# APPLYING FIPA STANDARDS ONTOLOGICAL SUPPORT TO INTENTIONAL-MAS-ORIENTED UBIQUITOUS SYSTEM

Milene Serrano and Carlos José Pereira de Lucena
*Department of Informatics, Pontifical Catholic University of Rio de Janeiro, Rio de Janeiro, Brazil*

Keywords: Ubiquitous Computing, Intentional Systematic Software Development, Intentional Multi-Agent Systems, Ontology, Knowledge Representation, Interface Dynamic Construction, Reuse.

Abstract: In this paper, we present the development of an Intentional-MAS-Oriented Ubiquitous System driven by the FIPA Standards Ontological Support. This support contemplates the development with a certain degree of commonality. Our main goal is to improve the Intentional Systematic Software Development for further Ubiquitous Systems, by considering the same language, vocabulary, and protocols in the agents' communication and inter-operability as well as an adequate context-aware knowledge representation for different smart-spaces.

## 1 INTRODUCTION

Ontology consists of a knowledge formal representation – a specification of conceptualization (Staab and Studer 2004.) We commonly use ontology in order to specify types of conceptualizations, obtaining a simplified version of the real world based on the concepts of the domain, and the relationships between these concepts. Particularly, Ubiquitous Computing must deal with the communication among different devices, smart-spaces, and people, which are distributed (Weiser 1991) (Bell and Dourish 2006). In this scenario, it is relevant and intuitive the use of an ontological support to improve the inter-operability among different smart-spaces and their entities as well as to consequently standardized the communication between them, and the knowledge sharing in their social/interactive events.

In the last years, we have different research groups that investigate the ontology use in ubiquitous and pervasive systems (Masuoka et al. 2003) (Chen et al. 2003), proposing interesting solutions to specifically handle ubiquitous issues:

In (Christopoulou and Kameas 2005), the GAS Ontology describes the semantic of some concepts and their interdependencies based on ubiquitous environments. The authors also provide a common language for the communication involving different devices, and a service discovery mechanism. The goal of the GAS Ontology is to deal with the semantic inter-operability among heterogeneous eGadgets and the semantic service discovery. Finally, the authors discuss about the GAS Ontology manager. This support runs on each device, by managing its ontology and processing the knowledge that each device needs over time.

In (Ranganathan et al. 2003) and (Ranganathan et al. 2004), the authors present the integration of ontology and Semantic Web technology into their pervasive computing infrastructure, the GAIA smart-spaces. According to Rangnathan et al. the focus of their work was in three main issues: *Discovery and Matchmaking*; *Inter-operability between different entities*; and *Context-awareness*. Moreover, the approach followed by GAIA combines different kinds of ontology, divided in two major groups: ontology for different entities, and ontology for context information. Based on their experimental work, the authors argue the relevance of ontological support to improve the development of pervasive environments, by overcoming challenges commonly found in pervasive contexts. For example, augmenting the system services, which includes configuration management; human interfaces, components interoperation, and context sensitive behaviour.

Considering some deficiencies presented on (Ye et al. 2007) that must be addressed in order to successfully apply ontological support to the next-generation systems in Pervasive Computing and Ubiquitous Computing; and also our Ubiquitous

Serrano M. and José Pereira de Lucena C. (2010).
APPLYING FIPA STANDARDS ONTOLOGICAL SUPPORT TO INTENTIONAL-MAS-ORIENTED UBIQUITOUS SYSTEM.
In *Proceedings of the 12th International Conference on Enterprise Information Systems - Software Agents and Internet Computing*, pages 114-121
DOI: 10.5220/0002969001140121

Computing Group needs, we are focused on multi-agent communication field. Moreover, we are concerned in ontological models for intentional multi-agent systems (MASs).

Intentional MAS paradigm is an emergent technological support, in which the "like me" recognition (Gordon 2005), goal formation (Dignum and Conte 1998), and Belief-Desire-Intention (BDI) Model (Bratman 1987) (Georgeff et al. 1998) (Pokahr et al. 2005) (Bigus and Bigus 2001) are intrinsic and intense. An Intentional MAS represents an adequate support to achieve explicit ascription of mental states; an essential feature to guarantee autonomy; and a respectable philosophical model of human practical reasoning. Thus, it poses some novel challenges to deal with the communication and inter-operability among cognitive agents; between the agents and the ubiquitous environments; and between the agents and the final users.

Unfortunately, we have few research groups that consider ubiquitous systems driven by Intentional MAS in order to, for example: *(i)* manager different smart-spaces and heterogeneous access devices using the intentional agents' rationale; and *(ii)* deal with the user's satisfaction based on belief-desire-intention-oriented support, and interface dynamic construction. Furthermore, if we consider the use of ontology in the intentional-MAS-oriented ubiquitous development, it is really difficult to find work that fills this gap. This technological gap motivated us to explore an ontological support to deal with the communication among different intentional agents.

In order to achieve our goals, we applied our ontological support to validate an extensive ubiquitous dental case study. Our experimental research in the Software Engineering Laboratories at PUC-Rio and UofT demonstrates that this ontological support offers adequate resources to the developing of Intentional MAS in ubiquitous contexts, by considering some important concerns: *(i)* users' intentionality, *(ii)* changeable contexts, *(iii)* devices heterogeneity (e.g. limited or not; and mobile or not); *(iv)* distributed smart-spaces; and *(v)* service omnipresence using a stable communication protocol based on the FIPA Coder and Decoder classes for SL Language. Moreover, the support can be reused and specialized for various ubiquitous applications in different cognitive domains.

