Implicit Context-Sensitive Mobile Computing Using Semantic Policies

Hamid Harroud, Ahmed Karmouch

Multimedia & Mobile Agent Research Laboratory,
School of Information Technology & Engineering (SITE), University of Ottawa,
161 Louis Pasteur St. Ottawa, ON, Canada K1N 6N5, Canada
{hharroud, karmouch}@site.uottawa.ca
http://deneb.genie.uottawa.ca

Abstract. The availability of portable computing devices and advances in wireless networking technologies have contributed to the growing acceptance of mobile computing applications and opened the door for the possibility of seamless and pervasive services in mobile environments. However, with the dilemmas of limited device capabilities, network connectivity, transmission range and frequent changes, due to users and/or devices mobility, Internet-oriented wireless applications face challenges in terms of computation and interaction. These challenges elaborate with the inevitable demand for such applications to adapt to users' situations, their profiles, and the resources provided by the operating environment. Taking into account contextual information is an essential ingredient to cope with the frequent changes in mobile environments and hence provide adequate solutions for achieving adaptability, reliability, and seamless service provisioning. This paper introduces the use of policies over semantically modeled context as a technique for generating implicit context information and making use of context in mobile computing in general and internet-computing in specific. A context-sensitive instant messaging application demonstrates our proposed model using agent technology.

1 Introduction

Current dynamism in life is driving people's behaviour towards mobility and towards the usage of wireless devices. These devices are getting more powerful in terms of hardware characteristics such as power and memory as well as computing capabilities. Consequently, applications and services are developed to bring user's desktop environment to her/his hand-held devices. Some of these applications include emailing, chatting, web browsing, online banking, distributed games, multimedia streaming and recently web services [1]. In fact, services designed for use in mobile computing are expected to experience rapid growth. This transition from the desktop environment to the portable devices environment did not come free of problems but is overwhelmingly loaded with challenges such as content adaptation, mobility management, network reliability, and application interoperability to name but few.

Mobile computing needs solutions for providing dynamic interaction between users and applications on one hand and solutions for autonomic adaptation to users and environments context on the other hand.
Context-aware computing has been advocated as a mechanism for providing such dynamic interactions and autonomic adaptation. Context awareness refers to the ability of an application or a service to adapt its behaviour in response to environmental changes. Accordingly, we designed a multi-agent system, MAS, that provides interoperability, adaptability and seamless service provisioning to users in mobile computing environment. MAS system is built on the concept of semantic modeling of context, its inference to policies, and the usage of inferred context policies enforcement.
Semantic can be defined as the study of meaning or change of meaning. When mapped to Internet applications it represents the ability of the applications to understand what they are performing as services, how to implicitly adapt to changes in the environment and how to interact with other entities.
Policies deal with the methods of manipulating applications and their profiles according to current situation. This includes where and how to store these profiles, how to automate adaptation, how to provide contextual access rights and how to retrieve relevant information at the right time, at the right place and to the right entity.
Our goal is to provide a model and infrastructure to support the development of context-sensitive applications and services by generating new contextual information using the semantic modeling of acquired context (*implicit context information*), and by inferring semantic context policies which, in turn, dynamically adapt the behaviour of different entities and services by enabling and executing appropriate policy actions (*implicit use of context*). In addition, the use of context level agreements (CLA) in the form of context negotiation [2] enables the ability to restrict an application agent's context to a subset of its specific policies. The CLA guides the interaction and information exchange among agents as they receive the agreed context in the form of semantic policies and make decisions accordingly on users' behalf.
The rest of the paper is organized as follows. Section 2 describes the multi-agent system in terms of agents' role and algorithms used. Section 3 discusses the automation process of manipulating contextual information and the semantic policies inference used by the system agents. Section 4 demonstrates an instant messaging application developed using the MAS. Section 5 presents the related work. And finally, section 6 concludes our work and highlights the future work.

