

Building Semantically Rich Network Management Interfaces

Boris Danev
Network Management Centre
Ericsson R&D
Athlone, Ireland
Boris.Danev@ericsson.com

David Cleary
Network Management Centre
Ericsson R&D
Athlone, Ireland
David.Cleary@ericsson.com

Abstract—As mobile operators’ requirement to reduce their cost base becomes a key business driver, so does the need to reduce the time and effort associated with the management of network equipment. Within the telecommunication industry this is referred to as OPEX (OPERating EXPenditure). From a radio network perspective the majority of OPEX related expenditure comes from network configuration, optimization and planning, thus the key is to explore new approaches in these areas. In this paper we propose a novel approach based on ontologies to capture network information as well as the domain and expert knowledge needed to represent configuration tasks. After semantically enriching the representation of the 3G Radio Access Network we show how formal ontology modeling techniques can be used to build smart configuration-aware network management interfaces, and thus reduce OPEX related to configuration management. The core of our analysis is based on findings during the development of an industrial proof-of-concept in the WCDMA domain.

Keywords: *Configuration management, Wireless access network, Network information model, Management interfaces, Ontology*

I. INTRODUCTION

The operation and management of wireless networks is now driven by major financial considerations. OPEX as a reoccurring repeatable cost is a focal point in trying to achieve these cost savings. Our belief is that extensions to network management that automate labor-intensive tasks are likely to yield these reductions. The field of configuration management (CM) is the area where much time is spent on optimizing and changing the network to increase revenue. Our focus is thus on reducing costs via automation in CM. This adds the ability to spend more time and effort building highly flexible and profitable value-added services.

CM is used for setting up, controlling and modifying communication traffic in network devices such as switches, routers, base stations, etc. It includes node configuration, software management, information model and logical connection management for initial system installation to establish the network, system operation to adapt the system to short term requirements, system update to overcome software

bugs or equipment faults and system upgrade to enhance or extend the network. CM usually requires following precise procedures to guarantee consistency and integrity of the configuration information. These procedures are critical for operators to provide the quality of service required by customers.

While configuration management tasks have achieved a considerable level of automation in the IP world in network devices such as bridges, hubs or routers, the same is not true in the 3G wireless radio access networks (UTRAN). These networks are difficult to configure or reconfigure because of their highly dynamic nature, the size of the information model to be manipulated, the complex interactions related to transport, signaling, redundancy and mobility, and last but not least the complexity of the protocols.

WCDMA networks are by nature flexible entities that change often to adapt to new situations, for example to cope with changing network traffic, load distribution, etc. These changes have a strong dependence on the external environment. This dependence makes automation difficult. In general such activities require precise expert human knowledge and careful advanced planning.

Complex interactions between managed entities are needed to achieve a given configuration goal. A typical configuration task in UTRAN involves manipulating layers in many protocol stacks and ensuring consistency and integrity between layers is a challenge.

Since the early formalization of network management [1] as a discipline, many derivations and simplifications of the ideas have been proposed [2][3][4][5]. For the most part these approaches have differentiated themselves by suggesting the use of “improved” syntactical protocols and processing models. This is an adequate (though not ideal) approach for two key areas, fault management and performance management. This approach is inadequate for configuration management or provisioning. Configuration management relies on a full understanding and knowledge of the network topology and state. Today, we rely on human and expert knowledge to fill this gap and configure our networks. This inability to create

value added configuration applications stems from the lack of agreement on or the definition of, formal semantics of configuration information.

If we look at the problems of current configuration management techniques, we can see analogies with knowledge sharing, reuse and reasoning about information. In the wider research community much work is ongoing in this fledgling field. There are fields in computer science that have successfully developed knowledge-based systems. The complexity of such systems prevents them from being used in general, and the lack of standardization, modeling, and reasoning tools undermines their future development. With the Semantic web initiative [6] ontology development regained life and attracted attention from the fast growing web community.

We believe that ontologies have the potential to solve challenging problems in configuration management, creating richer data models of networks and allowing more automation for the detection of bad configurations. This facilitates the creation of better business contracts between network elements and management stations to achieve a particular task through operator workflows [3].

To summarize, we examine a new approach based on ontologies to model WCDMA radio access network configuration management concepts in order to build semantically rich configuration-aware network management interface and thus reduce (OPEX) for wireless telecom operators.

II. FORMAL ONTOLOGIES

Before examining our proposed modeling technique and architecture, it is important to look at knowledge representation. In general, this stems from the fact that at the heart of any management task is the manipulation of data as network knowledge. The Semantic Web has focused the knowledge representation research community around a single goal that envisions a new publishing paradigm for creation and deployment of intelligent knowledge based services. The central building block for this infrastructure is the formal representation of the semantics of data in sources to enable merging information and thus answer users' queries. This paradigm centers on the notion of an ontology as a way to represent data in meaningful machine-processable way.

