YANG2UML: Bijective Transformation and Simplification of YANG to UML

Mario Golling*, Robert Koch*, Peter Hillmann*, Rick Hofstede† and Frank Tietze*
*Universität der Bundeswehr München, Department of Computer Science, 85579 Neubiberg, Germany
Email: {mario.golling, robert.koch, peter.hillmann, frank.tietze}@unibw.de
†Design and Analysis of Communication Systems (DACS), University of Twente, Enschede, The Netherlands
Email: r.j.hofstede@utwente.nl

***Abstract*—Specifically designed to exchange configuration information from a management platform to network components, the XML-based NETCONF protocol has become widely used. In combination with NETCONF, YANG is the corresponding protocol that defines the associated data structures, supporting virtually all network configuration protocols. YANG itself is a semantically rich language, which – in order to facilitate familiarization with the relevant subject – is often visualized using UML to involve other experts or developers and to support them by their daily work (writing applications which make use of YANG/NETCONF). To support this process, this paper presents an novel approach to optimize and simplify YANG data models, as current solutions tend to produce very complex UML diagrams. Therefore, we have (i) defined a bidirectional mapping of YANG to UML, (ii) developed a strategy to reduce the numbers of objects, and (iii) created a tool that renders the created UML diagrams, closing the gap between technically improved data models and their human readability.**

Keywords*—*YANG, NETCONF, UML, Transformation, Bijective Mapping.

I. Introduction

YANG is an extensible NETCONF data modeling language able to model configuration data, state data, operations, and notifications [1]. As such, YANG itself is a semantically rich language supporting virtually all network configuration protocols. However, when building applications that make use of YANG, developers often would like to have YANG visualized using UML diagrams for a better understanding [2]. The corresponding UML version has to be processable by applications as well. Trigerstripe is such a tool to develop APIs manually by hand; this tool is used by Cisco's network management system PRIME, for example. As such, a reverse mapping (from UML representation back to YANG) must always be possible in an automated fashion (i.e. without human interaction) in a unique way (*bijective function*). Using a variety of transformation rules and by integrating the user (who is often necessary to establish the overall context), this paper presents a bidirectionally mappable representation model that uses the well-established YANG data model and creates an UML presentation. Individual classes are reduced to (i) lower complexity and to (ii) improve readability. Therefore, we have defined a mapping of YANG to UML and developed a new transformation engine called YANG2UML for the automatic creation of compact object models that also renders the created UML diagrams.

The remainder of this paper is structured as follows. Section II gives an overview of related work. Section III discusses the YANG specification and our transformation policy. Thereafter, Section IV briefly explaines the corresponding implementation, before Section V concludes this paper.

II. Related Work

PYANG [2] is the most important work in this area up to now, as it also transforms the data model of YANG directly to UML. The first step of PYANG comprises a validation of YANG, before, within the next step, YANG's compact Structure of Management Information (SMI) like syntax is translated into an XML version. The result is called YIN (YANG Independent Notation) and thus is an XML version of YANG (lossless roundtrip conversion) [1]. The equivalent representation of YANG information in an XML notation allows developers to use existing XML tools and tools for data filtering and validation, and thus to reduce the programming effort. However, since there is no reduction performed in the number of objects, the corresponding diagrams are difficult for the people to read. See [3] for more details regarding the corresponding data model.

III. Concept

A. Top-Down vs. Bottom-Up Processing

As the hierarchies of YANG statements are arranged as trees, there are two possibilities to process them: Top-down respectively bottom-up. As each strategy influences the result and the quality of the data reduction process, both strategies are applied and processed in parallel to combine the advantages of the respective method and to detect problems and necessities for manual decisions, e.g., in case of discrepancies between the two methods (see Fig. 1). First, the classification of elements is done by the bottom-up approach, guaranteeing an unequivocal classification of elements. While moving upwards in the tree, this content can be used to generate a very compact model of the YANG structure. Second, the top-down evaluation of the tree is mandatory, because the objects created in UML are depending on each other upwardly; e.g., no attribute can be generated without proving a class that contains the required attribute.

