UPDATING CRO TO CRO2

Durga Suresh (d.suresh@neu.edu)[1], Mieczyslaw Kokar (m.kokar@neu.edu)[1], Jakub Moskal (jmoskal@vistology.com)[2], and Yanji Chen (chen.yanj@husky.neu.edu)[1]

[1]Northeastern University, Boston MA USA
[2]VIStology, Framingham, MA USA

ABSTRACT

An ontology defines the basic concepts in a domain and the relationships among them. It is typically used to share information among intelligent agents and facilitates the analysis of domain knowledge. In the cognitive radio (CR) domain, two radios can achieve interoperability by exchanging the knowledge about their communication protocols and various parameters. Cognitive Radio Ontology(CRO) was developed at the Wireless Innovation Forum to model CR domain and was expressed in a formal declarative language - the Web Ontology Language (OWL).

In this paper we are presenting our work on modifying the CRO. The result of this work will be CRO2. The major modifications that we are working on are in the top level structure, properties and the relationships between the classes and objects. An ontology can be evaluated from a number of different perspectives, including (1) modularity, (2) extensibility, (3) precision in defining classes and support for automatic inference capability, (4) compactness, (5) the coverage of knowledge for the domain, and others. This paper presents examples of how the first four of these features are being addressed in the context of the development of CRO2. CRO2 will be submitted to the Wireless Innovation Forum for standardization. The main goal of presenting this paper at this conference is to seek feedback before its final submission.

1. INTRODUCTION

Cognitive radio (CR) [1] is used to describe a radio communication paradigm which takes advantage of the Software Defined Radio (SDR) architecture and allows for dynamic change of a radio's operational behavior which includes interoperability, performance optimization, opportunistic use of resources and others. An ontology provides a shared vocabulary, which can be used to model the knowledge about a specific domain. An ontology captures the types of objects (classes) that exist in a specific domain and the relations among the various types of objects. Coginitve Radio Ontology (CRO) was developed by the Modeling Language for Mobility (MLM) work group of the Wireless Innovation Forum [2] with the intent of establishing a common language that would allow cognitive radios to interoperate. The CRO includes [2]: (1) a core part that covers the basic terms of wireless communication from the physical layer and the MAC layer, (2) concepts that are needed to express the uses cases developed by the MLM Working Group [3], (3) partial expressions of the FM3TR waveform, and (4) partial expressions of the Transceiver Facility APIs.

The top level of the CRO is patterned on Descriptive Ontology for Linguistics and Cognitive Engineering (DOLCE) [4]. Figure 1 shows the top-level classes of the CRO. DOLCE is based on the fundamental distinction between Endurant, Perdurant and Quality. Endurant refers to the entity that is wholly presented at any given snapshot of time. Examples include objects (material or abstract) for example; a table, a tree, a channel or a network. In the CRO the endurants are represented by the class Object. Perdurant is the entity that can be presented only partly at any snapshot of time. A process can have temporal parts and spatial parts [5]. In the CRO perdurants are represented by the class Process. A part of an object is also an object. Similarly, a part of a process is also a process itself.

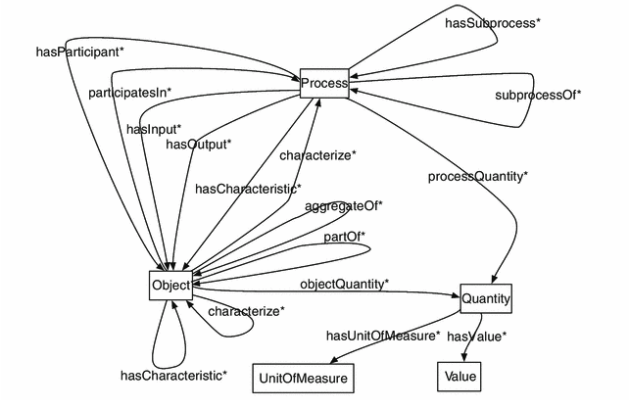

Figure 1: Top-level classes of the CRO

The following statements apply to the CRO (see Figure 1):