The term *ontology* has different meanings depending on the application domain. Its meaning changes slightly if we talk to a philosopher, linguistics experts, computer scientists, etc. As far as it concerns computer science an ontology can be concisely defined as an "explicit specification of a conceptualization" [7]. As such ontology means a hierarchy of organized concepts in some abstract domain, relations between them and axioms to formalize the definitions and relations. An ontology as an explicit specification must also rely on well-understood formal semantics, which can be processed by machines.

There are a number of reasons to develop ontologies. From a computer science perspective the most relevant ones are to share common understanding of the structure of information among software agents, to enable reuse and machine analysis of the knowledge.

The ontology research community has proposed a number of ontology description languages, including the W3C standard OWL (Ontology Web Language). However the ontology definition language constructs are generally translated to one of the two main representation paradigms Description logics (DL) [8] and Frame logic (F-Logic) [9] that have emerged as formal foundations for building software able to reason about ontologies.

Both DL and F-logic can be used to represent knowledge of a domain in a structured and formally well-understood way. They also provide assertion mechanisms to constrain the knowledge, as well as algorithms to help machine processing of the knowledge base. Machine processing can be classified in two different types: TBox (related to terminology) and ABox (query answering). In general, TBox reasoning tasks are subsumption of C by D concept, equivalence of C and D concepts and disjointness of C and D concepts. ABox is about answering queries over the knowledge base, such as consistency validation, checks if something is an instance or a concept, retrieval of individuals and attributes, etc. Because of their inherently different way of handling knowledge (unary/binary predicates for DL and frames for F-logic), both paradigms present limitations with respect to TBox and ABox [10], which must be considered during the choice of ontology languages and reasoning tools.

Given the two major choices of formal foundations for ontology engineering mentioned in the background, we proceeded in analyzing the problem domain to identify the most suitable knowledge representation and reasoning paradigm between DLs and FLs.

We found that the concepts in the WCDMA information model have no complex *is-a* inheritance hierarchy. All entities derive from the top most class Managed Object. The entire hierarchy is a complex type of containment and association relationships between concepts with associated cardinality constraints. Attributes of concepts can be of different types with possible range restrictions. We found also lots of invariants in the model, as well as lots of assumptions for configuration using the model. In the domain of configuration management, we identified the need for establishing business contracts between configuration tasks and network element models participating in the tasks.

DL based languages, such as OWL [11], support subsumption for classes, and are very helpful during modeling in detecting new *is-a* relationships between classes, thus avoiding errors in modeling new concepts. While being good at modeling time, DLs do not seem to be applicable for reasoning/querying in large sets of instances and thus cannot be used as a run-time system for ontology-based applications based on the query-answer paradigm [10].

FL based languages, on the other hand, also provide support for subsumption [8] by effectively reducing subsumption reasoning to query answering. However DLs remain in general more efficient [20]. The strength of F-logic languages comes from the fact that there are well-optimized implementations for query answering such as Flora-2 [12] and Ontobroker [13], which makes them a possible run-time environment for ontology-based applications.

FL based languages have rules with well-studied semantics integrated into the language itself. For DL based languages such as OWL, the need for rules also exists, and there is currently a SWRL W3C recommendation, which integrates with OWL. However at the time of our investigation, no proper SWRL implementations were available.

Given the object-oriented style of FLs (similar to the current WCDMA information model), their inherent support for rules to model domain invariants and establish business contracts between concepts, as well as the well-optimized implementations for query answering over a large number of instances (in the case of WCDMA configuration data), we propose using F-logic based languages and inference tools for knowledge representation and reasoning over network related knowledge.

In particular we adopted a commercial F-logic based language of the German company Ontoprise¹, as well as their inference engine Ontobroker [13].

III. MANAGING WIRELESS NETWORKS

Due to the dynamic nature of wireless networks, their management involves constant operator monitoring and frequent customization of network resources. Network resource data is structured in complex information models. These models are generally based on a set of object-oriented models that raise the abstraction from physical resources to a higher level of abstraction that can be used by software management applications.

