ONTOLOGIES AND METAMODELING: APPLICATIONS TO POLICY BASED RADIO CONTROL

Mieczyslaw M. Kokar
(Northeastern University, Boston, Massachusetts, USA; mkokar@ece.neu.edu)

ABSTRACT

This paper discusses concepts that serve the purpose of handling complexity of the life cycle of engineering systems in general and SDRs in particular: languages, metalanguages, models, metamodels, ontologies and architectures. The main objective of this paper is to shed some light on the Commercial Technology WG effort to develop a Metalanguage for Wireless Systems.

1. INTRODUCTION

Software defined radios in general, and cognitive radios in particular, are very complex systems. The high complexity of these systems is the result of the sophistication of the requirements that these systems must satisfy. The complexity is further increased by the fact that these requirements can change during the operation of the software radios and that the software radio systems must be capable of adapting to such changes during the operation. The adaptation may include not only adjustments to some of the operational parameters, but also modification of hardware (reconfigurability), software (code), as well as design.

Various concepts have been developed to deal with the high complexity engineering systems. One of the main approaches to dealing with the complexity of such systems is to use abstraction. Abstraction means using a simpler (more abstract) representation of a system, i.e., hiding details. An appropriate abstraction, however, will include all the important features and relevant relationships among them, so that logical analysis of the system can be done without getting into too much detail.

A number of abstract representations have been used to deal with the issue of complexity of systems. Among others, such concepts as architecture, reference model, language and metalanguage, model, metamodel and ontology have been used as intermediate representations of engineering systems. This paper discusses such abstract conceptual structures and analyzes some relations among them. In particular, major differences and similarities between particular structures are stated and analyzed. The min goal of this analysis is to come up with some suggestions on the potential usability of the particular abstractions to the life-cycle of a software radio system. It is the hope of the author that these suggestions might be useful to the SDRF Commercial Technology Working Group in the process of developing a Metalanguage for Configurable Wireless Systems [1].

2. LANGUAGES AND MODELS

First, let us discuss such concepts as “language”, “metalanguage”, “model” and “metamodel”. It should be noted that there is no common agreement on the meaning of these terms among those who use them. Therefore, by necessity, we need to pick some of the interpretations of these terms while recognizing that others might have a different opinion.

2.1. Models and Languages

The term “model” is used in various meanings in the literature. Here is a quote from the Stanford Encyclopedia of Philosophy [2]: “On the one hand, a model can be a representation of a selected part of the world (the ‘target system’). Depending on the nature of the target, such models are either models of phenomena or models of data. On the other hand, a model can represent a theory in the sense that it interprets the laws and axioms of that theory. These two notions are not mutually exclusive as scientific models can be representations in both senses at the same time.”

The spirit of this formulation is represented in Figure 1. The figure shows a basic ontology relevant to these concepts. It shows three classes of things – Model, Language and Phenomenon/Data. These three classes are related through three object properties (relations). First, Language is related to Model by the property of *expresses*; a model needs to be expressed in a language. Second, a model *represents* a part of a domain, i.e., either some data or a phenomenon. And finally, these two properties combined derive another property, *describes*. In other words, a language describes a phenomenon or data.

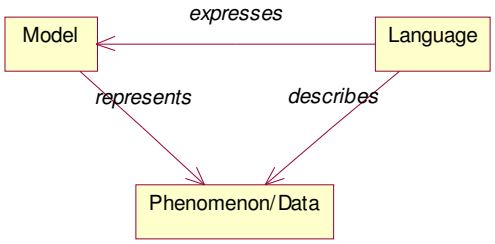

Figure 1. Model vs. Language

The above quote stresses that a model can be understood as either a representation of the world or a theory about the world (the laws and axioms). Note, however, that in either interpretation of this term, one needs to express the world in a language. In this paper we take the position that is closer to the model-as-theory view, i.e., a model is viewed as a logical theory about a phenomenon or data.

To further clarify this issue, let us delve a bit into the notion of language. Once we make a commitment to view models as logical (scientific) theories, we must also view language in formal terms, i.e., we will talk about formal languages. In other words, we will view a language as a component of a formal logical system.

A formal (first order) logical language [3] is usually defined in terms of a number of primitive symbols and a collection of rules (a grammar) that allow for combining symbols into well-formed forms (wffs), or sentences. The primitives include symbols for representing relations, functions and constants (names of individuals in a domain). Additionally, some non-logical symbols (like parentheses and such) are included.

A language is made into a formal system by adding some *axioms* and *rules of inference*. This gives a way for introducing logical arguments, or *proofs*. A proof is a sequence of wffs in which each wff is either an axiom or the result of the application of an inference rule to some of the previous wffs in the sequence. A logical system is required to be *sound*, i.e., each proof leads to a sentence that is true within the logical system. (It should be noted here that the notion of truth would require a much more involved discussion, which is well beyond the scope of this paper.) The notion of proof gives us the ability to derive which of the sentences are true and which are not. This gives a great economy or representation since not all of the true statements need to be stated explicitly. Once a collection of axioms has been selected, other true sentences can be derived through the use of inference rules.

