arguments and their statements to complete and improve the parser, and to add custom error or warning messages detailing the situation. For instance, the code presented in Figure 3 shows how the accuracy of the YANG Revision Statement argument format was checked. If this check fails, the editor shows the corresponding error. The *@Check* annotation added

allows the method to be invoked automatically, when validation takes place.

```
@Check
public void checkRevision(RevisionStatement revision){
    String date=revision.getDate();
    if(!date.matches("(\")?[0-9][0-9][0-9][0-9]-[0-9][0-9]-[0-9][0-9](\")?"))
        error("The date must be in form '4DIGIT-2DIGIT-2DIGIT' "
                ,YangPackage.REVISION_STATEMENT__DATE);
}
```

Figure 3: Check method for the YANG Revision Statement

After validating its language model it is possible to generate an application that can be executed as an editor to be used within the Eclipse. Xtext deducts automatically, from the written language model, basic editing features like the parser, syntax highlighting, outline, folding, code completion, content assist, automatic formatting and others. All the features can be modified, complemented or enhanced programmatically by editing the remaining files in the Xtext project.

The features implemented in our YANG editor (Figure 4) include: code parsing and syntax highlighting (6), code outline (2), content assist (3), error detection, annotation message (5) and code folding (3).

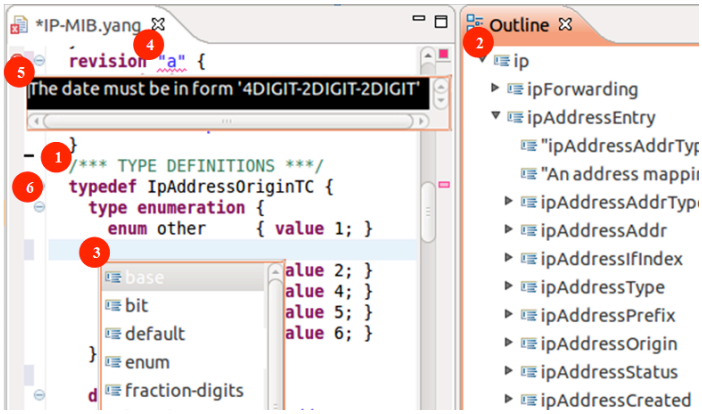

Figure 4: Yang plugin editor

C. *NETCONF application development process*

Once the YANG modules are validated, the editor allows code generation of the application skeleton, based on the XSD translation of the specified data model. The process includes three files: *netconf-soap_1.0.wsdl* describing the SOAP messages implemented by the service, *myNetconfService.wsdl* defining the service localization and *netconf.xsd* extracted from RFC 4741 that defines the NETCONF basic types.

The server creation process requires the user to create a new *Web Service Project* with the Tomcat Server as the runtime server and the Apache Axis2 as the Web service runtime creation tool. The client can be generated in a similar way, but choosing a *Web Service Client Project*. There are two alternatives for implementing data binding in the skeleton creation process: *XMLBeans* data binding and *Axis2 Data Binding* (ADB). The former alternative provides better XML parsing and fewer errors in creating the skeleton.

The code generation process creates a set of Java classes that represent the NETCONF operations as well as the types defined in the data model specified in YANG module. The generated classes, both in the client and in the server, are automatically grouped in a new package.

Once the code generation procedure is executed, the skeleton of the management applications is created. The generated classes completely implement communication between NETCONF client and server without any additional programmer action. This code needs to be complemented with the instructions that determine the behavior of the NETCONF server and agent.

V. RESULT ANALYSIS

This section describes the functional tests carried out on the developed software as well the test results. In order to test all the features offered by the editor, we used part of a well-known MIB and performed the translation to YANG. Then we translated YANG module to XSD and performed the creation of a NETCONF management application for that module. In tests we used the excerpt from the IP-MIB that describes the *ipAddressEntry* structure. The choice of this subset was based on the fact that it belongs to a well-known MIB and its information is often used in the management operations of network elements.