B. Data Reduction

As the number of objects of the model primarily depends on the number of classes and data types, trying to reduce these elements is a design requirement for the development of the set

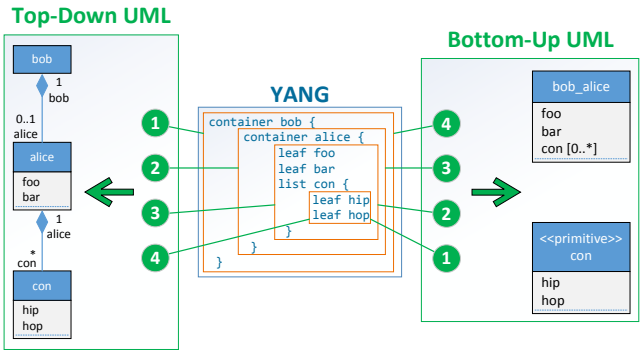

Fig. 1. Top-Down vs. Buttom-Up Processing.

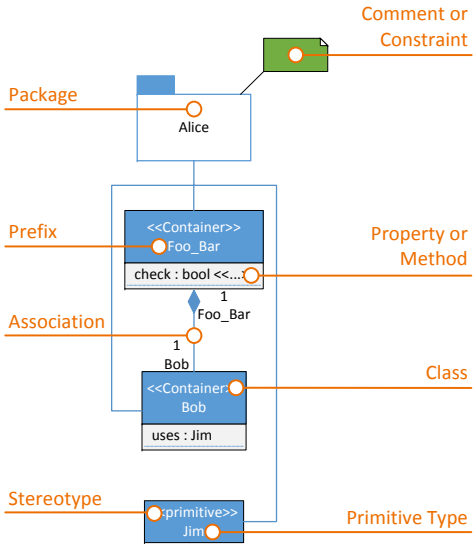

Fig. 2. Transformation from YANG to UML.

TABLE I. CONVENTION AND MAPPING OF YANG STATEMENTS FOR VISUALIZATION AND DATA REDUCTION.

YANG STATEMENT		UML ELEMENT
Name Convention		
Elements with same name	RFC 6020 / 6 cases	Uniqueness
module, submodule	⟷	remaining
extension	⟷	ext_<name>
feature	⟷	feat_<name>
identity	⟷	iden_<name>
typedef	⟷	td_<name>
grouping	⟷	gr_<name>
Removal of Illegal Characters		
- / _	⟷	Deletion
Mapping		
module	⟷	package, class PuK
submodule	⟷	class
leaf, leaf-list	⟷	attribute
list, notification	⟷	class complex datatype prefix
container	⟷	no elements: improvable class complex datatype prefix
grouping	⟷	one class: prefix otherwise: class
choice	⟷	state pattern
RPC	⟷	method, complex datatype input, output
AnyXML	⟷	attribute with comment

