ACHIEVING DYNAMIC INTEROPERABILITY OF COMMUNICATION: Transfer of Ontology and Rules Between Nodes

Leszek Lechowicz (Department of Electrical and Computer Engineering, Northeastern University, Boston, MA; llechowi@ece.neu.edu); Mieczyslaw M. Kokar (Department of Electrical and Computer Engineering, Northeastern University, Boston, MA; mkokar@ece.neu.edu)

ABSTRACT

In this paper we investigate the situation in which one of the Cognitive Radio (CR) nodes wants to use an advanced modulation scheme, which is not implemented in the other node. Consequently, it needs to tell the other radio what it means by that modulation scheme. Our approach assumes that there exists a basic ontology and a set of rules shared among the two CRs, as well as a basic communications protocol, which enables the nodes to communicate and share information between them. The node with the advanced modulation scheme may transfer its knowledge (i.e., ontology and rules) to its peer. The other node is using a reasoner and the transferred knowledge in order to construct the algorithm from other functional blocks achieving the interoperability at the modulation scheme level.

1. INTRODUCTION

Constructivism, a paradigm in cognitive science that followed behaviorism and cognitivism [1], considers knowledge as a structure that is *constructed* in the memory of a learner, rather than a static (absolute) entity that is common to all agents. The consequence of this theory is that knowledge cannot just be copied from one cognitive agent to another, but rather needs to be first communicated and then re-constructed using the mental capabilities of the learner. Here are some excerpts from [2] that outline the basic ideas behind the three theories of cognition and learning.

"*Behaviorism*; Based on behavioral changes. Focuses on a new behavioral pattern being repeated until it becomes automatic.

Cognitivism: Based on the thought process behind the behavior. Changes in behavior are observed, but only as an indicator to what is going on in the learner's head.

Constructivism; Based on the premise that we all construct our own perspective of the world, based on individual experiences and schema. Focuses on preparing the learner to problem solve in ambiguous situations."

The theory of cognitivism seems to be well suited to the problem of transfer of structural knowledge between cognitive radios. In this paper we analyze a scenario in which one cognitive radio (the teacher) conveys knowledge of its capabilities to another cognitive radio (the learner). The important aspect of this exercise is that the learner's architecture differs from the architecture of the teacher. Consequently, the learner has to interpret the teacher's knowledge and then re-construct it using its own components and architecture.

In order to achieve this kind of capability, the radios must be able to adapt their power levels, transmitting/receiving frequencies, modulation schemes, coding and encryption standards etc., according to the current needs and conditions, subject to constrains like power limits, band plans, supported modulation schemes and encryption techniques.

This paper reports on the progress in our investigation of the scenario described above. We concentrate on a case in which one of the nodes wants to use an advanced modulation scheme, which is not implemented in the other node. Consequently, it needs to tell the other radio what it means by a modulation scheme. Our approach assumes that there exists a basic ontology and a set of rules shared among the two CRs, as well as a basic communications protocol, which enables the nodes to communicate and share information between them. The node with the advanced modulation scheme may transfer its knowledge (i.e. ontology and rules) to its peer. The other node then uses a reasoner and the transferred knowledge in order to construct the algorithm from other functional blocks achieving the interoperability at the modulation scheme level.

2. BACKGROUND AND PREVIOUS WORK

Self-awareness is one of the most important prerequisites for the interoperability of Cognitive Radio nodes. The nodes can only be truly interoperable with other radios if they understand their own capabilities and limitations. In our previous research [3] we proposed that CR nodes could be made self-aware by using OWL ontologies and appropriate reasoners. We further proposed that such a self-aware node could use information about itself and additional data retrieved from its peers through queries, to optimize communication parameters with regard to certain predefined metrics (i.e. channel throughput, bit-error rate, power consumption etc.).

One of the possible methods of optimizing communications is switching to different modulation schemes depending on the quality and the bandwidth of the available channel. CR nodes sharing the same ontology could negotiate to switch to a particular algorithm. This kind of optimization however can only be effective if all participating nodes have similar capabilities so that the final modulation scheme chosen depends only on the parameters of the channel and the specifics of the communication to be conducted, rather than on the technical capabilities of the least advanced CR node participating in the exchange of data.