The MIB module importing process to YANG created a module (Figure 5) with the same name and a new namespace. The MIB subset structure was translated from SMI to YANG and a new YANG module was created where each of its elements is described using YANG syntax.

```
list ipAddressEntry {
  key "ipAddressAddrType";
  description "An address mapping for a interface.";
  leaf ipAddressAddrType {
    type inet-address:InetAddressType;
         config true; [Description removed]
  }
  leaf ipAddressAddr {
    type inet-address:InetAddress;
    config true; [Description removed]
  }
  leaf ipAddressIfIndex {
    type if-mib:InterfaceIndex;
    config true; [Description removed]
  }
  leaf ipAddressType {
    type enumeration {
    enum unicast   { value 1; }
    enum anycast   { value 2; }
    enum broadcast { value 3; }
    }
    config true; [Description removed]
  }
  leaf ipAddressPrefix {
    type smiv2:RowPointer;
    config false; [Description removed]
  }
  leaf ipAddressOrigin {
    type ip-mib:IpAddressOriginTC;
    config false; [Description removed]
  }
  leaf ipAddressStatus {
    type ip-mib:IpAddressStatusTC;
    config true;  [Description removed]
  }
  leaf ipAddressCreated {
    type yang:timestamp;
    config false; [Description removed]
  }
  leaf ipAddressLastChanged {
    type yang:timestamp;
    config false; [Description removed]
  }
  leaf ipAddressRowStatus {
```

```
    type smiv2:RowStatus;
    config true; [Description removed]
  }
  leaf ipAddressStorageType {
    type smiv2:StorageType;
    config true; [Description removed]
  }
}
```

Figure 5: YANG code

Since this YANG module references other two modules (yang-smi and INET-ADDRESS-MIB), they were downloaded from the Netconf Central modules database [18] and translated into XSD format.

Once the YANG module was translated to XSD, the process of distributed application generation code was run. Several Java classes were automatically generated: a stub class that implements the communication interface of the distributed application; a class for each of the operations described in the XSD file; a class for each data type defined in the XSD file; and a class for each data type belonging to YANG base types defined in netconf.xsd.

Besides, the generated classes were automatically organized into packages: a package was created containing the classes of communication interface for the client and another package for the server. The class diagram of Figure 6 illustrates the classes that were generated in the stub creation process.

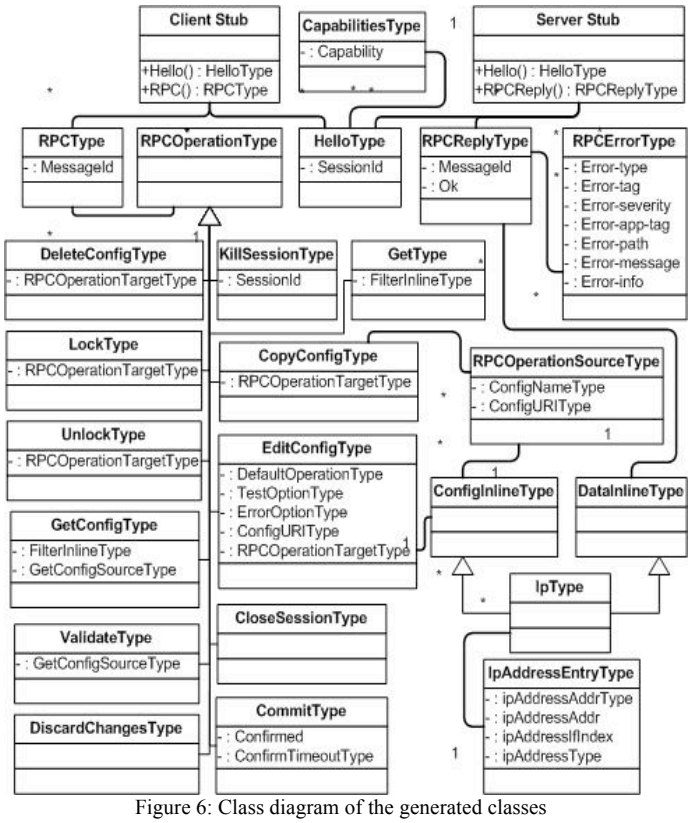

Figure 6: Class diagram of the generated classes

The code generation process is responsible for creating the communication interfaces for the client and server applications, which includes methods to establish and close the session, and to format, to send and to receive the messages. In order to test the NMS it is necessary to finalize the NETCONF client creating a java main class, since the generated code just implements the communication interface. Both applications have to be completed with the code that implements their behavior. The client application needs to be coded in order to define its interaction with the server; the server application code should be completed in order to implement its instrumentation.