2 Multi-agent System for Context Provisioning

We consider agents as the main computational entities of our system, as well as of applications and services requesting context information. Agents are provided with control abilities to govern their behavior and to make decisions autonomously by means of policies.
The introduction of policies inside agents increase their pro-activeness in facing new situations occurring in the surroundings due to users' mobility, devices' capability constraints, possibility of discontinuous interconnection and changes in the environ-

ment. The separation of the agents' coding logic (i.e. tasks) from their governing behaviors (i.e. policies) allows the reuse of agents' strategies and their overall behavior in order to dynamically adapt to various context information without significant impact on the mechanism used for their implementation.
An agent becomes context-sensitive, meaning that it can sense changes in its context and adapt its behavior in response, by making use of semantic context policies.
Figure 1 depicts the proposed agents, context sources, presence interpretation and inference process. The proposed MAS system consists of two main agents: the context management agent and the reasoner agent.

2.1 The Context Management Agent

The context management agent (CMA) is responsible for monitoring on-going context-based sessions and managing the environment resources. The CMA has the authority to cancel, modify and/or renegotiate context specifications according to changes that might occur in the environment, or if agents violate the tasks assigned to them during negotiation [2]. Agents register with the CMA both to project their presence to other agents in the environment and to obtain the authorization policies needed to use available services and have access to other agents and information. The CMA stores relevant information in a knowledge base repository for inference, consistency maintenance and knowledge sharing. It models context and system information using ontologies implemented using the Ontology Web Language (OWL) [3].
It formulates context specifications from agents' requests and profiles according to the designed ontology constructs, axioms and rules of composition.

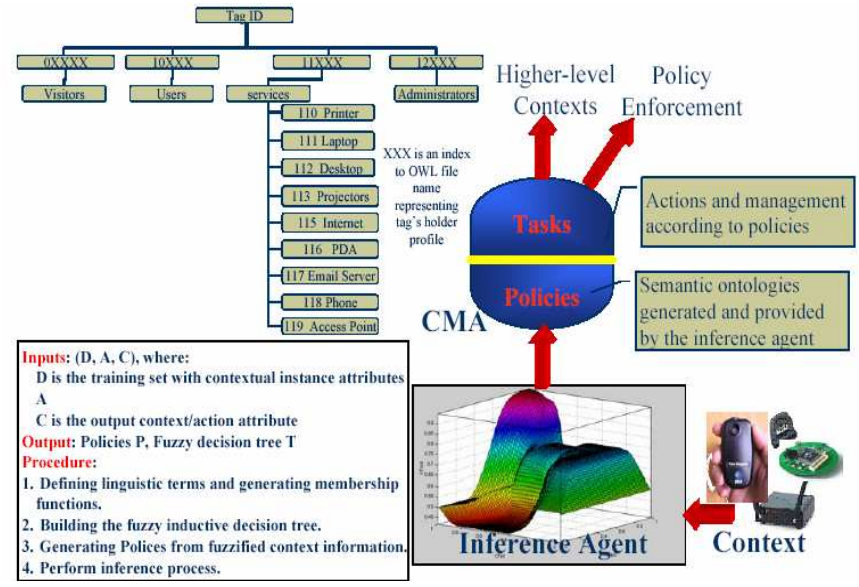


Figure 1. The multi-agent system model.

2.2 The Reasoner Agent (RA)

The RA contains novel inference machinery that integrates logical reasoning, fuzzy reasoning and semantic rule representation in one system. Context captured from sensors and from users' and services' profile is treated as facts in the inference process. The reasoner agent uses these facts to deduce new context information and/or trigger actions. The reasoner agent uses logic-reasoning to insure that captured contexts are consistent both with each other and with constructs defined in the context ontology. It uses fuzzy logics to cope with the uncertainty in captured context and generates implicit context information which is then issued to different entities in the environment requesting context.

The MAS uses an RF-tag system for acquiring contextual information. We extended the RF-tags to provide other contexts than just identity and location. By taking advantage of the unique ID value of each tag, ontology profiles of entities are indexed in the system repository using these ID values. These ontology profiles include information such as activities of the user holding the tag, capabilities of the service with the assigned tag and the policy profile generated for each entity holding a tag in the environment. Figure 1 shows the categories we developed for the different entities in the environment and how they are associated to the tag IDs. For example a detected tag with the ID 111754 is automatically interpreted by the CMA to be a device of type laptop. From the detected tag it knows the laptop location while it uses the Tag ID number to fetch the OWL profile from the repository and thus gains knowledge of its implicit context and its governing policies.