Therefore it appears that in order to maximize the benefits of switching modulation algorithms, all nodes participating in the communication should implement all modulation techniques that could ever be used in any given set of circumstances.

It is obvious that such an approach would not work very well in practice. First of all, since the nodes are supposed to share the same ontology, an extensive ontology covering all aspects of radio communication and beyond (e.g., encryption techniques) would have to be developed and accepted by all CR vendors. Additionally, any incremental change in the functionality would require that an ontology and operational code of the CR node be updated as part of the introduction of the change. Even if only one of the nodes on the network were not updated to the latest version, the whole network would not be able to use the new features. The logistics problems inherently present in the situation where one is trying to upgrade a large number of devices deployed in the field almost guarantee that a certain subset of devices would not be upgraded. This would not allow for a practical implementation of the idea negotiation and selective use of different components and algorithms.

Base ontology is a complete set of basic facts and properties such that all other facts and properties in any related ontology can be derived from. If a given set of CR nodes shares the same base ontology it is possible to transfer more advanced concepts from one node to another simply by expressing them in terms of the base ontology and adding them to the knowledge base of the target node. In a more complicated scenario, when the base ontology for node A is not the same as the base ontology of node B, the transfer of knowledge also involves *ontology mapping* which is beyond the scope of this research, as we assume that participating nodes share the same base ontology.

The successful transfer of knowledge, however, is only the first step towards the interoperability between the nodes. In order to make any practical use of that knowledge, the target node has to be able to construct an algorithm from the facts it has just learned.

Since most of the algorithms used in cognitive radios would share a lot of the primitive subroutines (e.g., algorithms for addition, multiplication, scaling, FIR filtering etc.), it seems that component-based algorithm composition might be the right approach to the problem.

The component-based approach to solving software engineering problems is an established paradigm and is widely discussed in the software engineering literature. Some of the ideas are particularly suitable for mobile/embedded systems with limited resources and processing power, such as cognitive radios.

For example, Urting et al. [4] have developed an interesting approach in which the system takes into consideration not only functional requirements for the service that is supposed to be composed from the simpler components, but also some non-functional aspects, like memory size restrictions or timing constraints. Their system can only use a particular software component if it satisfies so called *contract* (if one exists) i.e., an agreement with the system to limit the use of system resource or require certain acceptable level of service (e.g. latency). The contracts are used not only during the service composition stage, but also in the runtime to assure that imposed restrictions are not violated. It's worth noticing however that their framework doesn't address the problem of specifying which components are supposed to be connected and in what way in order to achieve the required functionality.

Preuveneers and Berbers [5] built upon the described above approach and introduced some elements of automated context-driven composition of the services. In their paper they describe a service called *Communication Service*, which, depending on the context (i.e., available network bandwidth, available processing power, available memory), can instantiate particular types of *Video Filter* and *Video Encoder* components. Their framework uses an OWL [6] ontology for the description of the component meta-model concepts, like components, ports, parameters, connectors, contracts etc. Since the component meta-model concepts are described in terms of OWL classes, the particular services, like *Communication Service,* can only be described in terms of OWL *individuals*. While an automatic OWL reasoner, like Pellet [7], can be used to prove that the particular individual (e.g. *Communication Service*) is consistent with the definition of the composite component, it cannot prove that two separate individuals are equivalent, even though they might be.

Clearly, if one could express composite components as classes in OWL rather than individuals, it would enable the use of automatic tools for reasoning giving a new level of potential capabilities.

3. PROPOSED INTEROPERABILITY SCENARIO

Since the overwhelming majority of the functionality of the software-defined radio is enclosed in its software, potentially as long as two software-defined radios from two different vendors don't differ in terms of hardware capabilities (e.g., the frequency they are able to use), they could be made fully interoperable simply by modifying the software behavior on any or both of the nodes. One way of making that happen is a brute-force approach of upgrading the radio's firmware when it is moved from one network to another. Of course that approach is an undesirable one as the radio cannot be used in two incompatible networks at the same time. Even more advanced techniques like automatic updates through the network don't solve the most fundamental problem – the fact that the traditional software defined radio can only operate in ways their designers predicted.