1. An object cannot be a part of a process, but rather participate in a process.

2. The inputs and outputs of a process are objects.

3. The capabilities of a component are collections of processes.

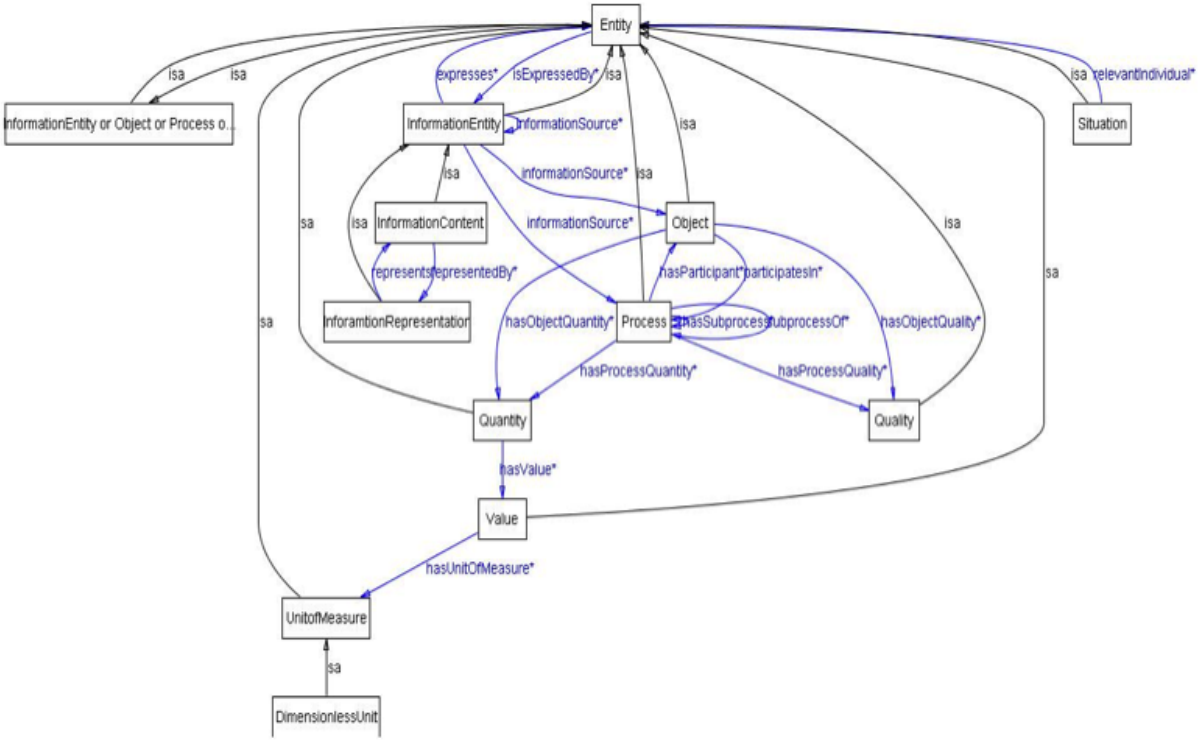

Figure 2: Nuvio Foundational Ontology

4. The characteristics of an entity, including objects and processes, can be represented as objects.

5. The qualities with units are associated with a type of Quantity. Quantity plays a similar role to that of Quality in DOLCE. Quantities can be sub-classified into different types.

6. Each quantity is associated with a UnitOfMeasure and Value.

Cognitive Radio Ontology was developed to capture the basic terms of wireless communications. The main ideas of a cognitive radio ontology originated from the Ontology-Based Radio (OBR) concept [6,7]. The OBR approach is based on the model-driven architecture implemented by means of ontologies, Java's reflection, representations of facts in Web Ontology Language (OWL) [8] and automatic inference provided by an inference engine associated with OWL. Additionally, the architecture of the OBR approach includes query support, e.g., in SPARQL [9]. Queries can be used to gather information about the structure and content of radio components, their characteristics and messages exchange. A query engine can reply to queries by analyzing the internal structure of a radio component using Java's reflection and infering facts using the systems inference capabilities. The big advantage of ontology based radios is the radio's self-awareness - a feature that is needed for cognition, as postulated by Mitola in [10].

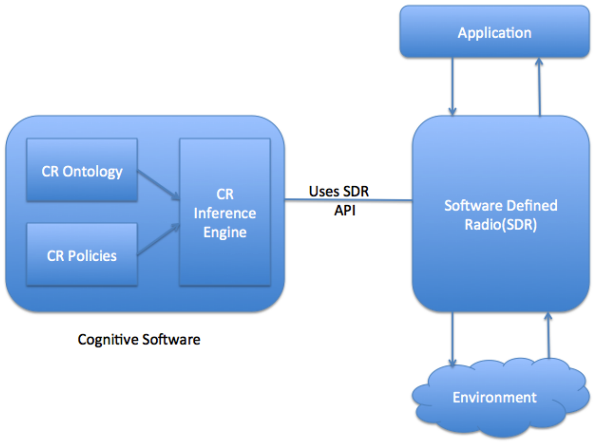

Figure 3: CRO in the realm of Cognitive Radio

Figure 3 shows the CRO in the realm of the cognitive radio. The figure shows how the CRO interfaces with the radio using the SDR API. The ontology is a part of the cognitive radio software which consists of the CRO, cognitive radio polices and the inference engine.