of mapping rules. Another aspect is the consolidation and combination of data to objects, e.g., in the form of attributes. The first step for the transformation, reduction and visualization of YANG datasets is a validation of the grammar and a translation to a further processable language. Verifying that a word and subsequently the entire document/file satisfies a regular expression is the starting point in checking that a YANG document is in accordance to the specifications [4]. RFC 6020 [5] defines an Augmented Backus–Naur Form (ABNF) metalanguage, which will be used to validate YANG sources. From the variety of tools available to translate YANG to other notations, PYANG has been selected as this tool comprises ABNF (and validates and converts YANG sources to different formats, for example YIN, UML or JSONXSL). Therefore, we make use of PYANG to generate YIN output, which is a XML format and thus somewhat human readable but also processable with available XML libraries: YANG $\xrightarrow{\text{PYANG}}$ YIN / XML. Supported by that, no further validation and conversion has to be developed by us and therefore the focus can be set onto the data reduction and visualization. A major task for the further processing is analyzing and deciding which elements can be reduced without having an influence onto the structure and semantic of the original YANG data. Furthermore, it is essential to treat similar structures equally during the reduction process to guarantee a uniform and easy understandable picture. This underlines the necessity of applying both strategies, bottom-up and top-down, during the analysis process: For preparing the required information, reducing the elements as well as prompting user decisions for undetermined situations. A further simplification of the UML representation can be realized by using equal constructs for the mapping of YANG elements (see Fig. 2). To keep the original semantic and guaranteeing a reversible transformation, stereotypes are bounded to UML elements. By that, the original differentiations are respected on the one hand and the presentation is easily interpretable for the user on the other hand. Another challenge is the translation of the different name definitions. To generate a bijective mapping, the original names of the YANG structure have to be conserved. This is realized by adding prefixes in the UML diagrams, which are generated from the names of statements and top elements. Based on the two processing strategies, differences may appear during the conversion and the user is asked to decide the preferred conversion. This decision has to be included within the UML model to maintain the reversibility of the mapping. Therefore, a list of transaction rules is defined and associations are used to keep the semantic of the YANG data models and the integrity rules. In the following, the required name conventions and transactions are described in more detail.

C. Name Conventions

Depending on the specific kind of object, every YANG element can have multiple transformations. Because the naming of packets restricts the further use of similar names in UML, rules have to be marked by names. The topmost namespace is built based on the *module* and *submodule* statements and has to be unique for the whole model, while inferior modules can have the same names. As modules define the root of the YANG tree, they will be represented by packages in the UML visualization. The second namespace is generated by the *extension* statements, which have to be unique within a module by definition. Therefore, if a collision appears, the

name of the corresponding UML object will be expanded with the prefix *ext*: <NAME> → EXT_<NAME>. Equivalent expansions are applied for the namespaces *feature*, *identity*, *typedef* and *grouping* (*feat_*, *iden_*, *td_* and *gr_*) in cases of collisions. All remaining statements share the last namespace, which is limited by the hierarchy of YANG: If a collision happens, due to equal names, the upper UML prefixes are added in front of the actual name while forbidden characters are deleted.

D. Mapping Rules

The bijective mapping between basic elements of YANG and UML is realized by the rules described as follows:

Module: Modules define the root of the YANG structure; therefore they are represented by UML packages. The respective packages can integrate subordinated elements and restrict their visibility (as in YANG, too). Because modules on the uppermost layer in YANG can contain nodes, which are represented by attributes, complex datatypes or methods in UML, an additional class named *PuK* is created to assimilate these components.

Submodule: Submodules are described by particular files in YANG, but because being dependend on superior modules, they will be integrated within the corresponding modules. As in the case of modules, an additional class will be generated for the submodule statement, if it contains YANG elements that have to be transferred to attributes, complex datatypes or methods. If no YANG elements exist, the statement will not be mapped to an UML object.

Leaf, leaf-list: The leaf statement is the smallest unit of a YANG structure; therefore it will be presented by attributes. Also, the leaf-list statement is mapped by attributes. Because these lists can be instanciated repeatedly, the attribute obtains the cardinality $[0..*]$.

List: The mapping of the list statement can be realized by multiple UML structures, depending on the required level of reduction and the YANG elements used. First, the analysis is done by the bottom-up strategy. Therefore, a list (i) can be mapped to a class (complex datatypes or mixed types in the substatements), (ii) can be mapped to a complex datatype (only attributes in the substatements) or (iii) can also be transferred to a prefix in case of an object reduction (substatements contain only type definitions and/or classes). After that, the analysis is repeated with the top-down strategy. If solely complex datatypes and classes are found, the list will be reduced to a prefix; otherwise it will be handled as class. The results of the two strategies are compared; if there are differences, the user will be asked for a decision.