This limitation would not apply if radios could learn their behavior from one another. This notion is the foundation of our proposal.

There are several assumptions in our scenario:

1. The CR nodes share the same *base ontology* i.e. the same set of basic concepts that all other concepts in CR domain can be derived from.
2. The CR nodes have a way of communicating with each other. They should also have a fall back procedure, in case that for some reason the communication between them fails.
3. The CR nodes can query for and can respond to queries for specific facts in their knowledge bases.
4. The CR nodes can reason about the facts in their knowledge base and the facts learned from the other nodes through queries. The results of that reasoning can be used to modify their internal data structures.
5. A CR node can query for an arbitrary ontology concept. Its peer responds by transferring a fragment of its ontology base related to that concept. The first node can then incorporate that knowledge into its own ontology and can generate a composite software component based on that knowledge. In case an unknown concept is defined in terms of other unknown concepts, the node will have to query for those concepts as well. That recurrent process ends only when all the necessary concepts are known.

Any specific node configuration (peer-to-peer, one master – many slaves, hierarchical structure) can be used; the selection of a particular one is beyond the scope of this research.

Table 1 illustrates an example of the exchange during a hypothetical negotiation between the nodes.

Table 1. An example of an interaction sequence

NODE A	NODE B
<query>error bit rate, channel equalizer coefficients</query>	
	< response> error bit rate = …, channel equalizer coefficients = {….. } </response>
[Node A uses the data received from Node B and its own communication parameters to make a decision to send a request to switch to QAM16]	
<request> change modulation to QAM16 </request>	
	<query>QAM16 modulator </query>
<response> QAM16 modulator is a composite component which consists of ……, quadrature modulator, … etc. </response>	
	<query> quadrature modulator </query>
<response> quadrature modulator is a composite component which consists of 2 multipliers, 1 adder, 1 phase shifter, connected in the following way;… </response>	
	[Node B builds the model of QAM16 modulation using collected facts and facts in it's knowledge base. Uses the reasoner to prove that such constructed component is consistent with the transferred facts.]
	<ack>changing to QAM16</ack> or <nack>cannot change to QAM16 </nack>

In this example Node A queries Node B for various communication parameters. Based on that information and its own communication parameters it makes the decision that in order to maximize the quality of the transmission, both nodes should switch to QAM16. It sends the request to Node B, which doesn't recognize the modulation scheme so it replies to Node A with the request for the description of QAM16 modulator. After receiving the description, Node B notices that it doesn't know what quadrature modulator is so it sends another query to Node A. After the definition of the quadrature modulator has been transferred, Node B has the full description of QAM16 modulator so it can try to build a model of that component. If that is successful, Node B uses a reasoner to prove that whatever it built is indeed the QAM16 modulator. If the building step was successful Node B can respond with a positive acknowledgement for the modulation change request. Otherwise it responds with negative acknowledgement.

4. CURRENT EXPERIMENT

One of the most important problems that need to be solved before the proposed interoperability scenario could be implemented is the process of constructing the composite component from the facts learned from other nodes. Such an algorithm would have to include at certain point a phase in which a candidate for the constructed component is scrutinized by the reasoner in order to verify if it is consistent with the fact base. Our initial research efforts have been directed towards that crucial phase.

In our experiment we were trying to find a way to define an OWL class for a composite component (quadrature modulator) in terms of other OWL classes representing basic components (e.g. adder, multiplier, etc.). The limitations of what can be expressed in OWL have become obvious very quickly.

