Context-Based Cooperation Architecture for Ubiquitous Environment

Minsoo Kim, Youna Jung, Jungtae Lee, and Minkoo Kim

Graduate School of Infromation and Communication, Ajou University,
Suwon, Korea (South)
{visual, serazade, jungtae, minkoo}@ajou.ac.kr

Abstract. Context-awareness which provides relevant information and services to situation using context is an important issue to solve problems in ubiquitous environment. In addition, many problems in ubiquitous environment are solved by interaction between computational elements rather than by performing individual actions. Therefore, it needs to approach context-awareness from in a cooperative point of view. In this paper, we discuss about context model for cooperation system in ubiquitous environment and describe context ontology focused on interaction process with general context. Moreover, we propose cooperation architecture based on context ontology supporting ontological processes. This architecture introduces community as an organization of elements solving common problem and provides context-aware mechanism within community.

1 Introduction

A dynamic and changeable ubiquitous environment gives rise to unpredictable and complex problems. Most of them are solved by interacting between elements rather than performing individual actions. Considering this issue, cooperation is an important issue to satisfy user's requirements and to provide intelligent service in ubiquitous environment. Cooperation system provides primitives for interaction with one another: interaction environments, common goals, interaction protocols, and so on. Elements in cooperation system are strongly dependent on environments and other elements to provide own and/or common services, so essentially, they must recognize the relevant information surrounding them - called *context*. However, because of the complex and dynamic features of ubiquitous environment, it is difficult to recognize or aware context. So to avoid increasing complexity and to ensure context-aware service, system and elements need to be capable to maintain well-defined context model and to provide efficient mechanism for context-awareness.

Context, generally, is referred to 'any information that can be used to characterize the situation of an entity (i.e. whether a person, place or object) that are considered relevant to the interaction between a user and an application, including the user an applications themselves'.[1] Many approaches attempt to do context modeling and context-awareness service in ubiquitous environment, and most of them have focused on the physical entities of the environment(e.g. the person or users, the devices) and little work has been done to develop models to support cognitive entities[2][3].

Information of physical environments, naturally, is important to determine element's activity and service, but cognitive context such as element status, activities and services is also important to build ubiquitous computing system efficiently.

Many cooperation systems are developed for ubiquitous computing environment with supporting context-awareness, and most of them are based on multi-agent system (MAS). Features of MAS such as adaptability, pro-activeness and reactivity, help to build cooperation system. Especially, in many approaches [4][5][6][7][8][9], interactions between agents can be defined using role, introducing the concept of 'role' as a *set of rights* in MAS. This can derive some advantages: separation of issues related to agent algorithmic and interaction with other agents, solution reusability (since roles apply to specific context, so they can be successfully adopted similar areas), and so on. In recent works for role-based system [10][11][12], researchers emphasize the social (organizational) structure and social role to build ubiquitous computing system efficiently. In fact, a role reflects social activities of users or devices, so roles should be comprehensible when they are embedded in social context. Since ubiquitous environment and elements are highly dynamic and changeable, cooperation system for ubiquitous computing environment must support dynamism to operate seamlessly. In other words, system gives proper roles to elements dynamically, and elements can also take the role and can perform a required action.

We think that the following Requirements must be met to build cooperation system for ubiquitous computing environment:

- Context-awareness: providing intelligent service and interaction between elements in system. Context is modeled focusing on cooperative environments with general context (i.e. information of physical environments)
- Role-based interaction: separation of concerns between element's algorithmic logic and interaction logic
- Organizational structure: considering role as required social activity. Organizational structures may provide interaction environment and context-aware mechanism.
- Dynamism: providing methods to solve unpredictable problems in ubiquitous environment, dynamically.

To satisfy upper requirements, we discuss context model for cooperation system in ubiquitous environment and describe context ontology focused on interaction process with general context. We also propose cooperation architecture based on context ontology supporting ontological processes. This architecture introduces community as an organization of elements solving common problem and provides context-aware mechanism within community. This architecture provides mechanism to solve unpredictable problems in a dynamic way which we will discuss later.