The RA also uses these OWL profiles along with the inference algorithm to build an inductive fuzzy inference decision tree capable of extracting features embedded in the contextual information and inferring policies that will be used by the CMA for monitoring and governing the environment and entities interaction.

3 The Automation Process of Context Provisioning

As described in our work detailed in [4], we divided the process of automating context provisioning into three parts: generating implicit contextual information, inferring context to semantic policies governing the system's behaviour and supporting entities at runtime in accordance to the inferred policies. The first process requires a semantic modeling of context so that generating implicit contextual information could be preformed based on explicit context. Context inference is performed by transforming the ontology-based context instances to a set of linguistic terms over the context's range of values. Each term has a certain membership function to be used instead of the original context instance. As shown in figure 2, each entity in the environment (i.e. user, device, service …) is associated with two configuration files: the Agent Configuration File (ACF) and the Context Information File (CIF). ACF holds information about the agent that represents the entity (i.e. contact information for a user and the service description for a service with the reference to the CIF file location). These files

are used by the inference agent to provide semantic policies and by the CMA to enforce and manage these policies.

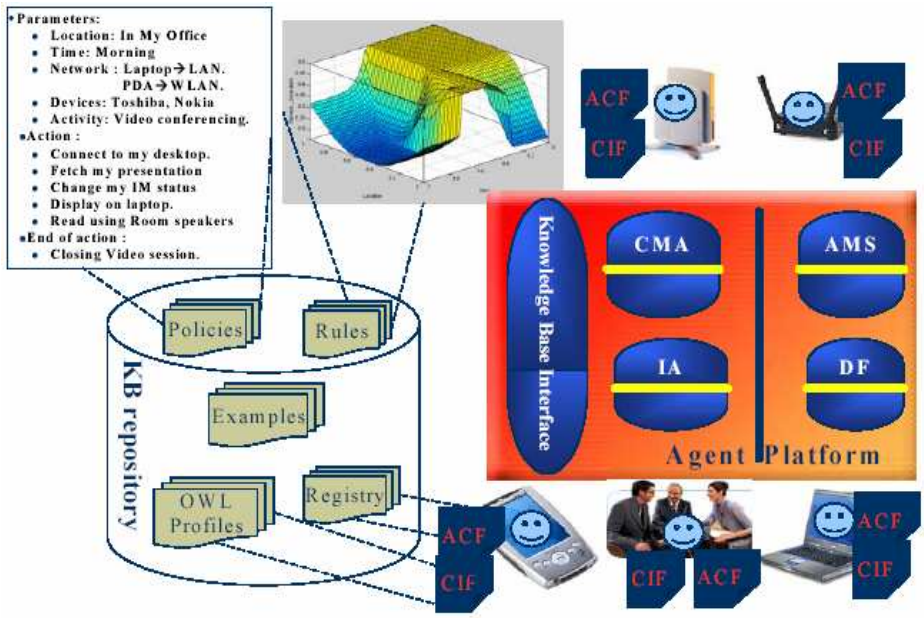


Figure 2 Components used in the automation of context provisioning.

A relation between a device and its owner is a sample example that illustrates the generation of implicit context. A crisp relation will have the following ontological representation.

```
<DeskTop rdf:ID="MMARL-10">
        <hasLocation rdf:resource="&Loc;B502"/>
        <belongsTo rdf:resource="& Confoaf ;Bob"/>
        ….
</DeskTop>
```