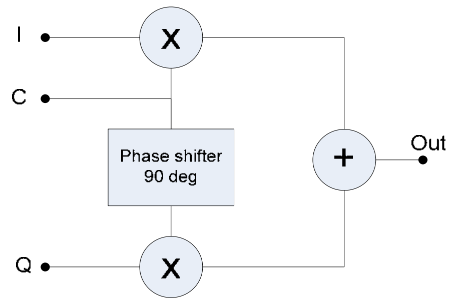

Figure 1. Quadrature modulator

For example the quadrature modulator (see Fig. 1) has two multipliers. We can express in OWL the fact that a particular class (such as *QuadratureModulator*) is in a relationship with other class (such as *Multiplier*) using a property (in our case *hasSubComponent*). We cannot however distinguish the relationship with one of the multipliers from relationship with the other one. Similarly we can express in OWL the fact that the *QuadratureModulator* is in the relationship with *InputPort*, but again there's no way to distinguish between particular relationships for input ports I, Q and C.

It became clear that we need to augment OWL ontology with rules in order to be able to express more complicated relationships. One of the options was to augment OWL with SWRL rules. We rejected however this idea because of the lack of efficient reasoning engines supporting SWRL and to the fact that SWRL is undecidable. As an alternative we decided to use BaseVISor rules in the experiment.

5. BASEVISOR REASONER

BaseVISor ia a forward-chaining inference engine developed by VIStology, Inc [8]. It is based on the Rete network optimized for the processing of RDF triples, it also incorporates axioms and consistency checks for R-entailment which supports all of the RDF/RDFS and a part of OWL-DL semantics. In exchange for the price of not supporting all of the OWL-DL elements, BaseVISor provides P-SPACE performance for ground RDF graphs [9].

BaseVISor uses a rule language based on XML syntax. The facts defined by triples consist of subject, predicate and object elements as it is shown below:

```
<triple>
  <subject resource="ll:QuadratureModulator/>
  <predicate resource="ll:hasComponent"/>
  <object resource="ll:Multiplier"/>
</triple>
```

BaseVISor's rules consist of `body` and `head` elements. The body usually contains one or more triples, which have syntax similar to the syntax of the facts, except that triples in the rule bodies can contain variables. The head portion of the rule is used to assert or retract facts into the knowledge base. An example of a simple rule is shown below:

```
<rule name="hasMultiplier rule">
  <body>
    <triple>
      <subject variable="comp" />
      <predicate resource="ll:hasSubComponent" />
      <object variable="mul" />
    </triple>
    <triple>
      <subject variable="mul" />
      <predicate resource="rdf:type" />
      <object resource="ll:Multiplier" />
    </triple>
  </body>
  <head>
    <assert>
      <triple>
        <subject variable="comp"/>
        <predicate resource="#hasMultiplier "/>
        <object variable="mul" />
      </triple>
    </assert>
  </head>
</rule>
```

The fact that the object *comp* is in a relationship *#hasMultipler* with the object *mul* will be asserted only if *mul* is a subcomponent of *comp* and *mul* is of type *Multipler.*

6. ONTOLOGY

The ontology we constructed for this experiment is shown in Figure 2, Figure 3 and Figure 4. The ontology includes only two top-level classes: Component and Port. As shown in Figure 2, Component includes two subclasses – Basic Component and Module. The intention here is to say that an instance of Basic Component is a primitive component that cannot be constructed out of simpler components. Module, on the other hand, can be constructed out of Basic Components. The building blocks for Module are Multiplier, Phase Shifter, Adder and Filter. Obviously, this is a very limited selection – the minimum that we needed to demonstrate the main idea in this paper.

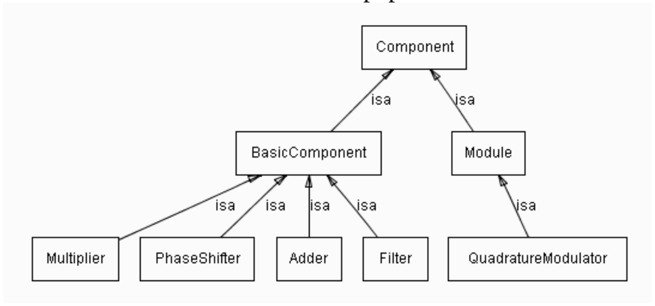

Figure 2. Subclasses of Component

In addition to Component, we also provide the Port class. As shown in Figure 3, Port includes two subclasses – Input Port and Output Port.

