THE USE OF ONTOLOGIES FOR THE SELF-AWARENESS OF COMMUNICATION NODES

J. Wang (Department of Electrical and Computer Engineering, Northeastern University, Boston, jiawang@ece.neu.edu); K. Baclawski (College of Computer Science , Northeastern University, Boston kenb@ccs.neu.edu); D. Brady (Department of Electrical and Computer Engineering, Northeastern University, Boston, brady@coe.neu.edu); M. M. Kokar (Department of Electrical and Computer Engineering, Northeastern University, Boston, kokar@coe.neu.edu); L. Lechowicz (Department of Electrical and Computer Engineering, Northeastern University, Boston, leszek_lechowicz@ltx.com).

ABSTRACT

This paper investigates an approach to establishing communication by explicitly maintaining self-awareness and communication of knowledge about the operation of the communication nodes. The self-awareness and communication of knowledge is based on the maintenance of an explicit, declarative knowledge base or *ontology* of communication. Hence, we refer to the concept as *Ontology Based Radio* (OBR). The proposed approach is based on the model-driven architecture implemented by means of ontologies, DAML-based annotations and Java's reflection capabilities. Each software module can be queried about its structure and contents using a DAML based query. It can then reply to the queries by analyzing its own structure using Java's reflection and the system's inference capability. In this paper we will show an example of such a functionality in which two nodes exchange information and then reason about the multipath structure. The net result is that after analyzing the multipath structure nodes can improve the efficiency of communication.

1. INTRODUCTION

Wireless transmission requires a robust and efficient communication protocol. When the channel has been estimated and the estimation has been sent back to the transmitter, then the transmission can be adapted according to the channel characteristics. The basic idea behind adaptive transmission is to maintain a constant SNR level, E_b / N_0, by varying the transmission power level, symbol transmission rate, constellation size and coding rate/scheme or any combination of these parameters [1].

Software radio is an emerging technology for building flexible, multi-service, multi-standard, multi-band, reconfigurable and reprogrammable radios [2]. Software radio is very attractive for adaptive wireless transmission because of its potential capability for dynamically adapting to the radio environment (or channel characteristics) through the reconfiguration of its components. Although migrating algorithms from hardware to software can increase the level of functionality, it does not necessarily change the fact that the communication protocols are established at design/development time. For example, if we want the designed SDR to be adaptive to the rmsDelay (root mean square delay spread) of the fading channel, we have to hard-code it at design time. That means that the adaptive characteristics are known at the time of design. While this kind of assumption is sufficient for many cases, there are situations in which it is not possible to design a communication scheme in advance. In those situations, the communication scheme needs to be constructed, managed and efficiently controlled dynamically at run time.

This paper investigates an approach to establishing communication by explicitly maintaining self-awareness and communication of knowledge about the operation of the communication nodes. The self-awareness and communication of knowledge is based on the maintenance of an explicit, declarative knowledge base or *ontology* of communication. Hence, we refer to the concept as *Ontology Based Radio* (OBR).

Here is what Mitola says about the lack of awareness, including self-awareness, of current radios [3]: Today's digital radios have considerable flexibility, but they have little computational intelligence. For example, the equalizer taps of a GSM SDR reflect the channel impulse response. If the network wants to ask today's handsets "How many distinguishable multipath components are in your location?" two problems arise. First, the network has no standard language with which to pose such a question. Second, the handset has the answer in the structure of its time-domain equalizer taps internally, but it cannot access this information. It has no computationally accessible description of its own structure. Thus, it does not "know that it knows." It cannot tell an equalizer from a vocoder. To be termed "cognitive", a radio must be self-aware. It should know a minimum set of basic facts about radio and it should