In the context of the WCDMA radio network, the hardware resource data model is available to management applications as a Managed Object Model (MOM). The MOM is part of the management adaptation layer, a higher-level abstraction of the resource layer, usually stored in relational databases. A central point in the MOM is the Managed Object (MO). The MO is an abstraction of some managed entity, such as a hardware resource, a mobile cell, or a communication channel. A manager controls an entity by creating, deleting, and modifying MOs that represent that entity. The main advantage of the MOM is that it follows the OO paradigm and thus is well structured and comprehensive. The management paradigm for all MOs is generic, irrespective of the entity that the MO represents. The Managed Object naming mechanisms consist of relative distinguished name (RDN), local distinguished name (LDN) and fully distinguished name (FDN) [14].

Each network element stores instances of MOs in a Managed Information Base (MIB). The relations and cardinalities of the MO instances are based on the MOM conceptual specification. MIB manipulation is based on the TMN's manager agent paradigm [15], where CM applications (managers) control the NE MIB (agent) by creating, modifying and deleting instance objects to facilitate the following [1] taxonomy of management.

A. Perform Configuration Operations on NEs

Currently operators perform configuration management in the Radio Access Network by following a well-defined process. First network configuration/reconfiguration is planned

offline (planning phase) based on some external event such as adding new equipment, distributing traffic load, etc. After the operator performs the planned configuration/reconfiguration by following the appropriate configuration task specification. This specification consists of steps to be executed in a well-defined order, also called configuration workflows.

Performing the configuration/reconfiguration task on network elements can be achieved by using the command-line interface (CLI), graphical user interfaces (GUI) or via machine-to-machine interfaces. In many cases management commands are grouped into scripts to speed up operational tasks. The 3G wireless network standardization body (3GPP) has specified a technology independent set of interfaces in UML. These interfaces are then mapped to solution sets in either CORBA or CMIP to provide implementation specific machine-to-machine interfaces. CM applications implement the logic of the configuration task (configuration workflows, etc) in some programming language. They provide a CLI or GUI interfaces for the customer support engineer to provide the necessary configuration input (collected during the planning phase). Validation is then performed by a separate application to ensure that the input respects particular constraints related to the configuration task. After validation, the application uses the available management interfaces on the NEs to push the programmed configuration into the network.

B. Issues with Current Configuration Techniques

Network configuration can be achieved using two different types of interfaces: man-to-machine and machine-to-machine. Man-to-machine interfaces are basically divided into command-line (CLI) and graphical (GUI). Table I outlines the main characteristics of the current configuration interfaces with respect to manipulated network element model, user input validation, application logic complexity to handle communication with network elements, maintenance of the interface and its reusability:

TABLE I. INTERFACE CHARACTERISTICS

	Configuration techniques		
	Man-Machine (CLI)	Man-Machine (GUI)	Machine-2-Machine
Network element model	MOM	UI model	XML model
User input validation	Basic	Enhanced	Further enhanced
Complexity to handle NE communication	High/Scripting	Higher/embedded in GUI	High/improved through RPC syntax
Maintenance	Difficult	Very Difficult	Difficult
Reusability	High	Low	High

From the table above, we can see that machine-to-machine interfaces have in general improved the configuration interface thanks to specific remote procedure call (RPC) syntax and processes for communication with the network elements. However because of missing semantics in the basic network element model (MOM), these interfaces do not expose rich enough network data semantics to allow more automation and intelligent processing of the network model within the interface implementation, as well as within the business logic needed to

¹ Ontoprise GmbH – <http://www.ontoprise.de>

build applications on top of the interface. Furthermore, during exploration of the WCDMA MOM, it can be seen that knowledge about the model is expressed in natural language such as:

a) *Specifying invariants on attributes in the MOM.* E.g. The Signaling ATM Adaptation Layer (SAAL) for use with ATM UNI specifies that congestion related attributes must be in a certain range.

b) *Specifying constraints between attributes in the same MO.* E.g. SAAL for use with ATM UNI specifies different congestion level attributes that follow a rule like: $0 \leq \text{congestionLevel1} \leq \text{congestionLevel2} \leq 100$.

c) *Specifying dependencies between attributes located in different Managed Objects.* E.g. The maxStat attribute defined in the UNI-SAAL profile is dependent on the AAL5 SDU size defined in the AAL5 layer MOs.

IV. ENABLING SEMANTICALLY RICH NM INTERFACE

We consider ontologies as a possible solution to represent the network management domain model and integrate expert configuration knowledge (formal configuration workflows) in order to provide a semantically rich configuration interface.

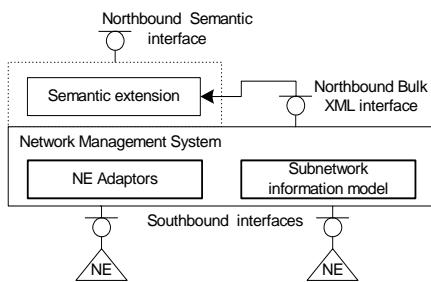


Figure 1. Semantic Northbound Interface

Architecturally the semantic extension in Figure 1 would ideally be inside the NMS to directly provide a new semantically rich configuration-aware northbound interface. For simplicity we developed our implementation on top of the existent NMS Northbound Bulk XML interface while achieving the same effect.