This paper discusses the updates being made to the CRO. Section 2. presents the original CRO and the applications of the CRO. Section 3. presents the updates being made to the CRO,

including the modifications made to the structure, properties, attributes, class and object relationships, as well as the rationale for the changes. The modified ontology will be named CRO2. Section 4. show examples of inference and consistency based on the DySpan's SCMML mappings to the CRO and CRO2. Section 5. summarizes the paper, states the conclusions made and future work on the CRO2.

2. USES OF CRO

CRO has been submitted by the Wireless Innovation Forum to the IEEE for standardization. It is being considered now by the IEEE 1900.5 Working Group. The work of this group has resulted in the publication of the requirements for an ontology-based policy language [11]. Currently, this Working Group is preparing a standard (IEEE 1900.5.1) for a policy language that partially satisfies the requirements specified in [11]. Additionally, this group is working on another related standard (IEEE 1900.5.2) [12], which will capture the specification of Spectrum Consumption Models [13]. SCM's include transmitter models, receiver models, and a combination of transmitter and receiver models to form a system model (collection of transmitter and receiver models) as well as collections that are comprised of groups of system models.

The conventions for combining the data structures of the constructs and combining the constructs to form models are described by the eXtensible Markup Language (XML) schema for spectrum consumption modeling, known as Spectrum Consumption Modeling Markup Language (SCMML). SCMML is used for communicating system models and collections. SCMML is a hierarchy of data types that build upon each other [14]. SCMML schema for communicating system models and collections written in XML is mapped to Web Ontology Language (OWL) using the CRO2.

Northeastern and VIStology are using parts of the CRO in the DARPA-sponsored project in which capabilities of RF devices are captured in an ontology and then used by the WALDO system (being developed by DARPA) for matching device capabilities with applications' needs. Moreover, VIStology is using some parts of the CRO in the development of a Spectrum Knowledge Framework, a project supported the Applied Physics Laboratory of the Johns Hopkins University (this project is being sponsored by the Office of Naval Research).

Another example of a successful application of the CRO was demonstrated by Lechowicz in [15], where a novel method for ontology-based waveform reconfigurability was described. The approach presented in [15] allows radios of different hardware or software architectures, using different software APIs and even non-uniform waveform description schemas, to interoperate. In this method cognitive radios share the same base software defined radio ontology (the CRO, extended with some concepts necessary to describe state machines), which allows the radios to understand the concepts in a uniform way, thus enabling transfer of more complex concepts between the nodes. In the process of reconfiguration, nodes can receive specifications of waveforms expressed in Web Ontology Language (OWL) and rules and then automatically configure their processing according to the specification.

CRO has been used in a combined approach of ontology and policy-based control for collaborative adaptation of cognitive radio parameters to improve the link performance as shown in [16]. This book presents many details behind the CRO and its development. A use case of collaborative link adaptation was selected to demonstrate the approach. To implement this use case, a collection of policies (expressed in the CRO) were developed and then executed by the inference engines of two radios collaborating and adapting their link.

3. MOVING TO CRO2

While the CRO is quite well accepted by the community and has many uses, there are still areas where improvements to the CRO are possible. While analyzing the CRO, we have identified the following issues that should be addressed: (1) improve the modularity, (2) provide more support for inference, (3) provide more specific concept descriptions, and (4) improve the extensibility of the ontology. Obviously, the scope of the ontology could be expanded to provide a more complete coverage of the domain, but the changes made in CRO2 so far do not address this; instead they improve the way the same knowledge as in the CRO is expressed.

Before going into the discussion of the above four issues, we first briefly mention one aspect of our ontology development effort that has impact on all of those aspects - the development of a top-level ontology. The authors in [17] [18] [19] describe a upper-level ontology, or a foundational ontology, as a means to provide reliable and reusable definitions to abstract concepts. Foundational ontologies have been developed to establish a set of concepts and definitions which could be shared by lower-level domain ontologies [17]. Foundational ontologies have been used to give a real-world interpretation to the primitives of modeling languages such as UML [20]. Foundational ontologies include highly general information modeling concepts that can be reused in the design of application ontologies for all sorts of domains [21].