In our experiments we implemented a dummy application pair that establishes a NETCONF session, changes a very minimal agent configuration and finally closes the session. Our applications were run, and since we implemented a SOAP transport, we could capture the network traffic. Figure 7 shows the SOAP envelop corresponding to an *edit-config* operation.

```
<soapenv:Envelope>xmlns:soapenv="http://schemas.xmls
oap.org/soap/envelope/"
   <soapenv:Body>
      <urn:rpc>message-id="101"
 xmlns:urn="urn:ietf:params:xml:ns:netconf:base:1.0"
      <urn:rpcOperation>xsi:type="urn:editConfigType
      "xmlns:xsi="http://www.w3.org/2001/XMLSchema-
      instance"
         <urn:config>
            <urn1:ip>xmlns:urn1=
      "urn:ietf:params:xml:ns:yang:smiv2:IP-MIB"
               <urn1:ipAddressEntry>
               192.168.1.1
               </urn1:ipAddressEntry>
            </urn1:ip>
         </urn:config>
      </urn:rpcOperation>
      </urn:rpc>
   </soapenv:Body>
</soapenv:Envelope>
```

Figure 7: SOAP message

VI. CONCLUSIONS

This paper describes implementation of an IDE for the development of NETCONF-based management applications. The editor allows the complete development cycle, allowing performance of tasks ranging from data model definition to debugging of the finalized application. The developed solution integrates in an Eclipse *plugin* a YANG editor and a set of YANG handling tools that allows the import of existing data models from an SNMP MIB, editing them as a YANG module, to transpose them to YIN or XSD formats, to generate module documentation and the skeleton of distributed applications.

The editor includes all the edition facilities available in Eclipse IDE, such as the syntax highly and syntactical error detection, and it reuses all the facilities to interact with the editor (menus, shortcuts, etc.), allowing reuse of the eclipse IDE use knowledge and thus minimizing the application learning curve. The Apache Axis2 tool has proved to be of great value because it alone can handle XML parsing as well as SOAP messaging, allowing the user to concentrate on programming the logic of the NMS.

Although completely functional, the applications generated with our IDE do not respect NETCONF standards since they will not support the SSH transport protocol mapping, defined as mandatory in RFC 4741. Another limitation of the implementation is the lack of authentication, which according to the draft [24], will become mandatory for NETCONF sessions.

Monitoring support was not considered in our application, despite the simplicity and reduced number of operations involved. The difficulty of developing it is related to the fact that during the monitoring process NETCONF agent and manager exchange their roles in accordance with the client-server model. Monitoring support was considered as future work and we decided to develop it in the next version of software.

The distributed application development process does not include the capability exchange support, forcing the programmer to develop code to do so. Since the optional operations that a NETCONF peer supports are described in YANG module, as well as associated data, development of the capability announcement can be automated. It could be included in the process of automatic code generation of the distributed application and was considered as future work to be developed in future versions of software.

Although the specialized editors for management data definition are not normally used, the addition of advanced edition features (syntax highly, code folding, ...) usually included in more recent IDE, makes the code editing process of the YANG modules much easier. Besides, SMI data models' import features simplify the application creation with legacy data models. The integration of data model specification functions with software development tools and automatic code generation allows the management application creation process to be carried out in one application, saving time and avoiding the data exporting procedures between several editor applications.

Being an integrated development environment that allows the complete development cycle in a single application, and automating most of the development tasks, this system can also be used for administrators to design and prototype NETCONF solutions, testing YANG data models and NETCONF operations before they have to deploy them in a real network system.