Our approach to building the semantic extension is as follows (see Figure 2): We take a snapshot of the subnetwork from the Bulk XML interface², then we use the provided XML configuration data to build instances of the formal ontologies representing the network. We store them in a database (Formal model repository). We then manually add ontologies and instances capturing configuration knowledge in the repository. At this point we have enriched network models stored in the Formal model repository. Now management applications can query and modify this enriched model. This is achieved through interaction with an inference engine. After any modification, the inference engine performs validation and consistency checking. At the end, the new configuration is converted back to XML for network deployment over the Northbound Bulk XML interface.

² The Bulk CM IRP [16] standard provides an XML representation of configuration data for use by external network management applications

The function of the controller (C) is to convert between the various XML and ontology instances. It also contains the inference engine to reason over the Formal model repository.

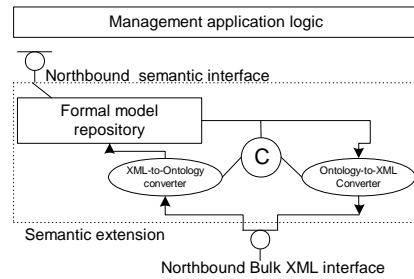


Figure 2. Semantic extension architecture

Central to our new approach are the following key features:

- Our new formal representation of network models is based on current standard ontologies to facilitate model reuse, sharing and exchange as a key to solve interoperability problems.
- Our ontologies also capture configuration task knowledge, and thus incorporate expert configuration knowledge into the configuration model.
- The semantically enriched models allow different categories of user interaction such as bulk unmanned background operation, web portals, thick clients, etc.
- The models contain domain and expert assumptions to validate the configuration at run-time, preventing errors, mis-configurations and inconsistencies.

A. Modeling Approach

We divided our formal ontology modeling into three logical parts. The first model (Enriched Network Model) is an enriched representation of the current Managed Object Model with formal semantics and axioms for domain model assumptions. The second model (Configuration Model) captures network configuration information in UTRAN. This includes configuration concepts about Transport and Radio configuration tasks, as well as configuration specific assumptions. The third model (Protocol Configuration Model) describes protocol configuration concepts and their constraints within protocol building workflows.

1) Enriched Network Model

In the first ontology, we modeled the network and corresponding equipment adding semantic information, such as cardinality and explicit containment and association relationships.

Figure 3 shows the modeling of the concept Network Element. NetworkElement inherits from default_root_concept, which is the top-level concept. Each concept must inherit from the top-level default concept. NetworkElement has an identifier (id) and an association relationship (hasManagedElement) to the top-level Managed Object called ManagedElement. ManagedElement has its own attributes and relationships to other concepts such as IpSystem and TransportNetwork, together with their associated cardinality constraints.

```

NetworkElement :: DEFAULT_ROOT_CONCEPT.
NetworkElement[id => xsd#STRING;
    hasManagedElement => ManagedElement].
ManagedElement :: ManagedObject.
ManagedElement[logicalName =>xsd#STRING;
    hasIpSystem => IpSystem;
    mincard@(hasIpSystem) -> 1;
    maxcard@(hasIpSystem) -> 1;
    hasTransportNetwork => #TransportNetwork;
    mincard@(hasTransportNetwork) -> 1;
    maxcard@(hasTransportNetwork) -> 1].

```

Figure 3. UML MO mapping to F-logic

Furthermore, we enrich the model by adding domain assumptions using F-logic axioms such as the one in Figure 4:

```

FORALL N1, N2 UniqueIdAxiom(S) <- EXISTS Id1, Id2 N1:
NetworkElement AND N2:NetworkElement AND N1[id -> Id1] AND
N2[id -> Id2] AND NOT equal(Id1, Id2) AND S is N1 + "has same
identifier as " + N2.

```

Figure 4. Formal domain assumption

This axiom enforces all network elements to have different identifiers.