As described earlier in this paper, the top level of the CRO was patterned upon the DOLCE ontology [4]. The CRO2 is based on Nuvio (Northeastern and VIStology) ontology, a new foundational ontology developed by our team. The Nuvio ontology was inspired by the original CRO, QUDT (Quantity, Units, Dimensions and Types) ontology [22], DOLCE Ultra-Light ontology (DUL) [4] and the Situation Theory ontology (STO-L) [23]. In the CRO, the concepts from DOLCE were loosely mapped to classes and properties. The mapping itself was described in documentation and papers. In the the CRO2, on the other hand, the Nuvio ontology is included through the use of the OWL concept of *import*. Consequently, the semantics of this inclusion is formally defined by the semantics of OWL.

3.1. Modularity

In 1972 Pranas [24] introduced the concept of modularization (in relation to software) as a mechanism for improving flexibility and comprehensibility of a system, the benefits of which were three fold: (1) shortening of development time, (2) flexibility to make changes to one module without changing the others and (3) being able to understand one module at a time [24]. A system thus consists of a number of components (modules) with well defined interfaces between the modules. This is in opposition to *monolithic system*, in which the whole system is just one module. The goodness of decomposition of a system into modules is assessed in terms of two metrics - *cohesion* (the strength of the internal dependencies within a module) and *coupling* (a measure of the degree of dependency between modules) [25]. A good modular design is one with high cohesion of each of the modules and low coupling among them.

The idea of modular design of software can then be mapped to the modularization of ontologies. Since currently there are no specific metrics for modularity of ontologies, in this paper we are following these ideas by analogy to these ideas for software. Our approach is based on three principles: (1) design ontologies in modular fashion, i.e., consisting of a number of modules, rather than being monolithic; (2) ensure that ontological modules have strong cross-concept relationships inside each of the modules; (3) align ontological modules with conceptual partitions of domains. This kind of design allows us to include modules whenever a specific part of the domain needs to be modeled, and leave them out otherwise.

Following the above principles, the analysis of the CRO has revealed the existence of at least four parts that could be modularized: the DOLCE-related high-level concepts, the core concepts of the physical and link layer, the FM3TR representation and the Transceiver API. Historically, the FM3TR part was included in the CRO because it was an example of a more complete model of waveform that was open source and not subject to any restrictions. The transceiver API was another more specific representation which resulted from the involvement of the Transceiver API Working Group of the Forum. These two parts are at a lower abstraction level than the other parts and thus can be used as examples of the use of CRO2 in practice.

In CRO2, all classes and properties specific to the Transceiver API and FM3TR have been moved to separate, dedicated ontologies which import CRO2 and indirectly import the Nuvio ontology as its foundational ontology. Following the modularity principles, if any changes need to be done to any one of these ontologies (Transceiver API or FM3TR), this should not affect other ontologies (CRO2 or Nuvio); these changes or modifications can be made independently of one another. Figure 4 shows a modular representation of the new ontological structure and the *import* dependencies among the parts (on the right) that were in the CRO (on the left).

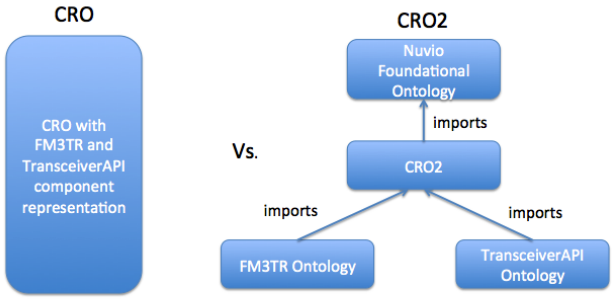

Figure 4: Modularity of CRO2 Vs. CRO

3.2. Extensibility

In the context of ontology development, extensibility refers to the features of an ontology that make it possible to extend this ontology in the future. Unlike reusability, which means using the same code without modification (copy/paste), extensibility means adding new code that just connects with the existing code. It must be doable with "minimum impact" [26]. In software engineering, a system is considered extensible, if (1) any changes can be made to any of the existing system functionalities and/or (2) addition of new functionalities can be done with, all with "minimum impact". This software development principle is used during the design phase while addressing functional and non-functional requirements. To address the aspect of extensibility in software engineering, design patterns are used [27]. A pattern describes a problem that occurs over and over again and then describes the core of the solution to that problem. Using patterns while designing systems makes it easier to understand how to extend them later. Design patterns can be grouped into three categories based on how they are used: (1) creational, (2) structural and (3) behavioral [27]. Creational patterns deal with the mechanism of instantiation of software, structural patterns are concerned with how classes and objects are composed to form larger structures, and behavioral patterns deal with algorithms and the assignment of responsibility between objects.