The remainders of the paper are structured as follows. Section 2 describes existing role-based system and analyzes role required to ubiquitous computing system. Section 3 introduces some ontological approaches for context modeling, and section 4 specifies our context ontology. Section 5 proposes cooperation architecture and cooperation mechanism supporting system dynamism based on context ontology introduced in section 4. In Section 6, we introduce a new approach 'Community Computing' for developing ubiquitous computing system, and apply context ontology and cooperation mechanism to community computing. At last the conclusions and future works are made in section 7.

2 Cooperation Systems for Ubiquitous Computing

There are many approaches to cooperate between computational elements or agents. The purpose of them is to achieve common goal or to solve common problem shared elements in cooperation system. Most of approaches are based on the concept of ‘role’. Even if role is defined as various ways in different systems or approaches, generally, role is defined as a set of rights. Comparison between some role-based approaches is shown in [7]. It is important to compare ‘roles at runtime’ and ‘openness’ phase for the ubiquitous computing. But we also consider organizational structure supporting role-based cooperation, because supporting ‘roles at runtime’ and ‘openness’ is a social or systematic capability rather than role’s own. In this section, we analyze some role-based approaches and social-concept centered approaches.

2.1 Role-Based Approaches

AALADIN [4] is a meta-model for multi-agent system and focuses on the organization rather than agent in himself. It assumes that the idea of collective structure permits two levels of analysis – concrete level and abstract, methodological level. Concrete level describes the actual agent organization through core concept of model: *agent*, *group* and *role*. Agent is member of one or more group, and group contains roles performing by agent. AALAADIN introduces methodological concept to serve analysis and design MAS. Organization is composed of overlapping groups, and group structure is introduced to partition organization and to specify the group information. Actually, group structure defines roles and interaction protocol between agents performing roles in the group. The organizational structure defines the group structure in the organization and the correspondences between them. In extended work for AALAADIN [5], dynamic aspect of an organization is added: it describes creation of group, entering and leaving of a group by an agent, and role acquisition.

BRAIN [6] is a frame work for Flexible Role-based Interactions in MAS. Role in BRAIN is defined *set of the capabilities* and an *expected behavior* corresponding to the main feature of agent: *pro-activeness* and *reactivity*. BRAIN introduces XML-based XRole notation to describe roles. XRole consists of three main parts to define a role: basic information, allowed action and recognized action. XRole can be translated into appropriate representations, so the different phases of the development of applications relies on the same information, granting continuity during the entire development. At last, RoleSystem in BRAIN is an interaction infrastructure that implements the interaction model of BRAIN. RoleSystem is composed of tow layer: subject layer is platform-independent part of system and wrapper layer is the platform-dependent (e.g. JADE) implementation entity in charge of supporting the subject layer.

Gaia [8][9] introduced a methodology for agent-oriented analysis and design to capture an agent’s flexible, autonomous problem solving behavior, the richness of an agent’s interactions, and the complexity of an agent system’s organizational structures. In Gaia, role is defined a well-defined position in the organization with an associated set of expected behaviors. MAS is regarded as a collection of computational organizations consisting of various interacting roles in Gaia.

2.2 Social Concept Centered Approaches

Cooperation system including MAS makes the design of system less complex reducing the conceptual distance between system and real-world application. In internal view of each agent, agent (or role) solves problem or achieves goal interacting with other agents. In external view, agents are organized to achieve common goal, and organization provides interaction environment to agents and can play a coordinator or negotiator for agents. Organizational or social view of cooperation system can make private agent's behavior to social one, not isolated behavior. And some methodology related to organize agents such as dynamic organization and role binding are efficient way to solve dynamic and changeable problem. In that point, to solve unpredictable problem in ubiquitous computing system, organization or agent society plays an important role.