The paper is organized in sections. Section 2 presents the use and the validation of the ontological support in a ubiquitous dental system development centered on Intentional MAS. Section 3 reports on the lessons learned. Section 4 is dedicated for the concluding remarks. Finally, Section 5 describes further work.

## 2 APPLYING ONTOLOGY

As previously mentioned, we are using ontology in order to improve the Intentional Systematic Software Development for Ubiquitous Systems – ISSD for UbSystems (Serrano et al. 2009.) In ubiquitous contexts, in which the interaction between different smart-spaces is intrinsic, it is interesting that the agents communicate with other agents by considering the same language, vocabulary, and protocols. Moreover, when the communication and the contents involved into this communication are standardized, the same represented knowledge can be easily shared and reused by ubiquitous applications in different cognitive domains.

As we are following the FIPA standards defined in the JADEX Framework (Braubach et al. 2004) and the JADE-LEAP Platform (Caire 2003), our Intentional MAS already supports certain degree of commonality. This Intentional MAS specifically uses the FIPA Coder and Decoder classes for SL Language (Bellifemine et al. 2007). This language standardized the messages exchanged among the agents of the platform. However, we observed that it would be appropriate to define the agents own vocabulary and semantic to adequately deal with the contents of the exchanged messages. In other words, we must define a specific ontology for this scenario.

We have different ways to define the ontology. We decided to use the FIPA SL Codec to do this. Thus, it involves the definition of the elements, which will be transferred into the messages as extension of predefined classes. The main idea is to describe the elements that compose the exchanged messages. The FIPA SL Codec can encode and decode these messages using the FIPA format, allowing the communication among intentional agents in different smart-spaces.

According to the FIPA SL Codec, the ontology is composed of the vocabulary and the nomenclature. The vocabulary describes the concepts terminology. These concepts are used by the agents in the interaction among them. The nomenclature describes the concepts semantic and structure, and depends on the relationships among these concepts.

Figure 01 (SADT) shows the construction process of our ontology. It includes some activities from the State-Of-The-Art investigation – as well as the experimental work conduction in the Software

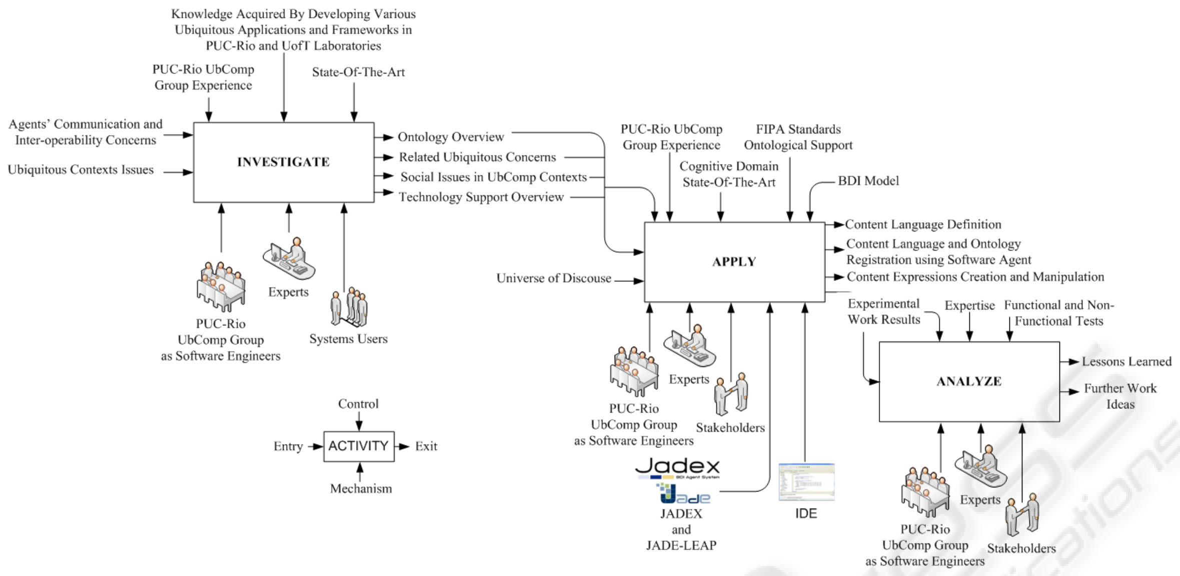

Knowledge Acquired By Developing Various Ubiquitous Applications and Frameworks in PUC-Rio and UofT Laboratories
PUC-Rio UbComp Group Experience
State-Of-The-Art
Agents' Communication and Inter-operability Concerns
Ubiquitous Contexts Issues
INVESTIGATE
Ontology Overview
Related Ubiquitous Concerns
Social Issues in UbComp Contexts
Technology Support Overview
PUC-Rio UbComp Group as Software Engineers
Experts
Systems Users
PUC-Rio UbComp Group Experience
FIPA Standards Ontological Support
Cognitive Domain State-Of-The-Art
BDI Model
Universe of Discouse
APPLY
Content Language Definition
Content Language and Ontology Registration using Software Agent
Content Expressions Creation and Manipulation
Experimental Work Results
Expertise
Functional and Non-Functional Tests
ANALYZE
Lessons Learned
Further Work Ideas
Control
Entry
ACTIVITY
Exit
Mechanism
PUC-Rio UbComp Group as Software Engineers
Experts
Stakeholders
JADEX and JADE-LEAP
IDE
PUC-Rio UbComp Group as Software Engineers
Experts
Stakeholders

Figure 1: Our ontology construction.

Engineering Laboratories at PUC-Rio and UofT for defining, registering, creating and manipulating the ontology – to the results analysis, generating the lessons learned and further work ideas.

Only to elucidate, we can consider a specific scenario based on an extensive dental case study, developed using our ISSD for UbSystems – briefly described on (Serrano et al. 2009).