While extensibility can involve many patterns, the most relevant pattern from this point of view is the *decorator pattern* [27]. The main idea of this pattern are captured by the UML diagram shown in Figure 5. The Component class in this figure represents represents the core functionality, which can then be extended (decorated) using the Decorator its subclasses. In the figure only one ConcreteDecortor is shown, however, one can stack many such subclasses, and use them as needed (even making particular selections at run time). A solution without using this pattern, on the other hand, would result in the repetition of the same functionality multiple times. While this might work fine, the problem is that the maintenance of such a system would quite complicated and error-prone, since "the same functional-

ity" would have to modified in all of the subclasses.

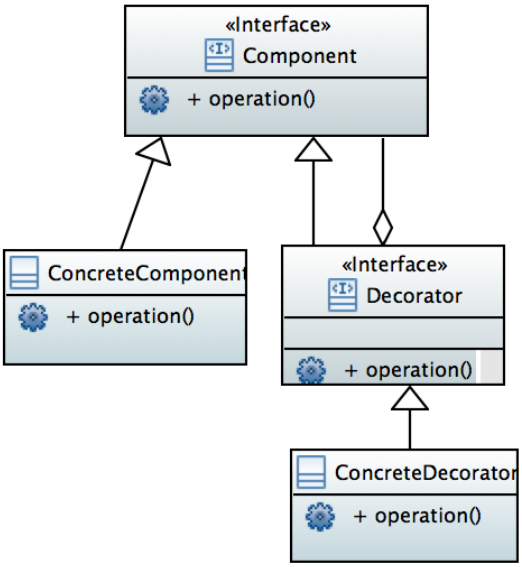

Figure 5: Design pattern: Decorator

In our development of CRO2, software engineering design patterns have been considered as the source of ideas for the structuring of both the internals of the ontologies and their relations to other ontologies. Since CRO2 is not being developed from scratch, but instead is based on the CRO, one of the design options is the restructuring of the CRO into a number of interrelated ontologies, as was shown in Figure 4. Additionally, since ontologies are not executable (procedural) code, the interpretation of extensibility is also different than the traditional meaning of this term. In particular, the ontology design patterns (cf. [28]), need to be taken into consideration.

One of the main differences between design patterns for ontologies vs. software is the semantics of patterns. While for procedural software components the behavior of the software needs to be considered, for ontologies it is captured by the preservation of the meaning of the particular concepts across the ontological structures. In OWL the meaning of the concepts is determined by the relations they are in with other concepts, both subClassOf relations and the *properties* defined in the ontology. Adding new classes that are not related to the original classes via the subClassOf relation with the concepts in the imported ontology may modify the meaning of the original classes. Similarly, adding new properties that are not subProperyOf properties in the imported ontology will add extra semantic knowledge that may have impact on the meaning of the imported ontology. Consequently, modifications to the meaning of the imported ontology by the ontology that imports it, in the perfect world, should not happen. To be fully compatible with this principle would require that all the classes in the importing ontology are subclasses of some of the classes in the imported ontology, and all the properties are the sub-properties. This requirement is rather difficult to enforce, however, many of the decisions about the CRO2 classes and properties have been striving to preserve these principles.

To analyze CRO2 from this point of view, we can review the way CRO2 extends the Nuvio ontology. In fact, all of the classes added by the CRO2 after importing Nuvio are subclasses of the Nuvio classes. Also, no new axioms are added to the top level classes in CRO2, i.e., the meaning of the Nuvio classes has not been modified. The object and datatype properties imported from Nuvio have preserved their semantics, too. However, many new properties, both object and datatype, have been added as top-level properties. This was the result of transferring the details of CRO into CRO2. Whether those properties should remain at the top is still a decision that is pending.

One of the decisions that had to be made with respect to both the extensibility of the ontology and to the fidelity of the representation of the domain knowledge was related to qualities, quantities and the units of measures of quantities. In the CRO, qualities were modeled as quantities, just without units of measurement. There was no separate class to represent qualities, but instead, they were represented as datatype properties, i.e., properties whose ranges are datatypes, e.g., int, boolean or float. This is somewhat misleading since the use of datatypes is not intended to treat the values of these qualities as such. For instance, even though the range of the quality property macNodeID is int, the intent here is to use int as a means of enumerating the IDs, but not to imply any specific ordering of the IDs, or suggest that the operations of addition and multiplication are applicable to such qualities. Finally, the fact that quality datatypes don't have units blurs the distinction between qualities and dimensionless quantities.