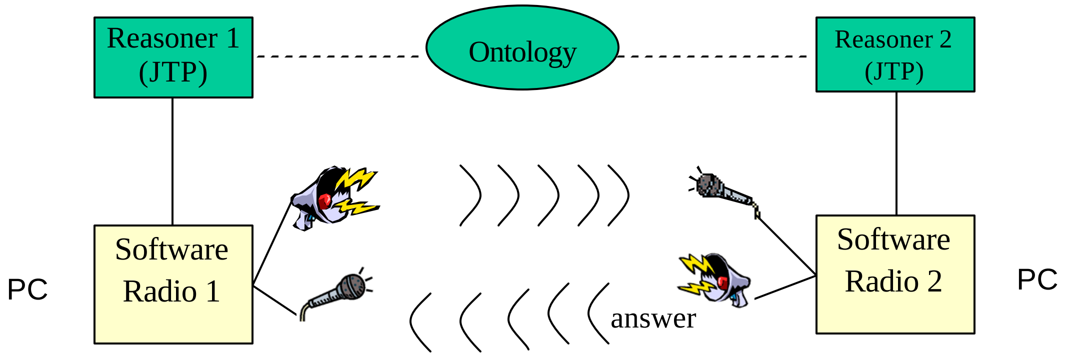

Figure 1: OBR simulation scenario

be able to communicate with other entities using that knowledge. For example, it should know that an equalizer's time domain taps reflect the channel impulse response.

The rest of the paper will explain the concept of OBR and the implementation of a simplified scenario. In section 2, we will present the concept of OBR. In section 3, a simple scenario will be used to show how to establish communication between two OBRs. In section 4, we show how the concept of OBR was implemented. In section 5, we show some results from our experiments. In section 6, we present the conclusions.

2. OBR CONCEPT

The main idea of OBR is that the communication nodes "understand" the contents of information to be transferred, their own capabilities and capabilities of the destination units. Based on this understanding, the communication units establish a protocol that is "good" with respect to the transmission needs, to the capabilities of the communicating units and to the current situation of the whole communication system. This understanding is expressed using ontologies. Ontologies are at the core of semantic information processing. Ontology captures the basic terminology (concepts) of the domain of interest and the relationships among the concepts. Ontologies can be expressed graphically using the Unified Modeling Language (UML) [4], represented using the DARPA Agent Markup Language (DAML) [5] for ease of interchange, and processed either off-line using a theorem prover or in real time using an expert system engine.

3. PROOF-OF-CONCEPT SCENARIO

Figure 1 is a simple scenario which shows how the communication protocol is established between two OBRs. We use two personal computers with a speaker and a microphone to simulate OBR. In other words, we simulate a piece of functionality of an OBR using an acoustic link. Three components are involved in establishing the communication protocol: Radio, Monitor and Reasoning Agent. The radio component is responsible for sending and receiving data. The Monitor is responsible for synchronizing the two OBRs and controlling the communication. The Reasoning Agent stores the ontology in its knowledge base and answers queries using this ontology and data obtained from the system using Java's reflection.

A typical process for establishing communication includes the following steps:

Step1: Monitor1 determines that the quality of service is too low and requests that Radio1 send a packet containing a query to Radio2. Radio1 sends the query.

Step 2: Radio2 receives a packet containing a query and forwards the query to Monitor2. Monitor2 then requests that Reasoning Agent2 answer the query. Reasoning Agent2 infers the data that is required to answer the query using the Ontology, then uses Java reflection to extract the data from the Radio2 software. Here the ontology is used for formulating the reflective method invocation.

Step3: Reasoning Agent2 sends the answers of the query to Monitor2 which then requests that Radio2 send a packet with the answer. Radio2 then sends the answer to Radio1.

Step4: Radio1 receives the packet and forwards the answer to Monitor1. Monitor1 chooses new communication parameters and requests that the radio send a packet with a command. Radio1 then sends the packet to Radio2.

Step5: Radio2 receives the packet and forwards the command to Montor2. Monitor2 requests that Reasoning Agent2 execute the command. Reasoning Agent2 infers changes that must be made to the software of Radio2 using the Ontology. Reasoning Agent2 then uses reflection to update the program variables of Radio2. Here the ontology is used for formulating the reflective method invocations.