Dental Scenario Description: *A patient wants to be registered in a dental clinic. Her/His personal agent, which is inside her/his device, requests the registration. The clinic agents receive the request and send the registration form.*

The first challenge in this scenario is that every decision is made by the agents at runtime. Moreover, the sent form depends on the request, user's preferences (e.g. privacy policies), devices features (e.g. interface limitations and memory and processing capacities), network specifications, and contract information. Thus, a pre-defined communication protocol must be considered in order to avoid further disagreements and mistakes. The communication is standardized using ontology.

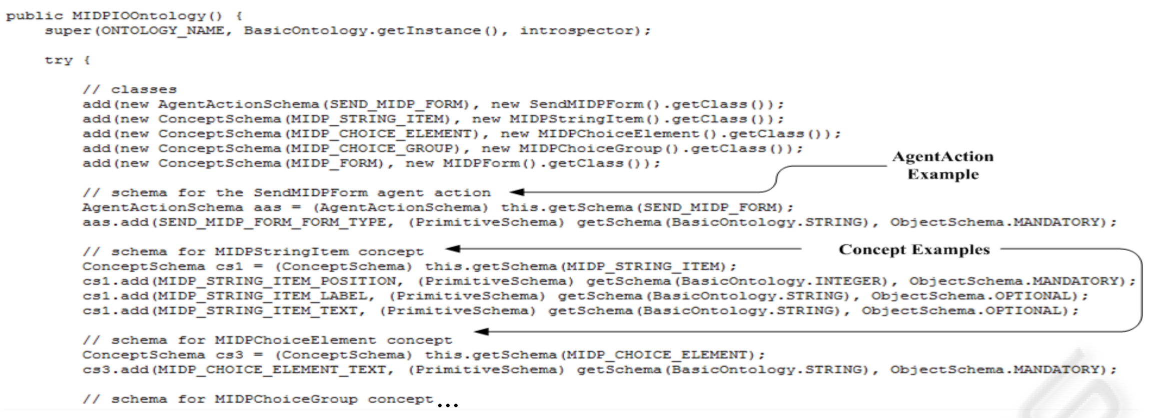
In order to implement the ontology for this scenario, we had to extend the classes *BasicOntology* and *ACLOntology*, predefined in the FIPA SL Codec, by adding the elements schemas that describe the structure of the concepts, agent actions, and predicates of the exchanged messages. The *Concept*, *AgentAction*, and *Predicate* are interfaces, which correlated classes are *ConceptSchema*, *AgentActionSchema*, and *PredicateSchema*. In fact, these interfaces have a super-class called *ObjectSchema*. As follows, we have a brief description of *Concept*, *AgentAction*, and *Predicate*:

- *Concept* represents expressions that indicate entities with a complex structure, such as: (Patient :id 000000 :name Mary :address "111 Something Street"). It means that there is a patient with the id 000000, the name Mary, and the address 111 Something Street.
- *AgentAction* represents concepts that indicate actions performed by the agents in the multi-agent systems platform, such as: (Request (Registration :clinic "Dental Clinic ABCD") (Patient :id 000000)). It means that the patient with the id 000000 requests the registration for the Dental Clinic ABCD.

*Predicate* represents expressions that inform some detail about the status of the world, such as: (Is-patient-of (Patient :id 000000) (Clinic :name "Dental Clinic ABCD")). It means that the patient with the id 000000 is patient of the clinic, which name is Dental Clinic ABCD.

We are particularly following the reference model proposed by Fabio Bellifemine, Giovanni Caire, and Dominic Greenwood in (Bellifemine et al 2007.) Figure 2 illustrates this model.

In this reference model we have the *Predicate*, *Concept*, and *AgentAction* as *Element*. Moreover, the *Predicate* is a *ContentElement*; the *Concept* is a *Term*; and the *AgentAction* is a specialization of *Concept*. A *ContentElement* can be used as a content of an ACL message. The *Term* can be an abstract entity or a concrete entity.



```
public MIDPIOOntology() {
    super(ONTOLOGY_NAME, BasicOntology.getInstance(), introspector);

    try {

        // classes
        add(new AgentActionSchema(SEND_MIDP_FORM), new SendMIDPForm().getClass());
        add(new ConceptSchema(MIDP_STRING_ITEM), new MIDPStringItem().getClass());
        add(new ConceptSchema(MIDP_CHOICE_ELEMENT), new MIDPChoiceElement().getClass());
        add(new ConceptSchema(MIDP_CHOICE_GROUP), new MIDPChoiceGroup().getClass());
        add(new ConceptSchema(MIDP_FORM), new MIDPForm().getClass());

        // schema for the SendMIDPForm agent action
        AgentActionSchema aas = (AgentActionSchema) this.getSchema(SEND_MIDP_FORM);
        aas.add(SEND_MIDP_FORM_FORM_TYPE, (PrimitiveSchema) getSchema(BasicOntology.STRING), ObjectSchema.MANDATORY);

        // schema for MIDPStringItem concept
        ConceptSchema cs1 = (ConceptSchema) this.getSchema(MIDP_STRING_ITEM);
        cs1.add(MIDP_STRING_ITEM_POSITION, (PrimitiveSchema) getSchema(BasicOntology.INTEGER), ObjectSchema.MANDATORY);
        cs1.add(MIDP_STRING_ITEM_LABEL, (PrimitiveSchema) getSchema(BasicOntology.STRING), ObjectSchema.OPTIONAL);
        cs1.add(MIDP_STRING_ITEM_TEXT, (PrimitiveSchema) getSchema(BasicOntology.STRING), ObjectSchema.OPTIONAL);

        // schema for MIDPChoiceElement concept
        ConceptSchema cs3 = (ConceptSchema) this.getSchema(MIDP_CHOICE_ELEMENT);
        cs3.add(MIDP_CHOICE_ELEMENT_TEXT, (PrimitiveSchema) getSchema(BasicOntology.STRING), ObjectSchema.MANDATORY);

        // schema for MIDPChoiceGroup concept...
```

Figure 3: Ontological Java class (*Interface Elements* Standardization) to facilitate the *Interface* agent communication/inter-operability with *Dental Domain* agents, heterogeneous devices, changeable contexts, and several *Patients* actors.