To account for dimensionless quantities, the class DimensionlessUnit was added as the subClassOf the UnitOfMeasure class. This is useful for extending the ontology to include the quantities and the units needed to model the radio domain. For instance, the inclusion of the quantity of Gain was achieved by asserting that its unit of measure is dB, which is a member of the class DimensionlessUnit.

In CRO2, there is a special class Quality; this approach is thus closer to the one used in DOLCE. Qualities don't have any properties that would associate them with units of measure. Instead, qualities can be defined as collections, e.g., enumeration classes. Extending CRO2 by adding additional qualities is a relatively simple operation; a subClassOf Quality needs to be added with its contents defined. Any object (or process) then can be associated with the new quality via the hasObjectQuality or hasProcessQuality, respectively.

3.3. Precision in Concept Definitions and Support for Inference

The main shift from CRO to CRO2, in terms of axioms, had to do with reusing properties by not defining their domain/range

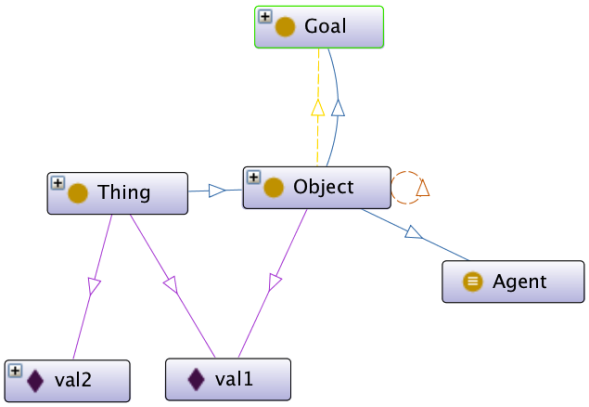

Figure 6: Agent in CRO

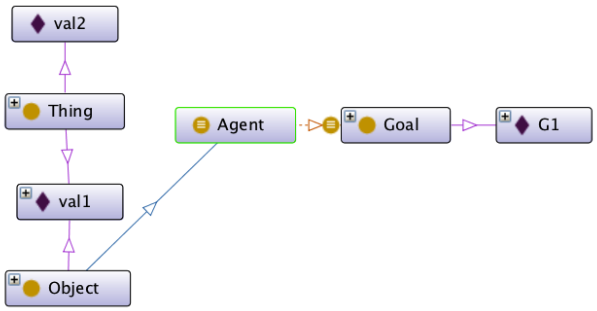

Figure 7: Agent in CRO2

relationships and adding class restrictions. For example, in the CRO, the class Agent is a subClassOf Object, where Agent is defined as a class that is equivalent to Thing that has at least one instance of Goal on the property hasGoal. Also, domain and range classes for the hasGoal property are Object and Goal, respectively. This is explained in Figure 6.

In CRO2, on the other hand, although Agent is still subClass of a higher class (in this case Object), it is also defined as an equivalent class to any Object that has at least one goal, i.e., is associated with a goal by the hasGoal relation. The domains and ranges of hasGoal are not defined, and thus if a modeler decides to associate a goal with something else than Object, e.g., Process, this is still allowable, however the inference engine would not infer that that process is an agent. This is explained in Figure 7. Thus by this kind of refactoring, it is possible to both support the inference, but also to avoid unnecessary derivations. This part of refactoring, at the time of the writing of this paper, is still in process. The plan is more operations of this type on many other classes and properties.