A number of approaches for building MAS take social concept into consideration. Organizational model for agent society by using contracts [10] takes a collectivist view on agent societies and defines organizational framework consists of three models: organizational model, social model and agent (interaction) model. This approach aims to build MAS and to specify interaction between agents defining contract as a statement of intent that regulates behavior among organizations and individuals. Another approach is based on social role-aware [12]. This approach emphasizes social role-aware as a key feature for ubiquitous computing and proposes multi-agent architecture being aware of social role. The roles are assigned dynamically based on both sensor-data from the environments and role ontology described human designer.

As mentioned above, role means not only agent capability for interacting with others, but also social activity to achieve common goal in the society or organization. In order to take a role as a social activity, cooperation system must support a organizational architecture which can bind a role to proper agent and provide interaction environment for assuring agent activities.

3 Context and Context Modeling

Context is general term adopted in different areas of the computer science and definition of context is also general – any information related a problem implicitly or explicitly. But generality of definition makes difficult to build context model, so smart method must be required to model and use context efficiently. In this section we capture the context in ubiquitous environment and describe method to build context model.

3.1 Scenario in Ubiquitous Environment

The Following famous scenario is staring part of Berners-Lee's article that has introduced the semantic web.[14] It is very short and simple, but we imagine internal process and also capture context required by that process. The scenario has the electronic devices, the agents, the person, and also some actions (i.e. send message, sound down).

The entertainment system was belting out the Beatles' "We Can Work It Out" when the phone rang. When Pete answered, his phone turned the sound down by sending a message to all the other local devices that had a volume control. His sister, Lucy, was on the line from the doctor's office: "Mom needs to see a specialist and then has to have a series of physical therapy sessions. Bi-weekly or something. I'm going to have my agent set up the appointments." Pete immediately agreed to share the chauffeuring.

3.2 Context

To capture and aware context, system requires information to be exchanged and used between different elements, and obviously that information is related element or problem. In scenario, some physical information can be captured: telephone calls, devices location. This information is recognized related elements, and then system can provide intelligent service. Most of the context modeling or context-awareness approaches capture this kind of information as context, including device location, time, temperature and all physical or sensing information. This information is important and very useful for context-aware application in various domains.

But, to complete upper scenario, other information is needed. For Pete's calling with Lucy, sound of silence is required, and device 'phone' must send message to all devices to turn the sound down. In classic approaches, this situation is resolved through interactions or communication between related elements rather than by using context model. In other words, messages from others or interactions in themselves is not information in context model. But, we think that these are also considered as information modeled context. The action 'send message' is enable aware context 'telephone calls' and similar the action 'sound down' is performed by recognizing context 'message from phone'. Moreover, an action in itself is also context: to send message, 'phone' must know whether receiver is capable 'sound down' or not. Finally, Social information as interaction environment is also important: which elements are in the group or community, who is performing a specific role, or what is a common goal that elements achieve by interacting with others.

Thus we regard both information from physical environments and cooperative or social information of system as context. By maintaining context model with them, it is possible to provide an efficient context-awareness and intelligent services.

3.3 Ontological Approach for Context Modeling

To model and aware context in the system, many methodologies is used – key value, mark-up scheme, graphical, object oriented, logic based and ontology based.[13] We choose ontology based approach to context model with following reasons: (i) ontology provides uniform way for specifying the model's core concepts, (ii) ontology enables to have a common set of concepts about context while interacting with one another, (iii) system can exploit reasoning mechanism with explicit semantic based on ontology, (iv) By reusing ontology, system can compose large-scale context ontology.

Some context-aware system for ubiquitous or pervasive computing are proposed with context-model based on ontology. SOUPA [15] ontology has proposed for pervasive computing, and SOUPA vocabularies adopted from a number of different consensus ontologies(time, person, places etc,.). CONON [16][17] ontology provides a vocabulary for representing knowledge about a domain and for describing specific situations in a domain. In a CONON's view, context has two categories: direct context that is acquired from a context provider directly such as sensed context and defined context, indirect context that is acquired by interpreting direct context through aggregation and reasoning process. By introducing classification and dependency, CONON allows the properties of entities to be associated with quality constraints that indicated the quality of context. GAS[18][19] ontology that provides common language for the communication and collaboration among eGadgets defines an architectural vocabulary and configuration rules including eGadget's roles and relations between them.