4. IMPLEMENTATION OF OBR

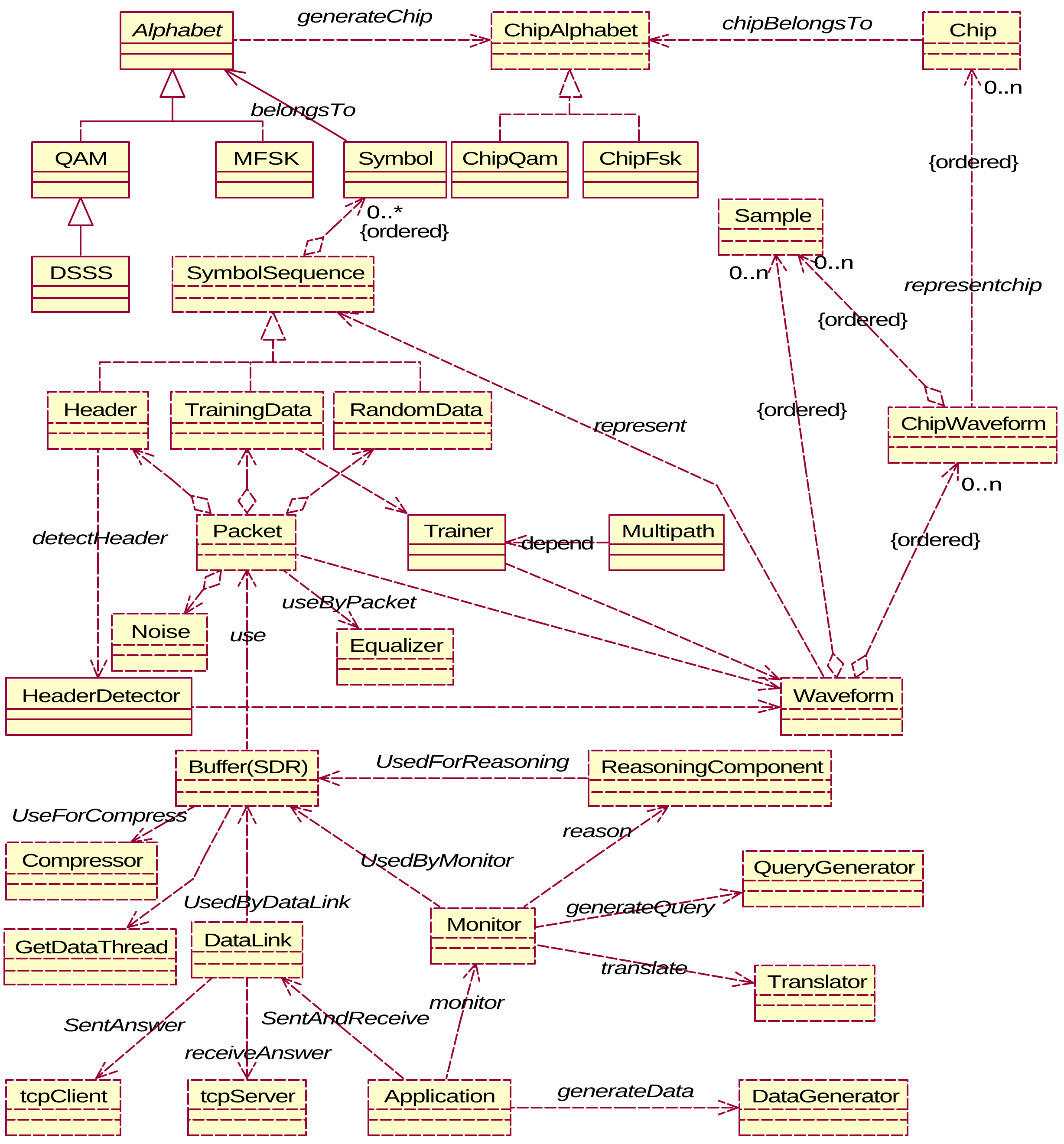

Figure 2: OBR Ontology (partial, details suppressed)

The proposed approach is based on the Model Driven Architecture implemented by means of Ontology, DAML-based annotations and Java's reflection capabilities. Each software module can be queried about its structure and contents using a DAML based query. It can then reply to the queries by analyzing its own structure using Java's reflection and the system's inference capability.