Container: Analogous to the list statement, the container statement can be transferred in different ways. While the semantic of the YANG elements is different, similar mapping rules can be created for list and container statements. In case of the container statement, an additional rule has to be applied: If a container neither has basic, nor extended or special elements, it will not contain important data. Therefore, the container can be reduced or mapped to a class - this has to be decided by the user: If the container may later be used for a extension of the YANG model, it cannot be removed.

Grouping: Grouping statements are reusable, respectively referenced definitions, represented by an association in UML. Therefore, the grouping must be an object, which can be the final point of the association. If there is only one element within the grouping, it will be represented as a reference to reduce the number of objects. In all remaining cases, the grouping will be represented as a class itself.

Choice: These elements describe the selection and instantiation of exactly one of the subordinated cases. Because of that, no UML element is able to preserve the semantic completely. Therefore, state patterns, which are implementing a condition-based presentation, have to be used for the mapping, irrespective of the complexity of the original construct. By that, the choice statement is realized as a superclass. The corresponding cases are represented as classes, which are generalizations to this superclass.

Notification: The notification statement is treated as mappings of list and container statements. Therefore, prefixes, complex datatypes or classes are used, depending on the super- and subordinate structure.

RPC: In contrast to other YANG statements, RPCs describe operations with input and output parameters. For this reason, a mapping has to be realized by generating a method in UML, which is embedded into the superior class of the module. The input and output substatements are represented by complex datatypes. These datatypes can be used as parameters within the respective methods.

AnyXML: The AnyXML statements contains XML source code which is processed during the instantiation. Therefore, the code has to be transferred to the corresponding UML object, but a visualization as a concrete UML object is not necessary. Consequently, the simple representation is using an attribute, including an appended comment.

Table I gives an overview of name convention and ruleset for the mapping between YANG statements and UML representation.

E. Descriptive Elements

Descriptive elements specify superior elements. Therefore, no own UML objects are created. Instead, these YANG elements are assigned to the corresponding UML elements in different ways. Within YANG2UML, IF-FEATURES, MUST and WHEN are transferred to constraints, TYPE and DEFAULT are transferred to attributes and CONFIG, KEY, MANDATORY, ORDERED-BY, PRESENCE, REQUIRE-INSTANCE, STATUS and UNIQUE are transferred to stereotype. The remaining elements are mapped to comments.

F. Extended Elements

With the help of these elements, other elements can be extended by adding objects and parameters, without the need to redefine existing modules. In addition, new elements can be added easily to define new functions. In the following, the mapping rules for these extended elements are briefly presented:

Augment: By using this statement, extensions for external modules or the module containing the augment, as well as their submodules can be made. Although many solutions are possible, within YANG2UML, augment statements are always mapped to separate classes.

Extension: By means of the extension statement, YANG can by extended with new constructs. The value of the statement is the new keyword, which can be used by an import in other

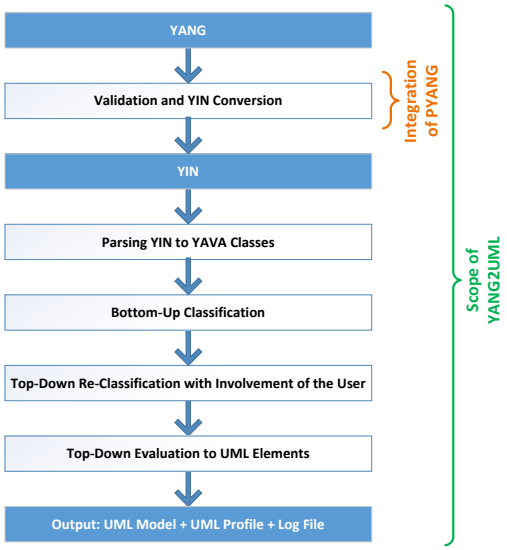

Fig. 3. Steps and Artifacts of the Conversion.

modules. Here, there is only one translation rule and that is using an UML class.

Include, import: Both statements are used to extend the main module. Includes are used exclusively in submodules and are therefore directly integrated into the UML package of each main module. For imports, new packages are built and associated to the corresponding main module.