2) Configuration Model

In this model we modeled knowledge about configuration tasks in the wireless access network. This basically includes tasks common to all network elements in the UTRAN such as Transport related tasks. We distinguished between high-level configuration tasks and low-level configuration tasks. High-level configuration tasks in RAN are for example adding new network element. High-level tasks are accomplished by the execution of a number of low-level tasks which create/update particular managed objects. Each low-level task is responsible for storing configuration data and assuring that this data is consistent with the configuration context in which the task participates. We identify three different sources a task may use to get information for its configuration:

- User supplied information. This information normally comes from detailed planning of the configuration task, and thus could not be automated. At this level, a human user supplies specific information, and the task can only validate or reject the user input.
- Information that could be inferred from the configuration context. This is derived from well-established business contracts between network elements taking part in the configuration. Each task in this case can suggest configuration values for its attributes based on its contracts with the network elements participating in the configuration. Thus, relieving the user from the burden of looking up information needed to complete the task.
- Information that comes from other previously configured or future tasks in the configuration process. The task implicitly fills gaps in its configuration, relieving the user from doing it manually.

A generic task is shown in Figure 5, which comprises of a name, direct successor tasks, direct predecessor tasks, status and description.

```

Task :: DEFAULT_ROOT_CONCEPT.
Task[name => xsd#STRING;
    directPre ==>> Task;
    directPost ==>> Task;
    status => xsd#STRING;
    description ==>> xsd#STRING].

```

Figure 5. Generic configuration task

Each specific configuration task inherits from this generic task. Each task has a number of attributes that must be supplied; these closely map to the attributes of managed objects the task will create or update. Each task has an internal link to empty managed object(s). This is an explicit link between the MO and the task it is related to. It also has links to other task objects indicating which information or related managed objects are needed to complete the current task.

Tasks in the Transport configuration capture concepts related to the ATM Transport Network, the standard 3G-transport protocol and also the IP protocol for management functions. Examples of concepts are Virtual Paths, Virtual Circuits, ATM ports and descriptors, etc.

```

TransportTask:: Task[atmPort => AtmPort;
    vp => xsd#INTEGER;
    vci => xsd#INTEGER;
    ..
    vplTpOb:FillInObject => VplTp;
    vclTpOb: FillInObject => VclTp;
    atmTrafficDescriptorLink => AtmTrafficDescriptor].

```

Figure 6. Transport task example

Figure 6 expresses the configuration of a virtual path (VP) and a corresponding virtual channel (VCL). The task has attributes *atmPort*, *vp* and *vci* that needs to be populated during configuration. *vplTpOb* and *vclTpOb* are special sub-properties of the property *FillInObject*. *FillInObject* property indicates which MOs this task should create or update. In order to complete the configuration, this task must use data from the *AtmTrafficDescriptor* concept. This is why there is an explicit relationship to that concept.

In order to model some generic task behavior, new knowledge derivation rules are used. These rules create new concepts based on already existing knowledge. The following rules create a new relationship “#pre” between two tasks and make the relationship transitive across tasks:

```

FORALL T1, T2 T1[pre ==>> T2] <- T1[directPre ==>> T2].
FORALL T1, T2, T3 T1[pre ==>> T3] <- T1[pre ==>> T2] AND T2[pre ==>> T3].

```

Figure 7. Task behavior modeling

The first rule in Figure 7 models a new relationship *pre* (task precedence) between two tasks based on the already existent knowledge about the relationship *directPre* (direct precedence) between the same tasks. The second rule models the well-known transitive property, namely if Task 1 has a relationship *pre* to Task 2 and Task 2 has a relationship *pre* to Task 3, this implies that Task 1 has also a relationship *pre* to Task 3.

3) Protocol Configuration Model

In this model, we captured low-level protocol configuration tasks and constraints between the Managed Objects part of the

protocol stack. Protocol configuration modeling consists of two parts. The first is identifying and modeling the low-level tasks for building the protocol configuration and the second is establishing business contract rules between tasks to accomplish consistent protocol configuration.

It is important to mention that many protocols such as NBAP-C, NBAP-D, Node-Synch, signaling protocols for the Iub [17] communication interface, share the same configuration tasks to build the protocol stack. So knowledge about the tasks is nicely reused. However some tasks have different business contracts for each protocol, which makes constraint axioms specific for a given protocol and impossible to reuse.

In order to model business contracts between tasks we use F-logic rules with context dependent relationships. These rules connect concepts in a way that satisfies some specific configuration requirement. They are not considered simply as checking rules because their role is to transfer information from one concept to another. We also call them inference rules.

```
FORALL BS, T, Id T[#baseId -> Id] <- EXISTS Controller
move(BS, Controller, T) AND T:ProtocolTask AND BS:BaseStation[#id
-> Id].
```

Figure 8. Protocol business contract rule

The rule described in Figure 8 infers the value of the #baseId attribute of the ProtocolTask *T* from the BaseStation #id when there exists a particular relationship (*move*) between a BaseStation, Controller and *T*. A direct consequence of this is that from a querying point of view, the user should not bother to know from where to retrieve the value of the protocol task attribute #baseId in a given configuration context. He just needs to query the owner of the attribute, in this case the protocol task, and the rest is left to the inference engine to find its possible values.