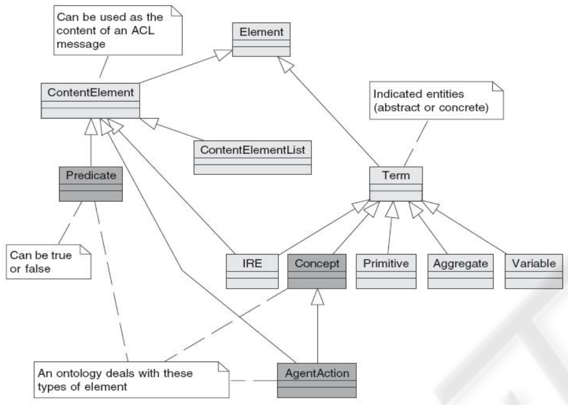

Can be used as the content of an ACL message
Element
ContentElement
Indicated entities (abstract or concrete)
ContentElementList
Predicate
Term
Can be true or false
IRE
Concept
Primitive
Aggregate
Variable
An ontology deals with these types of element
AgentAction

Figure 2: Content reference model (Bellifemine et al 2007).

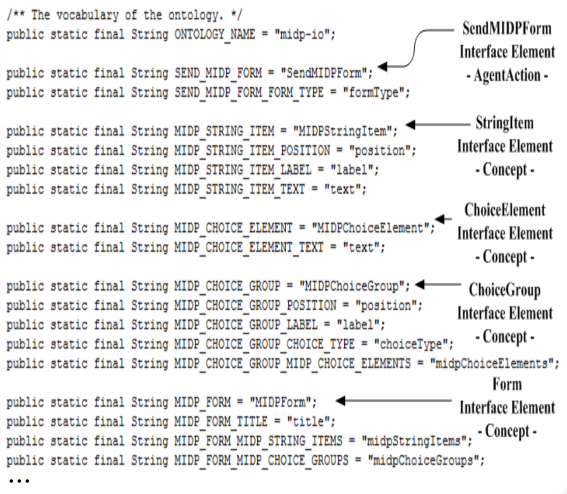
In our dental case study, we have *Elements* in the interface level; dental domain level, and dental application level. We firstly defined an ontological Java class that extends the *Ontology* class for each interface *Element* in the dental context. Each ontological Java class is declared as a singleton object as this class is normally not evolved during the agent's lifetime. For the same reason, we defined another Java class, which also extends the *Ontology* class, and contains a static method in order to access this singleton object. It means that different software agents that are in the same Java Virtual Machine can share the same ontology object.

An example of our ontological Java classes for the dental case study in the interface level is presented as a code fragment on Figure 3.

We are using the JADE-LEAP Platform execution modes (split and standalone) (Caire 2003) to deal with heterogeneous devices (MIDP and Personal JAVA). Thus, the code fragment illustrates our I/O MIDP ontology to support the communication between the *Interface* agent – located inside the MIDP device – and the *Domain* agents. Only to clarify the idea, some interface *Elements* – used by the *Interface* agent to dynamically construct dental forms that will be presented to the user using her/his own device and according to the user's preferences and the devices features (e.g. memory/processing capacities, screen size, and resolution) – are:

- *SendMIDPForm* agent action: is the ontological representation for an action performed by the *Interface* agent in order to send a form.
- *MIDPStringItem* concept: is an ontological concept that describes a *StringItem* element, which can be used to compose the *Form*, by representing a *spring*.
- *MIDPChoiceElement* concept: is an ontological concept that describes a *ChoiceElement*, which can be used to compose the *ChoiceGroup*, by representing the alternative text.
- *MIDPChoiceGroup* concept: is an ontological concept that describes a *ChoiceGroup*, which can be used to compose the *Form*, by representing a group of choices. Moreover, it can be composed of one or more *ChoiceElement(s).*
- *MIDPForm* concept: is an ontological concept that describes a *Form*, which can be composed of zero or more *StringItem(s)*, and zero or more *ChoiceGroup(s).*

We can observe that the ontological Java class basically contains the constructor and the structures of the schema for different *Concepts*, *AgentActions*, and *Predicates*. Each element in a schema has a name and a type. An element can be declared as "OPTIONAL" or "MANDATORY." An "OPTIONAL" element means it can assume a "null" value. On the other hand, a "MANDATORY" element means that an *OntologyException* will be thrown if a "null" value was found. An element in a schema can also be a list, in which, for example, the cardinality of this element is zero or more *String* type elements. The ontological Java class implements the Vocabulary Java class, which code fragment is illustrated on Figure 4.



```
/** The vocabulary of the ontology. */
public static final String ONTOLOGY_NAME = "midp-io";

public static final String SEND_MIDP_FORM = "SendMIDPForm";
public static final String SEND_MIDP_FORM_FORM_TYPE = "formType";

public static final String MIDP_STRING_ITEM = "MIDPStringItem";
public static final String MIDP_STRING_ITEM_POSITION = "position";
public static final String MIDP_STRING_ITEM_LABEL = "label";
public static final String MIDP_STRING_ITEM_TEXT = "text";

public static final String MIDP_CHOICE_ELEMENT = "MIDPChoiceElement";
public static final String MIDP_CHOICE_ELEMENT_TEXT = "text";

public static final String MIDP_CHOICE_GROUP = "MIDPChoiceGroup";
public static final String MIDP_CHOICE_GROUP_POSITION = "position";
public static final String MIDP_CHOICE_GROUP_LABEL = "label";
public static final String MIDP_CHOICE_GROUP_CHOICE_TYPE = "choiceType";
public static final String MIDP_CHOICE_GROUP_MIDP_CHOICE_ELEMENTS = "midpChoiceElements";

public static final String MIDP_FORM = "MIDPForm";
public static final String MIDP_FORM_TITLE = "title";
public static final String MIDP_FORM_MIDP_STRING_ITEMS = "midpStringItems";
public static final String MIDP_FORM_MIDP_CHOICE_GROUPS = "midpChoiceGroups";
...
```