The captured context indicates that MMARL-10 is a device of type DeskTop, it is located in B502 and it belongs to user Bob. This information is acceptable if user Bob is actually the only person who is using this desktop all the time. But he might be using it only during the morning hours which will make this fact inaccurate and might cause unacceptable actions. When fuzzy logic is used in this situation to represent the uncertainty in the relation between the desktop and the user, the semantic representation will include the property hasSimilarity that will link the crisp context ontology to fuzzy ontology, then linguistic terms are defined that describe the membership functions used in the relation.

```
<Fuzz:hasFuzzyTerm >
 <Fuzz:FuzzyTerm rdf:ID="Morning">
  <Fuzz:hasMembershipFn>
   <Fuzz:trapezoidal>
    <Fuzz:hasFirstpnt>9.0</Fuzz:hasFirstpnt>
    <Fuzz:hasEndPoint>12.0</Fuzz:hasEndPoint>
    <Fuzz:hasScndpnt>11.0</Fuzz:hasScndpnt>
    <Fuzz:hasStartPoint>8.0</Fuzz:hasStartPoint>
   </Fuzz:trapezoidal>
 </Fuzz:hasMemebrshipFn>
 </Fuzz:FuzzyTerm>
```

```
<Fuzz:hasFuzzyTerm>
 <Fuzz:FuzzyTerm rdf:ID="Afternoon">
  <Fuzz:hasMembershipFn>
   <Fuzz:triangular>
    <Fuzz:hasCenter>13.0</Fuzz:hasCenter>
    <Fuzz:hasEndPoint>15.0</Fuzz:hasEndPoint>
   <Fuzz:hasStartPoint>11.0</Fuzz:hasStartPoint>
   </Fuzz:triangular>
  </Fuzz:hasMembershipFn>
 </Fuzz:FuzzyTerm>
</Fuzz:hasFuzzyTerm>
```

```
<Fuzz:hasFuzzyTerm>
  <Fuzz:FuzzyTerm rdf:ID="Night">
     <Fuzz:hasMembershipFn>
       <Fuzz:Crisp rdf:ID="Zero"/>
     </Fuzz:hasMembershipFn>
  </Fuzz:FuzzyTerm>
</Fuzz:hasFuzzyTerm>
```

In this example, we defined three linguistic terms that describe the temporal relation between the desktop and the user. The first term is the “Morning” that defines the degree of certainty that Bob is using MMARL-10 during the morning from 8:00 AM till 12:00 PM. It is represented as a trapezoidal function which indicates that between 9:00 and 11:00 the system is certain that MMARL-10 belongs to user Bob. The second term is the “AfterNoon” which is described as a triangular function between 11:00 AM and 3:00 PM and the last term is the “Night” which is described as a crisp function with value equal to zero. This means that MMARL-10 is not belonging to Bob during the night.

With the aforementioned example that clarifies our intention of using fuzzy ontology to represent uncertainty in the context information, the next subsections describe the steps performed by the inference agent for reasoning about implicit context information.

3.1 Defining linguistic terms and generating membership functions.

Defining the linguistic terms and membership functions lie with our inference agent as users and application developers are assumed not to be expert in fuzzy logic and domain mapping techniques. Application developers will use the designed fuzzy-API to specify the number of linguistic terms to be used by the inference agent to map the context construct to them. For example, a developer may state that the location context inside a meeting room is specified by two linguistic terms “near” and “far”.

These two linguistic terms need to be mapped to the standard membership functions defined in the fuzzy ontology and stored in the system. For example, the two linguistic terms can be represented by Gaussian functions that have mean and variance determined by the range of the location and the overlap region. To generate the required membership functions, the context ontology is first used to check the range and the domain of the context construct and provides the result to the inference agent. The inference agent then uses a simple algorithm for membership function generation to

achieve the low overhead in computation requirements. The algorithm starts by dividing the context range into 2n+2 equal slots, where n is the number of linguistic terms required (in the location example above, n=2). It then assigns four slots to each linguistic term; each adjacent terms share one slot in common that will represent the overlapping region in the fuzzy sets. The four slots concept is chosen because of the trapezoidal function that is considered to be the general membership function from which other membership functions can be derived. After successfully dividing the context range into its 2n+2 regions, the CMA randomly selects samples from each region and requests from the application developer, or user, to specify the expected membership function for each of these values. This step is required in order for the inference agent to correctly deduce the best membership functions that resemble user expectation. After this decision, the inference agent applies the Lagrange's interpolation mechanism on these samples data followed by Euclidian similarity measure to decide the standard membership functions that best resemble user-defined data.