4.1. Ontology

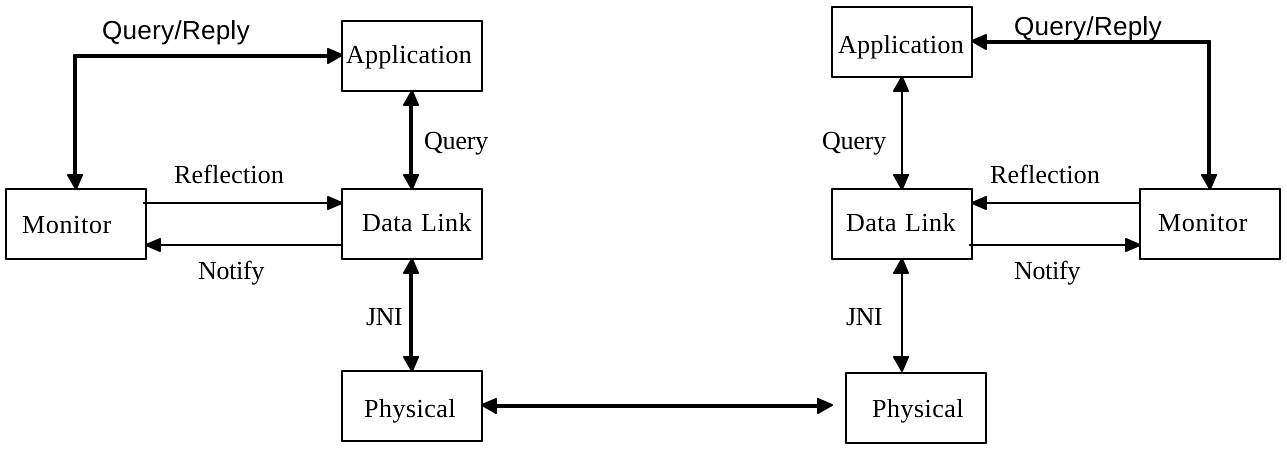

Figure 3: The architecture of OBR

Ontology captures the basic terminology (concepts) of the domain of interest and the relationships among the concepts. In Figure 2, we show the Ontology of OBR in UML. UML is a graphical language so it is easier to read by people, as compared to a language like DAML. But it is not computer-understandable. So we translated the UML expressed ontology into DAML, which can be used for information interchange and for processing by many available tools.

4.2. Architecture

In figure 3, we show the architecture of OBR. The Physical layer is simply a C program, which accesses the sound card directly. The Data Link layer does most of the work of SDR: compressing, filtering, modulation and so on. The RLS algorithm is used to calculate the multipath structure of the fading channel, and an equalizer based on the RLS algorithm is used in the receiver to process the received data [6]. The interface between the Physical and Data Link layers is the Java native interface (JNI). Queries are generated by a Monitor, which monitors the Data Link layer. The Data Link layer notifies the Monitor whenever something happens, wich then makes a decision whether the performance is OK so far, or if not, then it queries the Data Link layer using Java reflection. When it receives an answer or a request to update the Data Link layer, it again uses Java reflection. The queries generated from the Monitor or the replies from the Monitor are sent to the Application Layer, which then transmits the query/reply as normal data. Producer-consumer queues are used here to save the queries/replies generated.

4.3. Reasoning Agent

The reasoning agent of our OBR is a component of the Monitor. When the Monitor receives queries, the queries are sent to its Reasoning Agent, which keeps the OBR ontology in its knowledge base, infers the data that is required to answer the query using this Ontology, and finally uses Java reflection to extract the data from the Data Link layer software.

We use JTP for a Reasoning Agent. JTP is KSL's object-oriented modular reasoning system. It is based on a simple and general reasoning architecture. The modular character of the architecture makes it easy to extend the system by adding new reasoning modules (reasoners), or by customizing or rearranging existing ones [7]. JTP's knowledge bases are written in the Knowledge Interchange Format (KIF). JTP also provides support for querying knowledge represented in DAML. This is achieved by translating a DAML ontology into KIF when the DAML representation was loaded into the knowledge base.

Java reflection is a built-in feature of the Java programming language. It allows a Java program to examine or introspect itself during run time. In order to use Java reflection with JTP, a new reasoner, called the extractor, was written and added to JTP.

DAML Query Language (DQL) [8] is a formal language and protocol for agents to use in conducting a *query-answering dialogue* using knowledge represented in DAML. Since the knowledge base of JTP is written in KIF, a translator was used to translate between KIF and DQL before the query is sent to a Reasoning Agent and the answer is sent back.