SendMIDPForm Interface Element - AgentAction -
StringItem Interface Element - Concept -
ChoiceElement Interface Element - Concept -
ChoiceGroup Interface Element - Concept -
Form Interface Element - Concept -

Figure 4: Ontology vocabulary for *Interface Elements*.

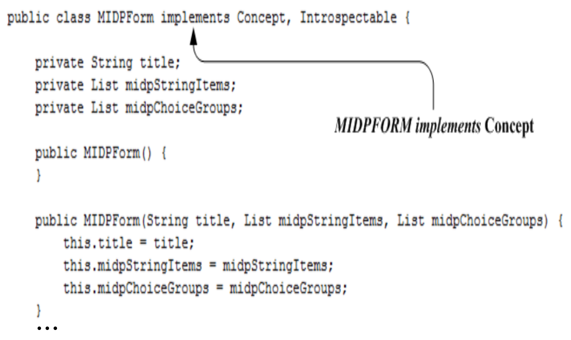
Based on the FIPA Coder and Decoder classes for SL Language, another important consideration is that each schema must implement its proper interface, as follows:

- for *ConceptSchema*, the class must implement the *Concept* interface.
- for *AgentActionSchema*, the class must implement the *AgentAction* interface.
- for *PredicateSchema*, the class must implement the *Predicate* interface.

In order to exemplify this consideration, Figure 5 shows part of the code based on the *MIDPForm Concept*.

As presented on (Bellifemine et al 2007), the next three steps are necessary to conclude the ontology: *(i)* define the content language; (ii) register the content language and the ontology using a software agent; and *(iii)* create or manipulate the content expressions as Java Objects.

The first step consists on defining the content language. Using the FIPA Coder and Decoder we have the possibility to choose the SL Language or the LEAP language. It is also possible to develop an agent that uses a proper language by implementing the *jade.content.lang.Codec* interface. The SL Language is a human-readable content language, which content expression is a *string*.

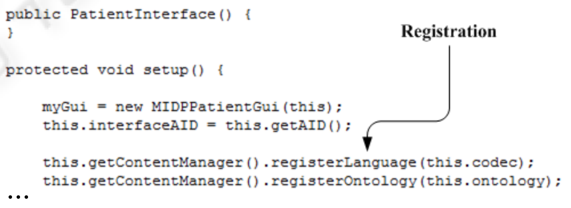


```
public class MIDPForm implements Concept, Introspectable {

    private String title;
    private List midpStringItems;
    private List midpChoiceGroups;

    public MIDPForm() {
    }

    public MIDPForm(String title, List midpStringItems, List midpChoiceGroups) {
        this.title = title;
        this.midpStringItems = midpStringItems;
        this.midpChoiceGroups = midpChoiceGroups;
    }
...
```

Figure 5: *MIDPForm* Concept ***implements*** Concept.

The LEAP language is a non-human-readable content language which content expression is a sequence of bytes. Moreover, the LEAP language is lighter than the SL language. This feature is particularly interesting in strong memory and processing limitations. In our dental case study we used the SL language.

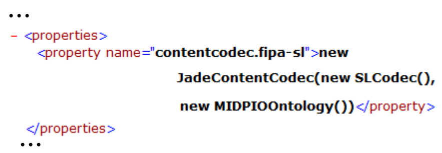
The second step consists on registering the content language and the ontology using a software agent. Normally, in behaviour-based agents, this registration is performed in the agent *setup()* method as presented on Figure 6 for the *PatientInterface* agent – a JAVA code fragment. As this *Interface* agent runs inside the MIDP device, we decided to use a "light" agent, based on behaviour to avoid problems with the device memory and processing limitations.



```
public PatientInterface() {
}

protected void setup() {

    myGui = new MIDPPatientGui(this);
    this.interfaceAID = this.getAID();

    this.getContentManager().registerLanguage(this.codec);
    this.getContentManager().registerOntology(this.ontology);
...
```

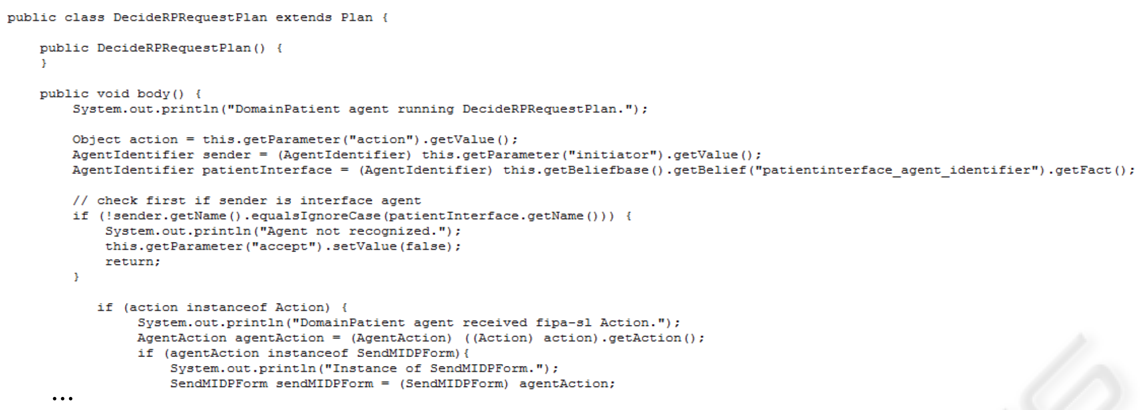
Figure 6: Registering content language and ontology using our behaviour-base *PatientInterface* agent.