3.2 Building the fuzzy inductive decision tree

After deciding the membership functions for the context information, the inference agent begins to build the fuzzy decision tree [5], shown at the bottom right in figure1, to perform the second process in the automation. Fuzzy decision trees improve the inference, robustness and generalization of classification, action triggering, and decision-making in context aware environments. In general, building an inductive decision tree requires a set of training examples, a heuristic method for choosing the best attribute to split at any node, and an inference process mechanism. Training examples in our system are acquired either from experts in the application domain or from sensors and historical actions of users. These examples are saved in the knowledge base and can be manipulated through the knowledge base API. The inference agent uses a heuristic splitting criterion that tries to predict the classification of context attributes. These values are used in conjunction with the entropy measurement to decide the winning attribute to use for splitting at any node.

The heuristic method used by the inference agent has two fold contributions. First, it reduces the example space through which the computation is performed, thus reducing complexity and increase speed of convergence. Second, it reduces the effect of a dominant linguistic term in the calculation because repeated values are counted using the fuzzy projection and then averaged. The root of the tree contains all the training examples. It represents the whole description space since no restrictions are imposed yet. Each node is recursively split by partitioning its examples. A node becomes a leaf when either its samples come from a unique class or when all attributes are used on the path. When it is decided to further split the node, one of the remaining attributes (i.e., not appearing on the current path) is selected. The examples present in the node being split are partitioned into child nodes according to their matches to linguistic terms of the wining attribute.

3.3 Policy inference process

From this inductive fuzzy tree, the IA proceeds to infer policies used in the inference process. Each path along the generated tree from root to leaf is converted into a policy with the antecedent part representing the context attributes of each succeeding branch, and the consequent part representing the class at the leaf with highest membership value. The membership value at each leaf is taken to be equal to the maximum membership value of all instances reaching that leaf and having that class as their output.
After generating these fuzzy policy sets from the tree, the inference agent sends them to the CMA in order to be used in the policy enforcement and provisioning of services. The process of inference with new instances and new implicit context information is processed from the generated tree. In this approach, context information is traversed along branches of the tree that satisfies its membership value. After reaching leafs, the class association value for each leaf is calculated. This is accomplished by taking the T-norm between the membership values of classes in the leaf, and the T-norm of that instance with the attributes along the branch from the root to the corresponding leaf.

3.4 Policy Specification and Enforcement

The last step in the automation of context provisioning is representing policies that govern the choices in behavior of the system in machine readable format.
We modeled policy information using OWL language [6] to easily integrate it with our developed ontologies for modeling context and fuzzy inference. The semantic policies in our MAS encapsulate the inferred policies from the fuzzy decision tree with meta-data information and enforcement role. The policy ontology has the following classes and properties:

- Policy (issuedBy*, appliesTo*, appliesWhen*, name*, appliesWhere*, priority*, equivalentTo*, composedOfPolicies*, policyRule*, enforcementType*, requiresOtherPolicies*)
  - Authorization ()
  - ContextSpecificPolicy (category*, description*)
  - Obligation ()
- PolicyEnforcement ()
  - instance Inform
  - instance Negative
  - instance Negotiable
  - instance Positive
  - instance Restricted
- PolicySimilar (degreeOfCertain*, similarIn*, similarTo*)
- Rule (hasAntecedent*, hasConsequent*, hasInferenceType*)

Policies are divided into: authorization and obligation policies [7]. We also have a generic policy class, ContextSpecificPolicy that can be used by application developers to specify its category and gives a description to it. Each policy has properties that state the entity issued the policy, the entities for which it applies the time and location of enforcing it. In addition the developer may state the equivalence of a policy with another one and state the degree of equivalence using ‘Similar’ class defined in the

fuzzy ontology. This concept allows the MAS system to invoke other policies in case of uncertainty and vagueness and thus increases its robustness and performance. To allow the MAS system deduce what action to perform at a policy enforcement, every policy is associated with PolicyEnforcement value that could, for instance, have the value of 'Negative' to prohibit doing the rule in the policy, or it could be 'Negotiable' to allow agents to negotiate how to enforce this policy in case of conflict or inconsistency.