Even though business contract rules nicely transfer information between concepts, they do not enforce the fact that this information is the only one possible. So they act as suggestion rules. In order to enforce that the protocol task *T* must accept only an identifier coming from a base station, the following axiom must accompany the business contract rule in Figure 8:

```
FORALL BS, Controller, T, ID ProtocolTaskBaseIdAxiom(S) <-
T:ProtocolTask and BS:BaseStation AND move(BS, Controller, T) AND
NOT equal(T.baseId, BS.id) and S is T + " protocol task identifier must
match the identifier from base station " + BS.
```

Figure 9. Constraint axiom for a business contract rule

V. ANALYSIS OF THE APPROACH

We considered the configuration task of Reparent Radio Base Station. The task involves two Radio Network Subsystems (RNS), one called source RNS and the other target RNS. Each RNS consists of an RNC (Radio Network Controller) and a number of Node Bs controlled by this RNC. The *reparenting* process consists of moving control of a Node B from the source RNS to the target RNS. This means creating control structures for the source Node B in the target RNS and deleting existing control structures in the source RNS without physically changing the location of the Node B.

Moving control from one RNS to another includes building the standard protocol Iub, Mub and Aal2 links on the target RNC side and involves changes in the Transport, Radio and IP Networks. This reconfiguration must be well planned and executed carefully following a large number of constraints represented in task workflows.

Because of the complexity of the task, we concentrated our modeling efforts on the configuration of the Transport layer of Node B Application Part (NBAP) protocol part of the Iub communication interface.

The Iub interface is the communication interface between a Radio Network Controller (RNC) and a RBS (Node B). NBAP is a radio network layer protocol, which maintains control plane signaling across the Iub interface. The 3GPP defines the required functionalities of NBAP in UTRAN [18].

Configuration of NBAP Common requires creating NBAP related MOs in its protocol stack. This includes creating MOs at the ATM layer, UNI-SAAL layer and NBAP Common layer with respect to well-defined configuration workflows to ensure correctness of the information in the layers and consistency between the different layers composing the protocol stack.

An example of configuration workflow for NBAP Common is shown in Figure 10:

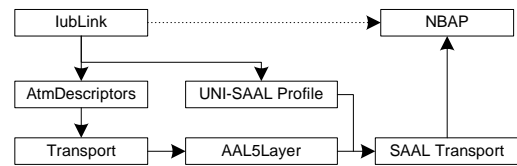


Figure 10. Example high-level NBAP workflow

It shows what kind of configuration tasks need to be accomplished in order to have a working NBAP protocol configuration. It also suggests in which order these tasks have to be completed and how NBAP configuration relates to the more global task of configuration of the Iub interface (IubLink).

A. F-logic Ontological Approach vs UML + OCL

In this section we discuss the merits of our modeling approach compared to the UML+OCL modeling approach from a configuration management perspective.

UML has formal semantics and can be used to enforce formal representation of knowledge, but because of the informal semantics enforced in commercial UML modeling tools, and the use of UML for only design purposes and visual understanding of configuration management data, the UML design of the Managed Object Model does not allow smart machine processing.

Because of their inherent object-oriented style, both UML and F-logic ontology languages overlap in many features. High-level mapping of UML concepts of package, class, class hierarchy, restrictions, data types and instances could be easily defined between UML and F-logic. Currently there are proposals for extending UML with additional richer semantics

to allow ontological development for richer modeling in the context of Model Driven Architectures (MDA) [19].

UML by itself does not allow expressing extra constraints in the domain, such as invariants. Ontology languages such as F-logic have inherent support for this, because they are used to describe knowledge and reason about knowledge. Object Constraint Language (OCL) can be used in conjunction with UML to annotate the model to specify invariants on attributes and associations and specify constraints on operations. Thus, enriching the current MOM model as shown in our approach, as well as part of the modeling of the Configuration and Protocol model and workflows can be achieved using UML and OCL as formal representation language.

The main merits of our ontological approach using F-logic in configuration management comes from using the richer semantics of F-logic, and especially its inherent and well-understood inference rules [20]. Inference rules are If/Then rules used to derive new facts from given facts. They are executed by an inference engine, thus special support for them is required at run-time. Inference rules together with the run-time support in F-logic allow smart machine processing based on the query-answer paradigm. Inference rules are successfully used in definition of high-level business processes³, as well as in some expert systems. Because inference simplifies information model manipulation, there are attempts to define UML + OCL extensions to enable support for some types of inference rules [21]. At present there no known tools implementing inference rules for UML + OCL based models, however the need for incorporating inference rules in MDA models has already been identified by the Ontology Object Management Group and work on formal definition of the rule semantics is under progress [22].