A logical system is just a system that guarantees sound reasoning. But it does not describe any part of the world. In order to make a connection to the world, an *ontology* needs to be introduced into the logical system. This notion will be discussed in a bit more detail later in the paper. For now we just assume that an ontology includes a vocabulary of terms relevant to a given domain of interest (phenomenon or data). Having an ontology, we can now use the power of proof to derive sentences that are true about a given phenomenon or data.

2.2. Metalanguage and Metamodel

Now the question is – what is the language that defines a given language? The answer is – it's the *metalanguage*. We can thus distinguish between a *defined language* and a *defining language*. In logic, the defined language is called the *object language* and the defining language is called the metalanguage. In other words, a metalanguage is a language *about* a language.

Now the next question is – what is expressed in terms of a metalanguage? And the answer is – metamodels. In other words, a metalanguage is used to express terms that are needed to represent models. Thus now we have a two-tiered structure – one layer that includes Model and Language and another layer that includes Metamodel and Metalanguage, as shown in Figure 2.

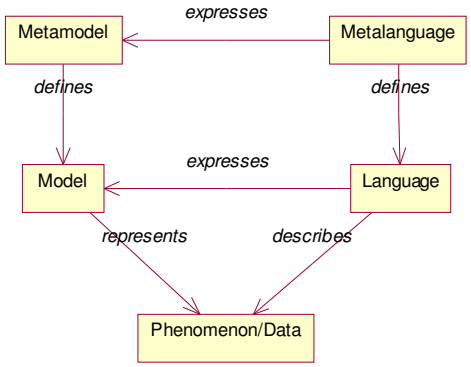

Figure 2. Metamodel and Metalanguage

While the object language is a formal language, logic uses natural languages (like the English language) to play the role of a metalanguage. Thus a metalanguage in logic is an informal language.

One could formalize the metalanguage, similarly as the object language. This is what has been practiced in computer science and software engineering. Perhaps one of the most known metamodeling architectures is the MOF Meta-data Architecture [4]. This version of MOF allows for any number of layers of models and languages.

An earlier version of MOF, 1.4 , [5] included only four layers as described below (see Figure 3). Judging by the actual use of the MOF architecture in various specifications, it seems that four layers is sufficient for most domains. Obviously, the lowest number is two, similarly as in logic.

- M0: Information Layer. This layer includes user objects, also referred to as concrete "data". To give an analogy, we can use the Java language as an example. If Java were defined in terms of MOF, Java objects would be represented at this layer. An example of a Java object could be a bit string, e.g., '1001'.
- M1: Model Layer. This layer includes metadata that describes the data at the level below. For instance, in case of Java, the Java class, BitString, would reside at this layer. The class would include some attributes, e.g., length, and methods, e.g., get().
- M2: Metamodel Layer. This layer would include meta-data about models, i.e., metamodels. An example of meta-data at this layer could be a description of a Java class Class. The notion of Class is used in all Java models. Similarly, descriptions of concepts from other languages, like the UML, would be here, for instance, the notion of the UML class Class.
- M3: Metametamodel Layer. This layer comprises a language for defining concepts in other languages, like UML, IDL and other modeling languages. In other words, this is the language for defining different types of meta-data. In order to "close" this metalanguage/metamodel stack, this top layer uses a subset of concepts from the lower layer as its language. This is similar to the approach used in natural language. It is normal in natural language to define natural language terms in the same natural language. In order to make the structure more rigid and more rigorous, the MOF uses a subset of UML as its defining language. This it is a structured language and not just a natural language.

3. ONTOLOGIES

Now we return for a moment to the issue of the role of ontologies, mentioned in the previous section. First of all, it seems useful to say a few words about the term 'ontology'. While this term became a buzzword in the past few years, it is important to recognize that various references to this term often have different intended meanings.

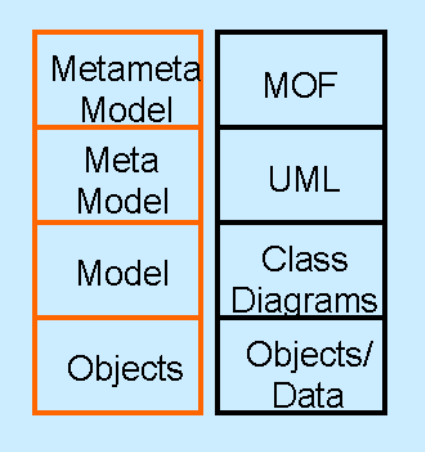

Figure 3. MOF 1.4.1 Metamodeling Architecture

The term 'ontology' originated in philosophy to represent "the science of being". In other words, ontology is a science that deals with the issue of what things do exist. A good discussion of this topic can be found in [6]:

1. "A science or study of being: specifically, a branch of metaphysics relating to the nature and relations of being; a particular system according to which problems of the nature of being are investigated; first philosophy.

2. A theory concerning the kinds of entities and specifically the kinds of abstract entities that are to be admitted to a language system."