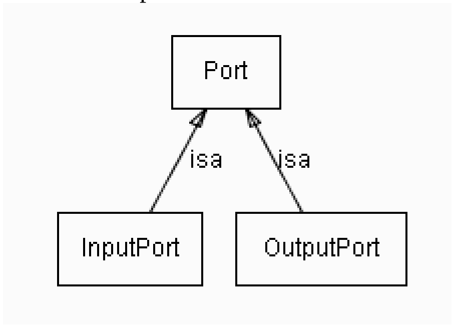

Figure 3. Subclasses of Port

Modules will be described by the components they are built from and by the Ports that are connected to them. Connections among components and ports are captured by the four properties represented in Figure 4: hasPort, isPortOf, isSubcomponentOf, hasSubcomponent and isConnected to. The domains and ranges of these properties can be inferred from Figure 4 by looking at the arrows representing the properties.

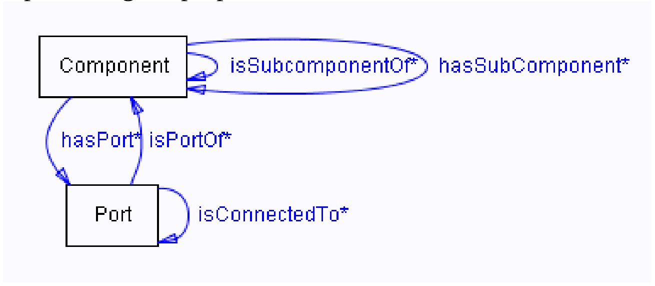

Figure 4. Properties

7. CONSTRUCTING RULES WITHIN OPEN WORLD REASONING MODEL

RDF and OWL use the open world assumption model. In that model facts that have not been explicitly asserted to be true are not presumed to be false, they simply are unknown. That kind of reasoning while beneficial for the expressiveness of OWL increased the complexity of rules that needed to be written in our experiment, because we needed to state in those rules not only what components and in what way should be connected with each other, but also the fact that there are no other components or connections.

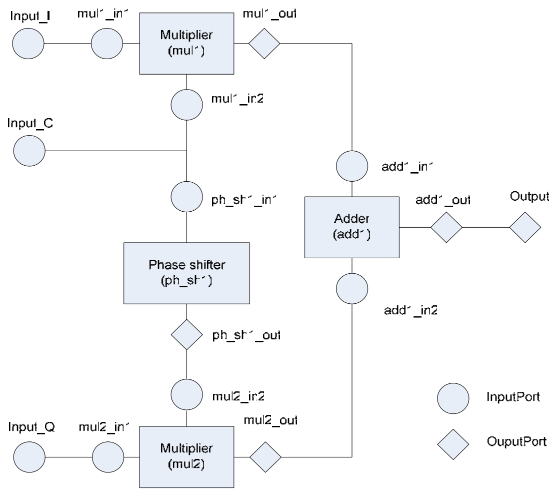

Figure 5. Graphical representation of the quadrature modulator structure expressed in BaseVISor rules