REFERENCES

[1] J. Schönwälder, M. Björklund, and P. Shafer, "Network configuration management using NETCONF and YANG", Communications Magazine, IEEE, vol. 48, pp. 166-173, 2010.

[2] "Yuma - YANG-based Unified Modular Automation Tools", http://sourceforge.net/projects/yuma/, 2011-02-21

[3] R. Enns, "NETCONF Configuration Protocol", RFC 4741, 2006.

[4] M. Wasserman and T. Goddard, "Using the NETCONF Configuration Protocol over Secure Shell (SSH)", RFC 4742, 2006.

[5] T. Goddard, "Using NETCONF over the Simple Object Access Protocol (SOAP)", RFC 4743, 2006.

[6] E. Lear and K. Crozier, "Using the NETCONF Protocol over the Blocks Extensible Exchange Protocol (BEEP)", RFC 4744, 2006.

[7] M. Badra, "NETCONF over Transport Layer Security (TLS)", RFC 5539, 2009.

[8] S. Chisholm and H. Trevino, "NETCONF Event Notifications", RFC 5277, 2008.

[9] M. Bjorklund, "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, 2008.

[10] B. Zores, R. State, and O. Festor, "YENCA", http://sourceforge.net/projects/yenca/, 2011-03-22

[11] Radek Krejcí, L. Lhotka, P. Celeda, and P. Špringl, "Secure Remote Configuration of Network Devices", presented at the CESNET 2008, Prague, Czech Republic, 2008.

[12] G. Palmeri, "NETCONF4Android project", http://code.google.com/p/netconf4android/, 2011-03-22

[13] S. Bhushan, H. M. Tran, and J. Schönwälder, "NCClient: A Python Library for NETCONF Client Applications", presented at the Proceedings of the 9th IEEE International Workshop on IP Operations and Management, Venice, Italy, 2009.

[14] I. Tumar, H. M. Tran, and J. Schönwälder, "NETCONF Interoperability Testing", presented at the Proceedings of the 3rd International Conference on Autonomous Infrastructure, Management and Security: Scalability of Networks and Services, Enschede, The Netherlands, 2009.

[15] Y. Zhichao, Z. Bin, L. Guohui, L. Yan, and G. Xuesong, "The implementation and comparison analysis of subtree filtering and Xpath capability in NETCONF", in Network Infrastructure and Digital Content, 2009. IC-NIDC 2009. IEEE International Conference on, 2009, pp. 921-925.

[16] H. Xu and D. Xiao, "Considerations on NETCONF-Based Data Modeling", in Challenges for Next Generation Network Operations and Service Management. vol. 5297, Y. Ma, D. Choi, and S. Ata, Eds., ed: Springer Berlin / Heidelberg, 2008, pp. 167-176.

[17] C. Huiyang, Z. Bin, L. Guohui, G. Xuesong, and L. Yan, "Contrast Analysis of NETCONF Modeling Languages: XML Schema, Relax NG and YANG", in Communication Software and Networks, 2009. ICCSN '09. International Conference on, 2009, pp. 322-326.

[18] "NETCONF Central Inc. Web page", http://www.netconfcentral.org/, 2011-03-22

[19] E. Nataf and O. Festor, "jYang : A YANG parser in java", Computing Research Repository, August 2009 2009.

[20] "Pyang", http://code.google.com/p/pyang/, 2011-03-20

[21] I. Tomoyuki, Y. Atarashi, H. Kimura, M. Kitani, and H. Okita, "Experience of Implementing NETCONF over SOAP", RFC 5381, 2008.

[22] "Apache Axis 2", http://axis.apache.org/axis2/java/core/, 2011-03-20

[23] "Xtext - Language Development Framework", http://www.eclipse.org/Xtext/, 2011-03-22

[24] A. Bierman and M. Bjorklund, "*Network Configuration Protocol Access Control Model",* draft-ietf-netconf-access-control-03, 2011.