The first definition captures the classical notion of ontology, i.e., ontology as a science. The goal of this science is to identify universals, things that exist. Examples of issues in this science include the study of such notions as class, object, part-whole relations, time, space. Since these concepts are the result of a science, they are termed "formal". This does not imply that they are described in a formal logic-based language. The next step after this is the process of "formalization", i.e., the representation of these concepts in a formal language with formal semantics. This then makes it possible to apply the power of proof, discussed earlier in this paper. Moreover, it also allows us to use various proof-supporting tools, called inference engines. The problem with this whole process is that the approach here is strictly top-down, i.e., the science of ontology deals with the very basic concepts first, with the expectation that more specific concepts would be introduced as a continuation of this process.

Unfortunately, this is not an easy task. For instance, it is rather difficult for engineers to start with such elementary concepts like time, space and being and refine them to the much more detailed level needed for practical applications. It can be observed that engineers and computer scientist tend to work in the opposite direction, i.e., bottom-up, from more specific to more general. This

approach is closer aligned with the second part of the definition of ontology.

One of the definitions of ontology in AI is [7]: "Definitions that associate the names of entities in the universe of discourse (e.g. classes, relations, functions, or other objects) with human-readable text describing what the names mean, and formal axioms that constrain the interpretation and well-formed use of these terms."

Comparing these definitions to the descriptions of models and metamodels, it appears that ontologies are in the same class as models. At the lowest level (information level), ontological representations are called "annotations", or "markups". The Model level of the MOF captures domain specific classes. This is exactly what domain ontologies capture, too. It is a normal practice to have higher, more abstract levels of ontologies. For instance, in OWL Full [8], one of the ontology representation languages, a class can contain other classes as instances. This is an exact analogy to the UML, where the metamodel includes classes whose instances are UML classes.

4. A NOTE ON ARCHITECTURE

Shaw and Garlan [9] proposed a definition of an architectural style for software-intensive systems: "An architectural style determines the vocabulary of *components* and *connectors* that can be used in instances of that style, together with a set of *constraints* on how they can be combined." This definition has then be largely incorporated in the IEEE standard 1471 [10]. As can be seen from these documents, architecture imposes constraints, most typically on the connections between components. Note that this is different than an ontology, which focuses on providing descriptions rather than constraints. In other words, systems that can use ontologies for interoperability purposes will be able to interchange information about their structure, if needed. But they will also be able to request of other systems to implement architectural constraints, if this is necessary or advantageous for the interoperability or other reasons, like optimization. The point we are trying to stress here is that ontological descriptions normally are less restrictive than architectural requirements.

5. CONCLUSION

This paper discussed various conceptual structures that are used by engineers as a means of handling complexity inherent in the life cycle of complex systems in general, software defined radios being one example of such systems. The main objective was to shed some light on what is needed in terms of the development of the Metalanguage for Configurable Wireless Systems. It is rather clear from this analysis that in order to achieve the goal set by the Commercial Technology WG of the SDRF two things are needed: 1). An ontology for the domain of SDR. The ontology needs to be a de-facto standard, i.e., it needs to be acceptable to the most of the SDR community 2). A language for expressing such ontologies. To achieve this goal, either a new language could be developed, or one of the existing ontology description languages can be selected.

Once such an ontology is developed, SDR systems that understand the ontology will be able to use the language and the particular concepts from the SDR ontology to exchange various types of information in an efficient way and to control own and piers behaviors using the policy based control [11].

REFERENCES

[1] P. A. Subrahmanyam and M. Cummings, "Perspective on a Metalanguage for Configurable Wireless Systems," Software Defined Radio Forum, SDRF-05-I-0003-V0.00, 17 January, 2005.

[2] Stanford Encyclopedia of Philosophy. Avaialable at: http://plato.stanford.edu/entries/

[3] C. C. Chang and H. J. Keisler. Model Theory. North-Holland, Amsterdam, 1992.

[4] Meta-Object Facility Core Specification. Version 2.0. OMG, formal/06-01-01. Available at: http://www.omg.org/technology/documents/formal/MOF_Core.htm, January 2006.

[5] Meta-Object Facility (MOF) Specification, Version 1.4.1. OMG, formal/05-05-05, July 2005.

[6] R. Corazzon, "Ontology: A Resource Guide for Philosophers". Available at http://www.formalontology.it/

[7] T. Gruber. "What is an Ontology?". Available at: http://www-ksl.stanford.edu/kst/what-is-an-ontology.html

[8] W3C. Web Ontology Language Reference, OWL. Available at: http://www.w3.org/2004/OWL/

[9] Shaw, M. and Garlan, D., An Introduction to Software Architecture. In Advances in Software Engineering and Knowledge Engineering, V. Ambriola and G. Tortora (eds.), River Edge, NJ: World Scientific Publishing Company, 1993.

[10] IEEE Standard 1471. "IEEE Recommended Practice for Architectural Description of Software-Intensive Systems."

[11] P. Marshall, "Spectrum Awareness". In Cognitive Radio Technology, B. Fette (Ed.), 2006.