However, as we are using intentional agents in the domain level to improve the cognition capacity, the "like me" recognition, and the goal formation, we also registered the content language and the ontology according to the JADEX specifications and the BDI Notation as shown on Figure 7 – XML code fragment of the *DomainPatient* agent (*property tag.*)



```
...
- <properties>
    <property name="contentcodec.fipa-sl">new
                        JadeContentCodec(new SLCodec(),
                        new MIDPIOOntology())</property>
  </properties>
...
```

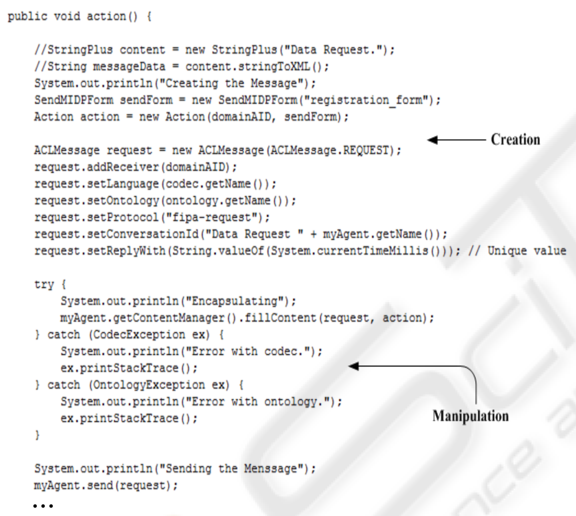
Figure 7: Registering content language and ontology using our intentional-oriented *DomainPatient* agent.



```
public class DecideRPRequestPlan extends Plan {

    public DecideRPRequestPlan() {
    }

    public void body() {
        System.out.println("DomainPatient agent running DecideRPRequestPlan.");

        Object action = this.getParameter("action").getValue();
        AgentIdentifier sender = (AgentIdentifier) this.getParameter("initiator").getValue();
        AgentIdentifier patientInterface = (AgentIdentifier) this.getBeliefbase().getBelief("patientinterface_agent_identifier").getFact();

        // check first if sender is interface agent
        if (!sender.getName().equalsIgnoreCase(patientInterface.getName())) {
            System.out.println("Agent not recognized.");
            this.getParameter("accept").setValue(false);
            return;
        }

           if (action instanceof Action) {
                System.out.println("DomainPatient agent received fipa-sl Action.");
                AgentAction agentAction = (AgentAction) ((Action) action).getAction();
                if (agentAction instanceof SendMIDPForm){
                    System.out.println("Instance of SendMIDPForm.");
                    SendMIDPForm sendMIDPForm = (SendMIDPForm) agentAction;
      …
```

Figure 10: Creating/manipulating the content expressions using our *DomainPatient* agent (*ExecuteRPRequestPlan*).

The third step consists on creating and manipulating the content expressions as Java Objects. Figure 8 shows the code fragment about this step using our *PatientInterface* agent.



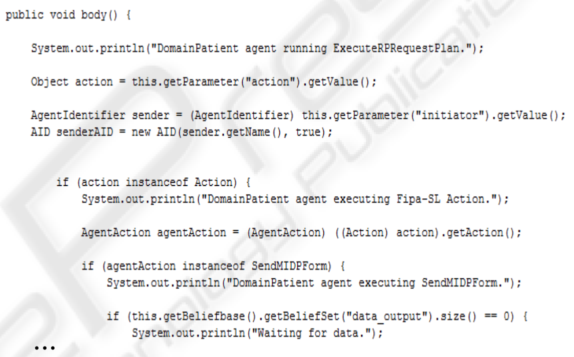
```
public void action() {

    //StringPlus content = new StringPlus("Data Request.");
    //String messageData = content.stringToXML();
    System.out.println("Creating the Message");
    SendMIDPForm sendForm = new SendMIDPForm("registration_form");
    Action action = new Action(domainAID, sendForm);

    ACLMessage request = new ACLMessage(ACLMessage.REQUEST);
    request.addReceiver(domainAID);
    request.setLanguage(codec.getName());
    request.setOntology(ontology.getName());
    request.setProtocol("fipa-request");
    request.setConversationId("Data Request " + myAgent.getName());
    request.setReplyWith(String.valueOf(System.currentTimeMillis())); // Unique value

    try {
        System.out.println("Encapsulating");
        myAgent.getContentManager().fillContent(request, action);
    } catch (CodecException ex) {
        System.out.println("Error with codec.");
        ex.printStackTrace();
    } catch (OntologyException ex) {
        System.out.println("Error with ontology.");
        ex.printStackTrace();
    }

    System.out.println("Sending the Menssage");
    myAgent.send(request);
    …
```

Figure 8: Creating/manipulating the content expressions as Java objects using our *PatientInterface* agent.