Our context ontology, will discuss next section, is based on similar idea of these approaches – context model that aims to ubiquitous or pervasive computing - , but we focused on context for cooperation between elements in the ubiquitous computing systems. Classic ontological approaches well-define for information of physical environments, but they have weakness to represents cooperative and social information. GAS ontology aims to communicate and collaborate between elements, but it strongly depends on centered their plug and synapse methodology, that is, it is limited to a little domain and difficult to use general cooperation system.

4 Context Ontology for Ubiquitous System

In this Section, we describe context ontology for cooperation system for ubiquitous computing. We assume that user or system can define domain specific ontology using well-defined ontologies, and system can perform a reasoning or inference process using them together with our context ontology.

Fig. 1 shows context ontology with major concepts and properties. We will write context ontology using OWL (Web ontology Language) recommended for describing web ontology by W3C (World Wide Web Consortium). OWL provides rich expressive power compared other ontology language such as RDF(S), DAML+OIL, and has capability of supporting semantic interoperability to exchange and share context or domain knowledge. Our context ontology is composed of three sub-ontologies. First, Physical Ontology is similar to SOUPA core ontology or CONON upper ontology. In fact, we do not explicit define concepts in physical ontologies, because we think that physical ontologies such as place, time are already well-defined by many researchers. For example, DAML-time ontology is good to conceptualize time and related many services. Moreover, in reusable point of view that is major advantage of ontology, reusing ontologies for information of physical environments is an efficient way to build context ontology.

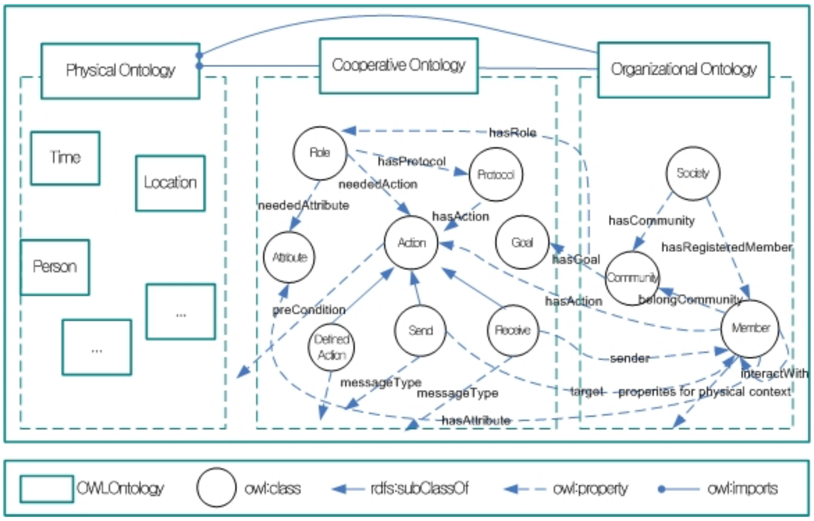

Fig. 1. Context ontology is composed of three sub-ontologies: Physical ontology, Cooperative ontology and Organizational ontology

Second, however, we define explicit the Cooperative Ontology related cooperative part of system. By using this ontology, developers describe interaction process and organizational structure. Cooperative ontology includes the following concepts: the role, the protocol, the action and the other concepts to explain the first three concepts. Role represents interaction between computational elements and social activity to support dynamism of system (see next section to get detail description). Therefore, description of role has protocols describing interaction process, and must include required capabilities (actions and attributes) to perform the role. Protocol that describes sequence of actions is an interaction process between roles. Action represents element's activities with pre-condition, inner-action, activate-time and other properties. Action has special two sub-actions 'Send' and 'Receive' for communication between elements: these actions have properties to enable working on network – 'target', 'sender', 'messageType' (It follows message type proposed by FIPA - Foundation for Intelligent Physical Agent), contents, and so on. 'target' property can have multiple elements. Moreover, actions can have special properties such as 'parallel' in programmatic point of view.