3.5 Semantic Context Modeling

The introduction of ontology, as a paradigm for modeling and representing the context, provides the ability to reason and deduce implicit information and enables the reuse of domain knowledge. As shown in figure 3a, we modeled the context in the form of expressiveness levels, starting from a high level of context abstraction down to levels, where context is expressed and refined in more detail. The highest level of abstraction is the ContextView which represents different types of context that belong to an entity (i.e. a user, a service, a device or the environment itself). ContextView has properties contains and invokes. The classes ContextFeatures and ContextEngagements are the respective ranges of those properties.

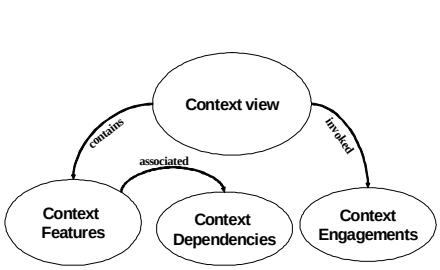


Fig. 3a. The upper context ontology

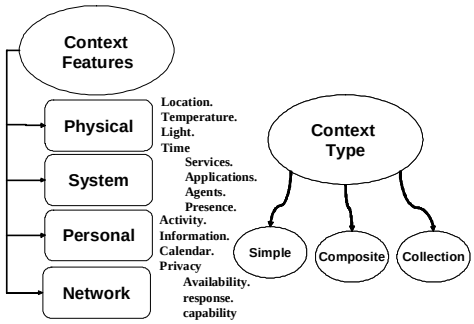


Fig. 3b The defined context features and the associated context types

The ContextFeatures, shown in figure 3b, consists of classes related to the physical environment (i.e. location), classes related to the system (i.e. services), classes related to users (i.e. identities and activities) and classes related to the network (i.e. availability and bandwidth.) Each of these classes is categorized to be either simple, a composite of other context information, or a collection of similar context information [2].

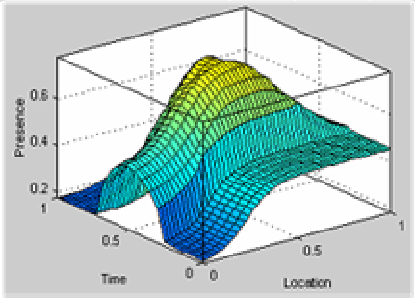
4 Implementation

In order to test the feasibility of the proposed MAS system, we have implemented a context-aware instant messaging system (CIMS). CIMS is an extended version of buddy space open source [8] where we wrapped its interaction mechanism with agent

technology and modified its internal structure to provide contextual information about users and services registered with our MAS system. CIMS provides tailored contextual information such as location, presence, event monitoring, activity alert …etc, based on the generated semantic context information and inferred context policies. ACF and CIF files for each entity are stored in the knowledge base repository. They are used by the inference agent to provide semantic policies and by the CMA to enforce and manage these policies.

Location L{near, within, far}			Time T{prior, nearly, past}			Identity I{uncertain, certain}		Presence_status P{unaware, online, meeting}		
0.8	0.2	0	0	1	0	1	0	0.6	0.3	0
0	1	0	0	1	0	0.4	0.6	0	0.1	0.8
0	0.3	0.7	0	1	0	1	0	0.7	0.2	0
0	1	0	0	1	0	0	1	0	0.05	1
0	0.8	0.2	0.7	0.4	0	0.2	0.8	0.1	0.9	0
..	..	..	..	..	..	..	..	..	..	..