Again, in order to create and manipulate the content expressions using intentional agents, we extended the *Plan* class specified on the JADEX documentation and we also implemented the *DecideRPRequestPlan* and the *ExecuteRPRequestPlan* as plans of our *DomainPatient* agent. Figures 9 and 10 respectively present code fragments of these plans.

## 3 LESSONS LEARNED

We have applied a FIPA-Standards-based



```
public void body() {

    System.out.println("DomainPatient agent running ExecuteRPRequestPlan.");

    Object action = this.getParameter("action").getValue();

    AgentIdentifier sender = (AgentIdentifier) this.getParameter("initiator").getValue();
    AID senderAID = new AID(sender.getName(), true);


        if (action instanceof Action) {
            System.out.println("DomainPatient agent executing Fipa-SL Action.");

            AgentAction agentAction = (AgentAction) ((Action) action).getAction();

            if (agentAction instanceof SendMIDPForm) {
                System.out.println("DomainPatient agent executing SendMIDPForm.");

                if (this.getBeliefbase().getBeliefSet("data_output").size() == 0) {
                    System.out.println("Waiting for data.");
    …
```

Figure 9: Creating/manipulating the content expressions using our *DomainPatient* agent (*DecideRPRequestPlan*).

ontological support to Intentional-MAS-oriented ubiquitous systems. Our experimental research contributes to the systematic development of Intentional-MAS-oriented ubiquitous systems as it has shown that ontology improves the inter-operability and the communication among intentional autonomous entities in ubiquitous environments. Among other contributions, our efforts provide a reuse-based standard support to specify the concepts, agent-actions and predicates based on the cognitive domain of the ubiquitous system-to-be.

As applied to our extensive dental case study, and reported on this paper using a scenario based on this case study, the proposed ontological support can be extended to deal with Multiple-Multiplicity in Ubiquitous Context (Tigli et al. 2009): *many* people accessing *many* applications/services, using *many* devices, communicating with *many* people located in *many* places, considering *many* issues, and so on.

As our ontological support is centered on intentional software agents, incremental updates on their beliefs bases – including insertion, exclusion, and modification operations – are simple to be performed. Moreover, FIPA-Standards-based ontological support deals with privacy and access control issues – intrinsic concerns in Ubiquitous Computing - by contemplating the notion of a personal agent into the agent's communication ontological model. Each final user – in our case study, each patient – has a personal and cognitive agent that represents her/him in the smart-spaces. This agent is responsible for achieving the user's goals, based on her/his profile (e.g. privacy policies, and preferences); and for controlling the user's data access by other users, agents, and organizations. If it is desired by the final user, the access negotiation process as well as the user's knowledge sharing can be "invisible" for the users. The process complexity invisibility is a concern in Ubiquitous Computing – idealized by Mark Weiser in his seminal paper (Weiser 1991) – that we try to deal with using intentional agents, standardize communication and inter-operability protocols.

## 4 CONCLUSIONS

We presented our first efforts to improve the systematic development of ubiquitous systems. These efforts consist on an extensive experimental research, in which we apply the FIPA standards ontological support to the development of Intentional MAS-oriented ubiquitous systems.

The work was performed in the Software Engineering Laboratories at PUC-Rio and UofT, where our Ubiquitous Computing group conducted experiments by developing ubiquitous systems in different cognitive domains (Serrano et al. 2008) (Serrano et al. 2009) (Serrano and Lucena 2010).

In this paper, we particularly described the ontology construction process – from the investigation activities to the definition, registration, creation and manipulation of the ontological model using a dental scenario, *Interface* and *Domain* agents, and *Interface Elements*. Moreover, we presented the use of this ontological support in a dental case study, which contemplated/addressed some important concerns of ubiquitous environments such as device heterogeneity, smart-spaces distribution, services omnipresence, and users' satisfaction.

We also enriched our contributions by incorporating this technological support in our intentional agent-oriented approach – briefly presented on (Serrano et al. 2009) – as a building block composed of the ontology developed by our group for *Interface Level*, *Domain Level*, and *Application Level*, following the same construction process presented in this paper with the *I/O MIDP Ontology* code fragments. Our intention is to provide a reuse-oriented support that aims the requirements and software engineers teams in the multi-agent communication standardization, knowledge representation and sharing, and ubiquitous-issue-based and context-aware interface dynamic construction.

## 5 FURTHER WORK

We are combining our ontological support with a dynamic database to appropriately store the knowledge in Ubiquitous Systems (Serrano and Lucena 2010). Among other contributions/benefits, the dynamic structure can deal with constant changes in ubiquitous scenarios – incorporation of new technologies on devices, or changes on the users' preferences and intentions, or even differences in the network features.

In this scenario, the knowledge can dynamically be created, modified, deleted, shared, and reused by multi-agents in a common agent-oriented platform supported by the ontology briefly presented here.

As special policies are needed in order to maintain the security and privacy (Campbell et al. 2002) (Kagal et al. 2004) of this knowledge, we are also developing a layer structure mainly centered on the concerns: security, privacy, and stakeholders' satisfaction (Serrano and Lucena 2010). The combination of this layer structure, the proposed ontological set, and the dynamic database model composes our building block for the Intentional Systematic Software Development of Ubiquitous Systems (ISSD for UbSystems.)

Furthermore, we are interested in experimental work to evaluate the SOUPA (Standard Ontology for Ubiquitous and Pervasive Applications) (Chen et al. 2004) in our ubiquitous projects, and to compare SOUPA and FIPA standard ontology.

## REFERENCES

Bell, G.; Dourish, P. (2006) Yesterday's tomorrows: notes on ubiquitous computing's dominant vision. Pers Ubiquit Comput, DOI 10.1007/s00779-006-0071-x, Springer-Verlag London Limited.