5. SOME RESULTS

Since the processing of radio frequency signals requires a great deal of computational power, we use an acoustic link, instead of an RF link. This allows us to experiment with the main ideas relevant to the SDR, and yet use a PC platform. Currently, one acoustic link has been implemented using a

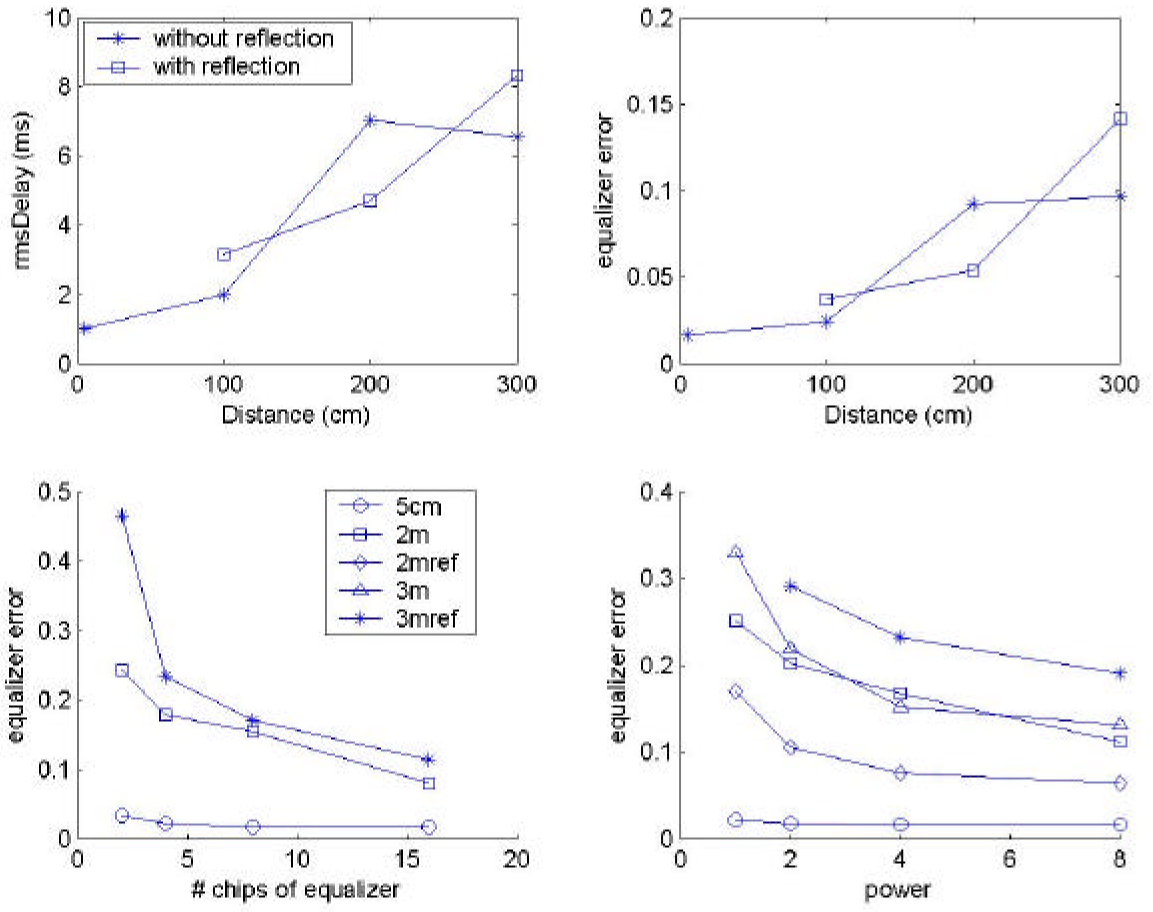

Figure 4: Experimental Results

Speaker and a microphone, and the other link uses an internet connection.