- **Identity**
  - o Certain
    - **Location**
      - Near
        - o **Time**
          - Prior → Online
          - Nearly → Meeting
          - Past → Online
      - Within
        - o **Time**
          - Prior → Online
          - Nearly → Meeting
          - Past → Online
      - Far → Unaware
  - o Uncertain → Unaware




```
<Obligation rdf:ID="Policy1">
 <name>Group_Presence</name>
 <appliesWhere rdf:resource="&Loc;MMARL"/>
  <enforcementType>
   <PolicyEnforcement rdf:ID="Positive"/>
<enforcementType>
 <appliesTo rdf:resource="#G1"/>
 <IssuedBy>
  <Confoaf:SoftwareAgent>
   <Confoaf:name>CMA</Confoaf:name>
  </Confoaf:SoftwareAgent>
 </IssuedBy>
 <policyRule rdf:resource="#Frule1"/>
</Obligation>

<Confoaf:Group rdf:ID="G1">
  <Confoaf:name>MMARL</Confoaf:name>
</Confoaf:Group>□
```

```
<Fuzz:Min rdf:ID="Frule1">
 <Fuzz:hasAntecedent>
  <Fuzz:Antecedent rdf:ID="Ant1">
   <Fuzz:hasFuzzyVariable>
    <Fuzz:FuzzyVariable>
     <Fuzz:hasName>Identity</Fuzz:hasName>
     <Fuzz:hasFuzzyTerm>
      <Fuzz:FuzzyTerm rdf:ID="Certain"/>
     </Fuzz:hasFuzzyTerm>
    </Fuzz:FuzzyVariable>
   </Fuzz:hasFuzzyVariable>
  </Fuzz:Antecedent>
 </Fuzz:hasAntecedent>
...
</Fuzz:Min>

<Fuzz:hasConsequent>
 <Fuzz:Consequent>
  <Fuzz:hasFuzzyVariable>
   <Fuzz:FuzzyVariable rdf:ID="presence_status">
    <Fuzz:hasFuzzyTerm rdf:resource="#online"/>
   </Fuzz:FuzzyVariable>
  <Fuzz:hasFuzzyVariable>
  </Fuzz:Consequent>
 </Fuzz:hasConsequent>□□
```

Fuzzification of the semantic context is provided by the application developer and stored in the knowledge repository and is used to generate the initial training examples that will be used to build the inductive fuzzy tree. The life cycle of context provisioning is depicted in the table above.

The first row represents the context information types used by CIMS to deduce the presence of entities in the environment. The second row shows the membership functions used to fuzzify these three contexts.

We divided the location information of the fifth floor in Colonel-by building at the university of Ottawa with respect to the MMARL laboratory into three categories (near, within and far) while the IA chosen the respective membership functions of these three categories according to the algorithm in section 3-1, the same process applies to the other two contexts as shown in the second row. Rows from 3 till 8 illustrate samples of the training examples used by the inference agent to build the inductive fuzzy tree (IFT). The inference agent builds the IFT shown in the first column of the 9th row while the second column depicts the rule surface extracted from the IFT. Sample of the inferred policies are shown in the last row of the table, where the first column represents a positive obligation policy of the rule shown in the second column of the last row, {*If Identity is certain, Location is near, Time is prior then Presence is unaware}*. The policy is applied to MMARL group, issued by the CMA and applies only to MMARL lab.

Figure 4a shows a snapshot of the CIMS user interface of user Bob. The figure shows the services that can be used by this user, the HP printer and the MMARL projector, according to his location, time and generated policies. The figure also shows that the projector is currently busy, as it has the red light, and also shows that his friend Balo is in a meeting. Hamid is unclear to user Bob which is an indication that the MAS system is un-aware of the presence status of Hamid. It means that Hamid could be present in the environment but his presence is vague.

Figure 4a Snapshot of the CIMS user Interface

Figure 4b The map GUI with entities activities and location projected on it.

Figure 4b is a map that projects the activity and location of entities found in the user's presence list. It is a two layer map where the above layer is the map of the fifth floor while the lower map is the map of Bob's office. Every user is projected in the map so that Bob can have knowledge about their location context spontaneously as they move in the fifth floor. Finally, every user can manipulate his ACF and CIF through the OntoWizard menu item and at the same time browse and search profiles of other entities if their policies authorize such actions.

5 Related Work

While the field of context-aware computing is relatively young, a number of powerful context-aware systems such as Context Toolkit [9] and Hewlett Packard's Cooltown [10] have been developed, but most of them shared a weakness in supporting knowledge sharing and context reasoning because of their lack of ontology [11]. This weakness has been addressed by projects like GAIA [12], PSI [13] and VTT research [14].
GAIA brought the functionality of an operating system to physical spaces. Common operating system functions are supported, such as events, signals, file system, security, processes, process groups, etc. Gaia extends typical operating system concepts to include context, location awareness and mobile computing devices.
PSI described a novel support for attaching context-aware services to physical locations or objects which enables context-aware interaction by utilizing context information from the user, network, and sensors; as well as automated use of available wireless and mobile infrastructure relative to what the application tries to accomplish so as to minimize exposing such decisions to the user.
VTT project proposed a new software framework that simplifies the development of context-aware mobile applications by managing raw context information gained from multiple sources and enabling higher-level context abstractions.
While these projects have successfully built systems that effectively interact with users and the environments' context, they do not incorporate the usage of inductive inference to generate new contextual information from the acquired ones (implicit context) and to automate the process of context provisioning by its inference to semantic policies (implicit usage of context). In addition exchanging inferred context policies and enforcing them using agent technology is a novel technique.