Bellifemine, F.; Caire, G.; Grenwood, D. (2007) Developing Multi-Agent Systems with JADE. Wiley Series in Agent Technology, John Wiley & Sons Ltd, ISBN-13: 978-0-470-05747-6, 286 pages.

Bigus, J. P. and Bigus, J. (2001) Constructing Intelligent Agents Using Java: Professional Developer's Guide. *2nd Edition*. ISBN 10:047139601X, ISBN 13:9780471396017, John Wiley & Sons.

Bratman, M. (1987) Intention, Plans, and Practical Reason. Harvard University Press, Cambridge.

Braubach, L., Pokahr, A. and Lamersdorf, W. (2004) Jadex: A Short Overview. In Net. ObjectDays'04: AgentExpo.

Caire, G. (2003) LEAP User Guide. TILAB, December.

Campbell, R.; Al-Muhtadi, J.; Naldurg, P.; Sampemanel, G.; Mickunas, M. D. (2002) Towards security and privacy for pervasive computing. *In Proceedings of Inter. Symposium on Software Security*, Tokyo.

Chen, H.; Finin, T.; Joshi, A. (2003) An ontology for context aware pervasive computing environments. Knowledge Engineering Review - *Special Issue on Ontologies for Distributed Systems*, ISSN: 0269-8889, Volume 18, Issue 3, Cambridge University Press, September.

Chen, H.; Perich, F.; Finin, T.; Joshi, A. (2004) SOUPA: Standard Ontology for Ubiquitous and Pervasive Applications*. International Conference on Mobile and Ubiquitous Systems*: Networking and Services (MobiQuitous'04,) pp. 258-267, Boston, August.

Christopoulou, E.; Kameas, A. (2005) GAS Ontology: an ontology for collaboration among ubiquitous computing devices. *Inter. Journal of Human-Computer Studies*, Volume 62, Issue 5, Pages 664-685, May.

Dignum, F; Conte, R. (1998) Intentional agents and goal formation. Intelligent Agents IV Agent Theories, Architectures, and Languages, *Springer Berlin/Heidelberg, ISBN*: 978-3-540-64162-9, Volume 1365/1998, pp. 231-243.

Georgeff, M; Pell, B.; Pollack, M.; Tambe, M.; Wooldridge, M. (1998) The Belief-Desire-Intention Model of Agency. In Proceedings ofthe 5th *International Workshop on Intelligent Agents: AgentTheories, Architectures, and Languages* (ATAL-98), J. Muller, M.P. Singh and A. S. Rao (eds.), 1999, pp. 1-10, *Springer-Verlag:Heidelberg*, Germany.

Gordon, R. M. (2005) Intentional Agents Like Myself. Perspectives on Imitation: From Mirror Neurons to Memes. Hurley, S. & Chater, N., *MIT Press, Chapter 15*.

Kagal, L.; Parker, J.; Chen, H.; Joshi, A.; Finin, T. (2004) Security, Trust and Privacy in Mobile Computing Environments. *Mobile Computing Handbook, Chapter 40*, ISBN: 9780849319716, pages 961-986, CRC Press, December.

Masuoka, R.; Labrou, Y.; Parsia, B.; Sirin, E. (2003) Ontology-Enables Pervasive Computing Applications, IEEE Intelligent Systems, pp 68-72, Sept/Oct.

Pokahr, A.; Braubach, L.; Lamersdorf, W. (2005) Jadex: A BDI Reasoning Engine. *Multiagent Systems, Artificial Societies, and Simulated Organizations, ISSN*: 1568-2617, Volume 15, pp. 149-174.

Ranganathan, A.; McGrath, R.; Campbell, R.; Mickunas, D. (2003) Ontologies in a Pervasive Computing Environment. Workshop on Ontologies in Distributed Systems at IJCAI, Acapulco, Mexico.

Ranganathan, A.; McGrath, R.; Campbell, R.; Mickunas, D. (2004) Use of Ontologies in a Pervasive Computing Environment*, Knowledge Engineering Review*, vol. 18, no. 3, pp. 209–220.

Serrano, Milene; Serrano, Maurício; Lucena, C. J. P. (2008) Framework for Content Adaptation in Ubiquitous Computing Centered on Agents Intentionality and Collaborative MAS. Fourth *Workshop on Software Engineering for Agent-oriented Systems (SEAS'08)*, 12 pages.

Serrano, Milene; Serrano, Maurício; Lucena, C. J. P. (2009) Ubiquitous Software Development Driven by Agents' Intentionality. *11th International Conference on Enterprise Information Systems (ICEIS'09)*, vol. SAIC, pp. 25-34, Milan, Italy, May.

Serrano, M.; Lucena, C. J. P. (2010) Agent-Oriented Dynamic Database for Intentional Development of Ubiquitous Systems*. To be submitted for IDEAS'10* on March.

Staab, S.; Studer, R. (2004) Handbook on Ontologies. *Springer-Verlag, ISBN 3-540-40834-7*, 500 pages, January.

Tigli, J-Y.; Lavirotte, S.; Rey, G.; Hourdin, V.; Cheung-Foo-Wo, D.; Callegari, E.; Riveill, M. (2009) WComp middleware for ubiquitous computing: Aspects and composite event-based Web services. Ann. Telecommun. 64:197–214, DOI 10.1007/s12243-008-0081-y, *Institut TELECOM and Springer-Verlag* France.

Weiser, M. (1991) The computer for the 21st Century. Sci Am 265(3):94–104.

Ye, J.; Coyle, L.; Dobson, S.; Nixon, P. (2007) Ontology-based models in pervasive computing systems. *The Knowledge Engineering Review*, ISSN:0269-8889, Volume 22, Issue 4, pp. 315-347, December.