The diagram above presents a visualization of the quadrature modulator rules as they were implemented in our experiment. The relationships (thin solid lines) between ports are equivalent to *isConnectedTo* properties defined in OWL with additional restrictions stating that except those connections that have been explicitly stated in the rules no other connections can exist. Similarly relationships between components and their ports have similar restrictions as to the number of ports as well as the exclusivity of their relationship with their designated components. The last set of restrictions is necessary to avoid an erroneous situation in which the same input port is in a relationship with different component. The following BaseVISor rules is an example of the definition of the exclusive relationship between input port and the component.

```
<rule name="NonExclInput Rule">
  <body>
    <triple>
      <subject variable="inp" />
      <predicate resource="#inputPortOf" />
      <object variable="comp1" />
    </triple>
    <triple>
      <subject variable="inp" />
      <predicate resource="#inputPortOf" />
      <object variable="comp2" />
    </triple>
    <not>
```

```
      <triple>
        <subject variable="comp1" />
        <predicate resource="owl:sameAs" />
        <object variable="comp2" />
      </triple>
    </not>
  </body>
  <head>
    <assert>
      <triple>
        <subject variable="inp" />
        <predicate resource="#isNonExclusiveInputPortOf" />
        <object variable="comp1" />
      </triple>
      <triple>
        <subject variable="inp" />
        <predicate resource="#isNonExclusiveInputPortOf" />
        <object variable="comp2" />
      </triple>
    </assert>
  </head>
</rule>

<rule name="ExclInput Rule">
  <body>
    <triple>
      <subject variable="inp" />
      <predicate resource="#inputPortOf" />
      <object variable="comp" />
    </triple>
    <not>
      <triple>
        <subject variable="inp" />
        <predicate resource="#isNonExclusiveInputPortOf" />
        <object anonVar="true" variable="__c" />
      </triple>
    </not>
  </body>
  <head>
    <assert>
      <triple>
        <subject variable="inp" />
        <predicate resource="#isExclusiveInputPortOf" />
        <object variable="comp" />
      </triple>
    </assert>
  </head>
</rule>
```

Predicates *#isNonExclusiveInputPortOf* and *#isExclusiveInputPortOf* are complementary i.e., one of them cannot be true if the other one is true. Due to the open world assumption, however, they cannot be expressed as a simple logical negation of each other. In the above example that leads to the necessity of adding a clause enclosed in <not> </not> tags, which specifically states that in order for the input port *inp* to be an exclusive port of component *comp* it cannot be in non-exclusive port relationship with any other component.

8. CONCLUSIONS AND FUTURE WORK

In this paper we proposed that the constructivism paradigm in cognitive science could be applied to the problem of interoperability between Cognitive Radio nodes. In the scenario we proposed, the node with richer knowledge base can transfer portion of that knowledge to another node which can further use it to construct composite software services it previously didn't know. We reported on progress in our research efforts which initially have been concentrated on problems of expressing complex software components as OWL classes and logical rules under the open world assumption reasoning model.

Our current and future efforts will concentrate on problems related to knowledge transfer and component composition, which are essential to the success of the proposed approach. The use of reasoners in the Cognitive Radios opens many other interesting research problems. One of the topics that would be very interesting to explore is the use of the local knowledge, which is not shared with other nodes to reason about the facts the node learned from other nodes. It is especially important if the node has a specialized hardware that could and should be used whenever possible to help with the data processing. For example, if the node learns about a composite component that includes multipliers and adders (like quadrature modulator), and if the node has a specialized hardware implementing multiply-accumulate operation, it could potentially use that hardware in the implementation of that composite component.

9. REFERENCES

[1] Ference Marton and Shirley Booth. *Learning & Awareness.* Hillsdale, N.J.: Lawrence Erlbaum Associates, 1997.

[2] CSCL (Computer Supported Collaborative Learning), Pedagogical Information Science at the University of Bergen, Norway. Avaialble at: http://www.uib.no/People/sinia/CSCL/web_struktur-4.htm.

[3] J. Wang, D. Brady, K. Baclawski, M. Kokar, 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, 2003.

[4] D. Urting, S. Van Baelen, T. Holvoet, Y. Berbers. *Embedded software development: Components and contracts.* In Proceedings of the IASTED International Conference Parallel and Distributed Computing and Systems, 2001, pp. 685-690.

[5] D. Preuveneers, Y. Berbers. *Automated Context-Driven Composition of Pervasive Services to Alleviate Non-Functional Concerns*. International Journal of Computing & Information Sciences, Vol. 3, No.2, August 2005, pp. 19-28

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

[7] Pellet homepage: http://www.mindswap.org/2003/pellet/

[8] VIStology, Inc. http://www.vistology.com/

[9] C. Matheus, K. Baclawski and M. Kokar. *BaseVISor: A Triples-Based Inference Engine Outfitted to Process RuleML and R-Entailment Rule.* To appear in Proceedings of the 2nd International Conference on Rules and Rule Languages for the Semantic Web, Athens, GA, Nov. 2006