Finally, we define social ontology represented core elements in our cooperation architecture: Society, Community, and Member. Society manages all communities and members, but it is not necessary maintain all information they have. Basically, society represents 'hasCommunity', 'registeredMember' property. Community has 'goal' and 'role' properties and member has 'action', 'attribute' and properties related physical ontology. Member status in cooperation system is necessary context to interact with one another. In cooperation system based on MAS, agent status is represented by using general terms such as 'activate', 'busy', 'idle', but it is not enough to explain an agent to system or other agents. We need more concrete description: for example, agent 'A' can perform 'move' and 'run' actions and is located in a garden, and he is communicating with radio. We emphasize concrete description of member status, because by grasping a member status, other elements in the system can decide and perform a proper action without additional interaction to get his status. In order words, describing status of member and community or other cooperative information can improve the performance of cooperation system. Thus, organizational ontology is designed to satisfy for describing elements of system in concrete way with cooperative ontology.

5 Cooperation System Using Context Ontology

We define organizational structure for cooperation system introducing three core concepts: society, community and member. Each elements works based on context ontologies with some components supporting ontological process and system functionality. Organizational structure and processes to enable cooperation with three core concepts is shown in Fig. 2.

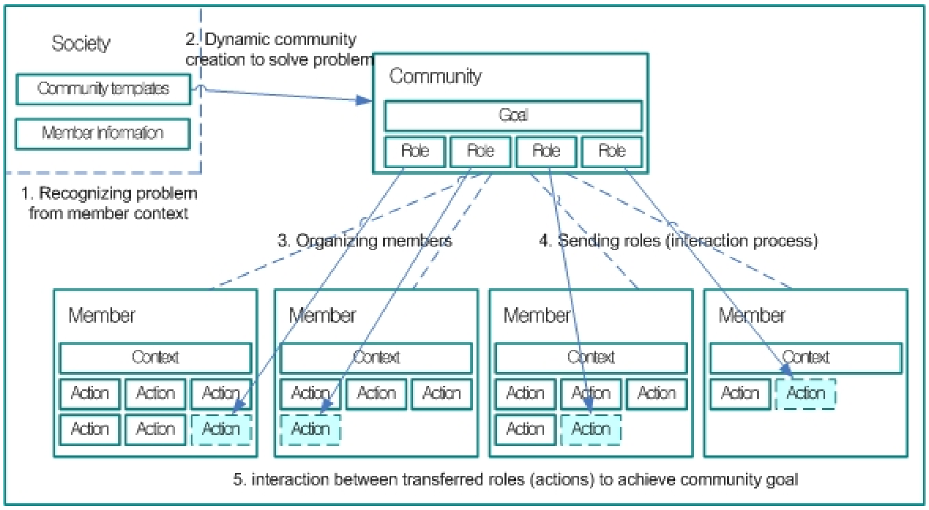
Society
Community templates
Member Information
2. Dynamic community creation to solve problem
Community
Goal
Role
Role
Role
Role
1. Recognizing problem from member context
3. Organizing members
4. Sending roles (interaction process)
Member
Context
Action
5. interaction between transferred roles (actions) to achieve community goal

Fig. 2. Organizational structure and cooperation process

Society is an administrative concept of system and an abstraction of ubiquitous environment. Society must belong all elements and manages them using context ontologies – member ontologies and community ontologies. It is not a space occurring interaction between members to solve problem, but an environment that occurs unpredictable problems, exists members independently, is created community to solve problem.

Community is a cooperative group composed of related members to solve a problem. Problems occurred in ubiquitous environment are not static, but unpredictable and dynamic. Therefore, community does not always exist, but dynamically created by society when problem occurs. Society recognizes the occurrence of a problem using context ontology, then creates community which can solve that problem according to community ontology. Community is created with following information: (i) goal that will be achieved, and (ii) roles implying participants in community, and (iii) interaction process to achieve community goal. With this information, community can dynamically cast members and also dynamically transfer interaction processes to each cast members. Transferred interaction process is an instance of 'Protocol' class in context ontology described previous section. When goal is achieved, community is finalized dynamically, and members belonged to community leave the community and perform their own actions.