6 Conclusion and Future Work

In this paper, we presented a novel approach for supporting the context-awareness of applications and services in mobile computing. Our implicit context-sensitive technique for inferring new contextual information, a set of semantic policies, and distributing these policies among a number of agents provides various applications and services to be context-aware. As the context changes, new contextual information is generated and semantic context policies are dynamically created to reflect the changes

in the state of the environment. The creation of semantic policies provides spontaneous adaptability to applications and services for handling vagueness in context information. Automated context provisioning through policies also helps developers to rapidly build more reliable context-aware applications and services.
In addition, the separation of system behaviour, in the form of policies, from the system implementation is also a key factor in our MAS system that enables interoperability and system reusability (i.e. encapsulating context in policies eliminates the need for rewriting context management code across applications).
We plan to develop more applications to validate different aspects of the MAS system. This includes inter-domain context provisioning, dynamic tuning to the inductive process of policy generation and context negotiation among multiple domains.

Acknowledgment

This work was partly supported by a Research Grant from the University of Pierre et Marie Curie, ParisVI (France), and the Natural Sciences and Engineering Research Council of Canada.

References

[1] N. Davies, H. Gellersen, “Beyond Prototypes: Challenges in Deploying Ubiquitous Systems” IEEE Pervasive Computing, pp. 26-35, Vol.1, No.1, January-March 2002.
[2] M. Khedr, A. Karmouch, " A Semantic Approach for Negotiating Context Information in Context Aware Systems” IEEE Intelligent Systems Magazine, 2005.
[3] McGuinness,D.L.; “Question answering on the semantic web” Intelligent Systems, IEEE Volume 19, Issue 1, Jan.-Feb. 2004.
[4] M. Khedr, A. Karmouch, “An Infrastructure for Managing Context Information in Pervasive Computing Environments,” PhD Thesis, University of Ottawa, SITE, 2004.
[5] S.R Safavian, et. al, “A survey of decision tree classifier methodology”, IEEE Trasn. Systems Man Cybernet Volume21, Issue 3, May-June 1991.
[6] “OWL Web Language Ontology Reference,” http://www.w3.org/TR/owl-ref.
[7] A. K. Bandara, E. C. Lupu, A. Russo, “Using Event Calculus to Formalise Policy Specification and Analysis”, 4th IEEE Workshop on Policies for Distributed Systems and Networks (Policy 2003), Lake Como, Italy, June 2003.
[8] http://www.buddyspace.org
[9] D. Salber, A. Dey, G. Abowd, “The Context Toolkit: Aiding the Development of Context-enabled Applications”, CHI (1999) 434–441.
[10] T. Kindberg, J. Barton, “A Web-based Nomadic Computing System”, Computer Networks 35(4) (2001) 443–456.

[11] H. Chen, T. Finin, J. Anupam, “Semantic Web in a Pervasive Context-Aware Architecture”, Proceedings of Artificial Intelligence in Mobile System 2003.
[12] M. Román, C. K. Hess, R. Cerqueira, A. Ranganathan, R. H. Campbell, and K. Nahrstedt “Gaia: A Middleware Infrastructure to Enable Active Spaces.”, IEEE Pervasive Computing, Volume 1, Issue 4, pp. 74-83, Oct-Dec 2002
[13] T. Kanter, "Attaching Context-Aware Services to Moving Locations", IEEE Internet Computing, Volume 7, Issue 2, March-April 2003, pp.43 - 51.
[14] Panu Korpipää, el al, “Managing Context Information in Mobile Devices”, IEEE Pervasive, pp. 42-51, Vol.2, No.3, July-September 2003.