Typedef: Through typedef statements, new data types are defined for modules or submodules. In case of an enumeration, this is represented in UML as an enumeration, too. In all other cases, the primitive data type is selected in UML. In order to keep the data of the typedef, attributes representing the typedef are created as well.

IV. Implementation

In order to support different environments, Java (with the Eclipse development environment) has been selected as the programming language for the Proof of Concept. Within the first step, the tool PYANG is executed with a command line call (see also Fig. 3). Thereafter, the YIN files will be parsed in again in order to access the elements in a convenient way. Since YIN files have a XML format, the Java-integrated XML library JAVAX.XML.PARSERS.DOCUMENTBUILDERFACTORY, an API for converting XML documents into Document Object Model (DOM) trees, will be used. By importing the libraries JAVA.WRC.COM.*, each element of the tree can be accessed very comfortable. Within each class, the parser looks for substatements of RFC 6020 [5] and adapts them to the corresponding UML specification (see Table I). Within the next step, the typedefs are processed as they are a prerequisite for the processing of elements that access these types. Only then all other statements can be parsed and thus the complete structure is built up. Another important aspect is the changing of names (i) that occur more than once and/or (ii) which can not be represented in UML objects. As YANG and UML have different namespaces, to this end, objects in UML are provided with a prefix if necessary. As already explained, a combination of both, bottom-up and top-down, is used in succession. Directly after bottom-up and during the conversion to UML elements, the top-down analysis takes place. Here, it should be noted that the top-down analysis uses the information gained from the bottom-up analysis to classify the elements and involves the user in case of non-uniqueness. In addition, various associations exist within the object model, which can only be created after the complete modeling of the objects has been performed. As the objects are generated gradually in one layer and are therefore not staggered in different layers, it is tested during the analysis which association is required. The corresponding template is our self-developed Java class SASSOCIATION, since the startup object of the association can be a class or a primitive data type. As already mentioned, at the end, the template is forwarded to the root, so that the objects are connected with the specific association. For the UML output, the UML2 API of the Eclipse development environment is used.

V. Conclusion

Although a mapping of hierarchical data models (YANG) to object-oriented models (UML) is, in principle, easily possible, practice has nevertheless shown that a meaningful and semantically correct mapping poses specific challenges. Especially the fact that a language with a lot of different constructs and different semantics (YANG) has to be mapped to another language with very few constructs (UML) represents a major problem. Nevertheless, our object models are more compact than existing solutions, and contain more information as, e.g., imports and includes are included as well. In addition, the program also offers scope for extensions and customizations.

Acknowledgment

This work was partly funded by FLAMINGO, a Network of Excellence project (ICT-318488) supported by the European Commission under its Seventh Framework Programme.

References

[1] J. Schönwälder, "Introduction to netconf and yang," *Autonomous Infrastructure, Management and Security (AIMS)*, June 2008, http://www.aims-conference.org/issnsm-2008/06-netconf-yang.pdf, last seen on 26.04.2014.

[2] Google Project Hosting, "pyang - An extensible YANG validator and converter in python," https://code.google.com/p/pyang/, last seen on 26.04.2014.

[3] M. Bjorklund, *A YANG Data Model for Interface Management*, Internet-Draft, Network Working Group, Internet Engineering Task Force, Nov. 2012, Std., 2012. [Online]. Available: http://tools.ietf.org/html/draft-ietf-netmod-interfaces-cfg-08

[4] A. Balmin, Y. Papakonstantinou, and V. Vianu, "Incremental Validation of XML Documents," *ACM Transactions on Database Systems (TODS)*, vol. 29, no. 4, pp. 710–751, 2004.

[5] M. Bjorklund, *YANG – A Data Modeling Language for the Network Configuration Protocol (NETCONF)*, RFC 6020 (Proposed Standard), Internet Engineering Task Force, Oct. 2010, Std., 2010. [Online]. Available: http://www.ietLorg/rfc/rfc6020.txt