5.1. Example of Query/Answer

To demonstrate the concept, we show a query about the ExcessDelay (multipath delay spread) and rmsDelay (root mean square delay spread) of the multipath structure of the channel, and the mean square root error of the equalizer. Here the mean square root error of the equalizer represents the average distance between the equalized data (the input data of the equalizer multiplied by the equalizer chips) and the output of the equalizer (the estimated symbol). BPSK is used here, and it is assumed that the two symbols are in the unit circle.

```
<SOAP: Envelope>
  <SOAP: Body>
    <dql:query>
      <dql:queryPattern>
        <rdf:RDF>
      <rdf:Description
rdf:about="&obr;buffer">
        <obr:currentrmsdelay
rdf:resource="&var;x"/>
      </rdf:Description>
      <rdf:Description
rdf:about="&obr;buffer">
        <obr:currentexcdelay
rdf:resource="&var;y"/>
      </rdf:Description>
      <rdf:Description
rdf:about="&obr;buffer">
        <obr:currentequalizererror
rdf:resource="&var;z"/>
      </rdf:Description>
    </rdf:RDF>
      </dql:queryPattern>
      <dql:mustBindVars>
        <var:x/>
    <var:y/>
    <var:z/>
      </dql:mustBindVars>
      <dql:answerKBPattern>
        <dql:kbRef rdf:resource="&obr;"/>
      </dql:answerKBPattern>
    </dql:query>
  </SOAP:Body>
</SOAP:Envelope>

The answer of this query is:

<SOAP:Envelope>
  <SOAP:Body>
    <dql:answerBundle>
      <dql:answer>
        <dql:binding-set>
          <var:x>1.0078370372505556</var:x>
          <var:y>1.062759005498691</var:y>
          <var:z>0.025987243652343</var:z>
        </dql:binding-set>
      </dql:answer>
    </dql:answerBundle>
  </SOAP:Body>
</SOAP:Envelope>
```

ExcessDelay and rmsDelay are in milliseconds.

5.2. Summary of Results

Figure 4 represents some experimental results with our system. The top left plot shows that the rmsDelay increases as the distance between the speaker and the microphone increases. We tried four distances: 5cm, 1m, 2m and 3m for a configuration without reflections, i.e., when there were no obstacles in the line of transmission. Similarly, we used three distances in a configuration with reflection: 1m, 2m and 3m. In this case there were some objects around the speaker and the microphone that were able to reflect the sound wave. The ExcessDelay showed the same trend as the rmsDelay, but we did not plot it here. The upper right plot shows that the mean square root error of the equalizer is also increased as the distance increases. In this experiment we used an equalizer of size 48, 16 feedback chips and 32 feed forward chips.

The bottom plots show the effect of the length of the equalizer and the transmission power on performance. As the length of the equalizer of the receiver increases, or as the transmission power of the transmitter increases, the performance improves.

There is a trade off between performance and processing time. As the equalizer length increases, the processing time on the receiver increases. So to maintain the same performance as the distance between the speaker and microphone increases, or when lower power is used, the length of the equalizer has to be increased, resulting in increasing the processing time of the receiver. This can be achieved through negotiation between two nodes using the approach presented in this paper.

6. CONCLUSION

We have presented an approach for improving communication by explicitly maintaining self-awareness and communication of knowledge about the operation of communication nodes. A simulated communication scenario was presented to prove the concept we proposed. A partial implementation of this scenario has been developed. A PC with a speaker and a microphone (acoustic link) was used to simulate wireless communication. An ontology of communication software was written in DAML and processed by JTP, which was used as the reasoning agent. Some preliminary results have been shown. The main goal of this was to show that communication can be improved by exchanging information among nodes and adjusting some communication parameters accordingly. The main advantage of this approach is that an ontology provides communication nodes with a very expressive language for querying each other's capabilities and communication parameters. The work described in this paper is just a first step towards the goal of a Cognitive Radio.

7. REFERENCES

[1] A. J. Goldsmith, "Variable-Rate Variable-Power MQAM for Fading Channels." IEEE Trans. Commun., vol. COM-45, pp. 1218-1229, Oct. 1997.
[2] E. Buracchini, "The Software Radio Concept." IEEE Communication Magazine, Sep, 2000.
[3] Joseph III Mitola, "Cognitive Radio: An Integrated Agent Architecture for Software Defined Radio." PHD thesis, Royal Institute of Technology (KTH), 2000.
[4] OMG Unified Modeling Language Specification, Version 1.3. Technical report, Object Management Group, 2001.
[5] DAML. DARPA Agent Markup Language web site, 2001. www.daml.org.
[6] John G. Proakis, "Digital Communications", McGraw-Hill, New York, 1995.
[7] R. Fikes, J. Jessica, and F. Gleb, "JTP: A System Architecture and Component Library for Hybrid Reasoning.", Proceedings of the Seventh World Multiconference on Systemics, Cybernetics, and Informatics. Orlando, Florida, USA. July 27 - 30, 2003.
[8] DQL Specification, Technical report, Joint US/EU ad hoc Agent Markup Language Committee, 2003.