In the modeling of the Configuration model and Protocol workflows models we used new knowledge derivation and pattern matching propagation inference rules to model consistent knowledge. The former were used to derive the meaning of a general tasks' pre and post conditions based on the meaning of direct pre and post conditions. Also the modeling of some generic task behavior such as *move* or *reparent* was derived from existence of other relationships available between network elements and the support of predicate logic inside the F-logic ontology language. The latter were used to propagate implicit values between tasks as described in Section 4, thus enforcing workflow consistency in the model itself.

B. Software Development Considerations

Network domain ontologies derived from the WCDMA Managed Object Model were rather easy to express because of the available UML design of the model. As discussed in [19], a high-level mapping can be defined between UML and ontologies: classes are mapped to concepts, packages to ontologies, etc. Because ontology based models have richer semantics, UML semantics are a subset of the ontology semantics. However modeling containment relationships between classes presented two interesting approaches: explicit modeling of the containment or implicit modeling.

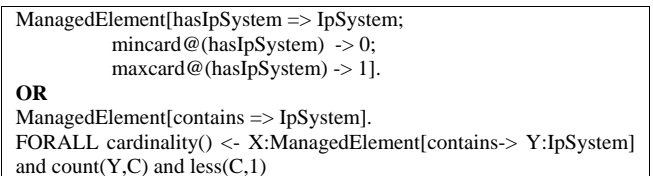


Figure 11. Containment modeling approaches

Figure 11 shows the two approaches to model the containment relationship and corresponding cardinalities between a ManagedElement and an IpSystem. In the first approach the containment relationship has its unique explicit name (*hasIpSystem*) and uses the generic approach for modeling constraints (*mincard* and *maxcard*). In the second approach the containment is expressed by using a generic name relationship (*contains*). This relationship is reused for all containment relationships in the model, thus constraining a triple (concept, contains, concept) must be modeled for each triple, which represents a big overhead during modeling time, compared to the generic approach (*mincard*, *maxcard*), which requires only one constraint checking rule. The advantage of the second approach is when writing queries to retrieve information from the model. Using this modeling approach, a query can link any two concepts by the use of the generic relationship *contains*. Because of the transitivity of *contains*, the inference engine will dynamically match the concepts no matter how far the concepts are from each other in the containment hierarchy. Using the first approach the explicit containment path from one concept to another must be provided in the query in order for the inference engine to be able to execute the query.

A hybrid approach is also possible. This can be achieved by modeling the containment hierarchy using the first approach, and either define the meaning of *contains* relationship by using the F-logic new knowledge derivation rules or using the property – sub-property mechanism in F-logic to express for example *hasIpSystem* as a sub-property of *contains*. The hybrid approach combines the advantages of both approaches, thus allowing easy cardinality modeling and simple implementations of queries over the knowledge base.

After the high-level mapping of the UML concepts available in the MOM, lots of time was spent on enriching the basic ontology model with network domain invariants. From a development perspective, much time was spent testing the domain assumptions modeled as rules by creating test cases. Currently development tools do not help with finding modeling mistakes in the rules. We have clearly seen that developing reusable and consistent models in the configuration model requires a good testing infrastructure and tool support. Also in the query-answer paradigm enforced by using ontology knowledge bases, application logic interacts with the knowledge base using the inference engine API to add, delete, modify and query the knowledge base. This couples the queries to the ontology model concepts inside the knowledge base and makes maintenance of the application logic specific to the current version of the model used. Thus, an interesting research question is how version control of the ontology model compares to version control currently used in implementing applications over the MOM.

³ Business Rules Community <http://www.brcommunity.com>

From a reusability perspective, the benefits of a well-defined formal modeling are well-known [23]. In our particular use case scenario of Reparent, we achieved a significant reusability ratio. The Configuration model included knowledge about configuration tasks in the Transport layer, which is the common layer all high-level protocols use for network communication. From this point of view, configuration knowledge representing the transport layer was completely reused in all high-level protocol knowledge modeling. The reuse at high-level protocols was more difficult to achieve because of the specificity of each protocol. However some interesting observations made possible the following reuse:

- The configuration concepts and constraints of the NBAP-C configuration workflow for building the active signaling link for NBAP-C were completely reused for building also a standby signaling link for NBAP-C. Only one modification was needed to connect the *standby* relationship of the NBAP-C configuration task to the supplying an ATM virtual circuit.
- The configuration workflow for building active and signaling links for NBAP-D configuration reused many of the constraints modeled in the NBAP-C workflow, because of the inherent similarities in both protocols. However new specific constraints had to be added.