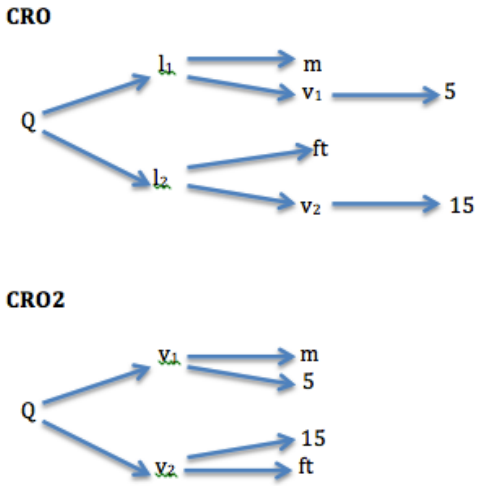

Figure 8: Example of Quantity **Length** described in CRO Vs. CRO2

3.4. Compactness

Intuitively, compactness of an ontology is the measure of its size with respect to the size of the domain knowledge that it covers. Since there is no good measure of the latter, we consider this measure only informally. We could, on the other hand, provide a quantitative measure of the size of an ontology by counting the number of classes, properties and axioms that it includes. Here we provide some discussion on how the issue of compactness has been addressed so far. So first of all, as mentioned earlier, two chunks of representation (FM3TR and Transceiver API) have been isolated as modules. This makes CRO2 much leaner in terms of classes, sub classes, object properties and data properties. The separation of radio specific domain and moving the general concepts to Nuvio has resulted in the removal of the classes and various properties. Moreover, the CRO properties have been refactored into a new property hierarchy. For example the property aggregateOf in the CRO had 13 sub properties, but in CRO2 these have been deleted and replaced with just two properties.

Another example of refactoring is the representation of quantities. In CRO2, the approach used in the QUDT ontology [22] has been adopted. The main idea of this refactoring is shown in Figure 8. This figure shows how two values, v_1 and v_2, are represented in CRO (top) and in CRO2 (bottom). Clearly, the CRO2 representation is more compact and it has the advantage that the association between values and units of measure is very clearly represented.

CRO has 50 top-level object properties, while CRO2 has 44. In total, CRO has 156 properties, while CRO2 has 81. CRO has 41 datatype properties, while CRO2 has 18. Note that for CRO2 these numbers include both the classes/properties from CRO2 and Nuvio. The representation of terms of the cognitive radio ontology has not changed due to this change, in fact we are able

to infer from CRO2 all the facts that were inferable from CRO, and more.

4. SUMMARY OF CHANGES

The CRO consists of 228 classes, 190 properties [5] covering the basic terms of wireless communications from the PHY layer, MAC layer and network layer. CRO2 consists of 169 classes and 98 properties covering all the terms that are covered by the original CRO. The major aspects of the development of CRO2 included:

1. A new foundational ontology (Nuvio) has been developed that is inspired by the original CRO, and other ontologies like QUDT, STO-L and DOLCE.

2. The representation of terms of the Transceiver API and FM3TR have been extracted to separate ontologies.

3. Quality features of ontologies have been analyzed and applied to CRO2. Some those features were inspired by similar considerations in the software engineering community.

4. The quality features have influenced the process of refactoring of the particular versions of the ontology.

5. CONCLUSION AND FUTURE WORK

This paper shows that the CRO2 ontology being developed at Northeastern University and VIStology. This ontology will soon be submitted to the Wireless Innovation Forum as an input document. The expectation is that the Wireless Innovation Forum will submit it to the IEEE 1900.5 Working Group for incorporation into the standard that this group is currently working on.

In this paper we discussed the various design issues that have been considered in the process of developing this ontology, including such features as modularity, extensibility, compactness, precision of concept definitions, and support for automatic inference.

Our future work will involve using the CRO2 which imports the Nuvio foundational ontology to write rules to show the advantages of having a foundational ontology rather than just having top level classes as in the original CRO. CRO2 is also being used to represent the Model Based Spectrum Management approach and describe Spectrum Consumption Models (SCMs). model and will have the SCM's ontology developed.

The refactoring of this ontology is still work in progress. We are seeking input from the community for all aspects of this endeavor, including recommendations on the coverage of the concepts from the domain of wireless communications, the structure of the ontology and the definitions of the particular concepts. The ontology in OWL can be downloaded from www.vistology.com/ont/CRO2. Comments and suggestions can be emailed to mkokar@vistology.com.

REFERENCES

[1] B. A. Fette, *Cognitive Radio Technology (2nd Edition)*. Elsevier, 2009.

[2] M. M. Kokar, D. Brady, and K. Baclawski, "The Role of Ontologies in Cognitive Radios," in *Cognitive Radio Technology Chapter 13*, B. Fette, Ed. Academic Press, Elsevier, 2009, pp. 401–428.

[3] MLM Working Group, "Use cases for MLM language in modern wireless networks (SDRF-08-P-0009-V.1.0.0)," The Software Defined Radio Forum, Tech. Rep., January 2009.

[4] C. Masolo, S. Borgo, and A. Gangemi, "DOLCE : a Descriptive Ontology for Linguistic and Cognitive Engineering," Institute of Cognitive Science and Technology, Italian National Research Council, Technical report, 2003.

[5] S. Li and M. M.Kokar, "Cognitive Radio Ontology," in *Flexible Adaptation in Cognitive Radios*, ser. Analog Circuits and Signal Processing. Springer New York, 2013, pp. 67 – 78.