Thanks to the high-level of reusability of the Configuration model, application logic for building redundant links for NBAP-C and NBAP-D is straight forward, requiring only a new instantiation of an already modeled consistent workflow.

In order to handle conversions between 3GPP Bulk XML representation of network data and instances of the ontology models, we developed an algorithm that automatically converts XML data to ontology instances and vice versa. We discovered that this process is directly related to how we model the semantics of the FDN (Full Distinguished Name) of the managed object concepts in the configuration knowledge. We identified two approaches to model the FDN. The first approach was to encode a string representation of the FDN for a given MO in the identifier of the concept instance. The second consisted in adding additional semantics in the Managed Object to enable configuration tasks to automatically assign the appropriate FDN of a newly created instance MO. While the first approach is simple enough, because of the missing meaning of the FDN (it is only syntactically encoded), some software engineering workarounds were needed to assign the right FDN to a newly created instance MO by a task. This resulted into hard-coding domain knowledge in the application logic itself. The second approach had the advantage to integrate the assignment of FDN in the Configuration model itself. These added semantics, together with inference capabilities of the model allowed us to dynamically assign FDNs and thus simplified the FDN management.

VI. CONCLUSION

In this paper we have shown a novel way of building more powerful northbound configuration management interfaces.

Our approach of augmenting network management models with semantic knowledge, combined with configuration knowledge modeling and consistent constraining of this configuration knowledge in protocol building workflows allowed a suggestion based approach to configuration with implicit value propagation within the formal models. This removed configuration steps and allowed user supervised semi-automatic configuration on top of the semantically richer network management interface.

ACKNOWLEDGMENT

We would like to thank **ontoprise®** GmbH for their support and assistance during our prototyping activities.

REFERENCES

- [1] ITU, "ITU-T Recommendation M.3400: TMN management functions." Geneva, Switzerland, 2000.
- [2] "DMTF Distributed Management Task Force." <http://www.dmtf.org>.
- [3] TMF, "Enhanced Telecom Operation Map, version 4," in *GB91*, March 2004.
- [4] "TINAC Telecommunication Information Networking Architecture Consortium." <http://www.tinac.com>.
- [5] 3GPP, "TS 32.101: Telecom Management Principles and High Level Requirements v6.0.0."
- [6] T. Berner's Lee, "The Semantic Web," *Scientific American*, May 2001.
- [7] T. Gruber, "A translation approach to portable ontologies," 1993.
- [8] F. Baader, I. Horrocks, and U. Sattler, "Description Logics," in *Handbook on Ontologies*, Springer, Ed., 2004.
- [9] M. Kifer, G. Lausen, and J. Wu, "Logical foundations of object-oriented and frame based languages," *Journal of the ACM*, pp. 42:741-843, 1995.
- [10] J. De Bruijn, D. Fensel, R. Lara, and A. Polleres, "OWL DL vs. OWL Flight: Conceptual Modelling and Reasoning for the Semantic Web," Nov 2004.
- [11] G. Antoniou and F. Van Harmelen, "Web Ontology Language: OWL," in *Handbook on Ontologies*, Springer, Ed., 2004.
- [12] G. Yang and M. Kifer, "FLORA: Implementing an Efficient DOOD System Using a Tabling Logic Engine," 2002.
- [13] Ontoprise, "Ontobroker tutorial," 2004.
- [14] 3GPP, "TS 32.300: Name convention for Managed Objects."
- [15] ITU, "ITU-T Recommendation M.3010: Principles for a telecommunications management network, ITU." Geneva, Switzerland, 1997.
- [16] 3GPP, "TS 32.611: Bulk CM Integration Reference Point (IRP)."
- [17] 3GPP, "TS 25.430: Iub Interface: general aspects and principles."
- [18] 3GPP, "TS 25.432: UTRAN Iub interface: signalling transport."
- [19] K. Baclawski, M. Kokar, and P. Kogut, "Extending the Unified Modeling Language for Ontology Development," 2004.
- [20] J. Angele and G. Lausan, "Ontologies in F-logic," in *Handbook on Ontologies*: Springer, 2004.
- [21] K. Wilson and I. Maung, "UML Patterns for Modeling Inference Rules Using OCL," 2003.
- [22] OMG PSIG, "Semantics of Business Vocabulary and Business Rules (SBVR)," 2005.
- [23] P. Visser and T. Bench-Capon, "On the reusability of ontologies in knowledge-system design," presented at 7th WorkShop on Databases and Expert Systems Applications, 1996.