[6] J. Wang, D. Brady, K. Baclawski, M. Kokar, and L. Lechowicz, "The use of ontologies for the self-awareness of the communication nodes," in *Proceedings of the Software Defined Radio Technical Conference SDR'03, SW3-004*, 2003.

[7] J. Wang, M. M. Kokar, K. Baclawski, and D. Brady, "Achieving Self-Awareness of SDR Nodes Through Ontology-Based Reasoning and Reflection," in *Proceedings of the Software Defined Radio Technical Conference SDR'04*, 2004.

[8] W3C, "OWL 2 Web Ontology Language Document Overview," 2009, `http://www.w3.org/TR/owl2-overview/`.

[9] ——, "SPARQL Query Language for RDF. W3C Candidate Recommendation," 2006, available at: `http://www.w3.org/TR/2006/CR-rdf-sparql-query-2006040`

[10] J. Mitola., "Cognitive Radio: An Integrated Agent Architecture for Software Defined Radio," Ph.D. dissertation, Royal Institute of Technology (KTH), 2000.

[11] "IEEE Standard for Policy Language Requirements and System Architectures for Dynamic Spectrum Access Systems. IEEE Std 1900.5^{TM}-2011," IEEE Communications Society, 2011.

[12] IEEE, "IEEE DySPAN 1900.5.2 standard," 2015, `http://grouper.ieee.org/groups/dyspan/5/index.htm`.

[13] L. Grande, M. Sherman, H. Zhu, M. M. Kokar, and J. Stine, "IEEE DySPAN 1900.5 Efforts To support Spectrum Access Standardization," in *2013 IEEE Military Communications Conference*. IEEE, 2013, pp. 1750–1755.

[14] J. A. Stine and S. Schmitz, "Model-Based Spectrum Management," *Version 2.0 MITRE Corporation*, April 2011. [Online]. Available: https://www.mitre.org/sites/default/files/pdf/11 2071.pdf

[15] L. Lechowicz and M. M. Kokar, "Waveform reconstruction from ontological description," *Analog Integrated Circuits and Signal Processing*, vol. 78 (3), pp. 753–769, 2014.

[16] S. Li, "Collaborative Adaptation of Cognitive Radio Parameters Using Ontology and Policy Based Approach," Ph.D. dissertation, Northeastern University Boston, 2011.

[17] L. Magee, "Upper-level ontologies," *Towards A Semantic Web: Connecting Knowledge in Academic Research*, p. 235, 2011.

[18] L. Schneider, "Designing Foundational Ontologies," in *Conceptual Modeling ER 2003*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2003, vol. 2813, pp. 91–104.

[19] I. Niles and A. Pease, "Towards a Standard Upper Ontology," in *Proceedings of the International Conference on Formal Ontology in Information Systems - Volume 2001*, ser. FOIS '01. New York, NY, USA: ACM, 2001, pp. 2–9.

[20] G. Giancarlo, H. Heinrich, and W. Gerd, "Towards ontological foundations for UML conceptual models," in *On the Move to Meaningful Internet Systems 2002: CoopIS DOA and ODBASE*. Springer, 2002, pp. 1100 – 1117.

[21] G. Aldo, G. Nicola, M. Claudio, O. Alessandro, and S. Luc, "Sweetening ontologies with DOLCE," in *Knowledge engineering and knowledge management: Ontologies and the semantic Web*, 2002, pp. 166 –181.

[22] R. Hodgson and P. J.Keller, "QUDT-quantities units dimensions and data types in OWL and XML," *Online (September 2011) http://www. qudt. org*, 2011.

[23] M. M. Kokar, C. J. Matheus, and K. Baclawski, "Ontology-based situation awareness," *Information Fusion*, vol. 10, pp. 83–98, 2009.

[24] D. L. Parnas, "On the Criteria to Be Used in Decomposing Systems into Modules," *Commun. ACM*, vol. 15, no. 12, pp. 1053 – 1058, Dec. 1972.

[25] L. L. Constantine and E. Yourdon, *Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design*. Youdon Press, 1978.

[26] B. Annappa, R. Rajendran, K. Chandrasekaran, and K. C. Shet, "Analyzing Design Patterns for Extensibility," in *Computer Networks and Intelligent Computing*. Springer, 2011, pp. 269 – 278.

[27] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design patterns: elements of reusable object-oriented software*. Pearson Education, 1994.

[28] NeOn, "Ontology design patterns," 2013, `http://ontologydesignpatterns.org`.