Member is a basic element in the system, representing person, device, hardware or software, similar to agent in MAS. Member provides services (actually member actions and described using 'Action' class) when satisfying 'precondition' property in 'Action' class. In Fig. 2, transferred interaction process is added as member action. We consider both alone-action and interaction between members as same kind of thing, since all actions are performed when satisfying precondition of themselves by recognizing context. Fig. 3 shows an internal architecture of member that enable this process based on context-awareness.

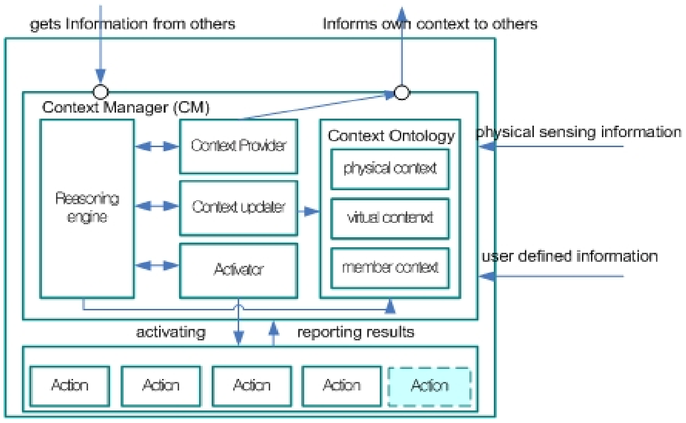

Fig. 3. Internal architecture for member

Context Manager (CM) contains a number of components including context ontology. CM gets information from other members (message), physical sensors, users and results for own actions, and then context updater updates context ontology with reasoning engine. Note that messages from other members are also information for context, not just message in traditional cooperation system. The reasoning engine provides reasoning process to other components in CM: checking pre-condition of an action, interpreting ontology, and other useful reasoning. Context provider informs own context to other elements including society and community in system. Note that 'send' action for interactions between members is performed by context provider ('receive' action is performed by CM). In other words, 'Send' and 'Receive' are not special action; they just decide target/send and type of message, and so on. Actuator activates actions that satisfy precondition by reasoning context ontology with reasoning engine.

Developers describe cooperative and organizational ontology as a program or set of rules, then system interprets this ontology and performs proper actions. For example, member ‘m1’ performs action ‘a1’ when receiving message ‘msg1’ from member ‘m2’, and after performing ‘a1’, send message ‘msg2’ to ‘m3’, ‘m4’, and ‘m5’. When receiving message ‘msg1’, first, context updater updates context ontology with ‘msg1’, then CM checks preconditions of actions through reasoning engine. If ‘msg1’ come from ‘m2’, that is, satisfying precondition of ‘a1’, actuator activates ‘a1’. When ‘a1’ performed, ‘a1’ reports result to CM, and CM updates context ontology again. Finally, context provider sends context ‘msg2’ to ‘m3’, ‘m4’ and ‘m5’.

Society and community have similar to member architecture shown in Fig. 3. In order words, all elements in system work based on context management and context awareness, including to interaction with one another through CM, not individual action. Implementation of actions is portion of developer in difference platform, so we do not consider specific platform or implementation method in real-world applications.

6 Related Work - Community Computing

Community Computing [20] is the community-based service development methodology, proposed to develop ubiquitous spaces. Community computing focuses on how ubiquitous spaces satisfy their requirements with cooperation between predefined entities while the existing distributed object approaches focus on what entities are needed to satisfy the requirements. A community computing system means that a service providing system developed using the community computing model. A ubiquitous space can be developed as a community computing system, and the requirements of a ubiquitous space are fulfilled by communities.

To development a community computing system, community computing takes the development process with the MDA (Model Driven Architecture) approach. The MDA is an approach to system development, which increases the power of models in that work. It proposes to start the process by building high-level abstraction models obtained by requirement analysis, and then refine them until obtain models that directly represents the final system. For applying MDA approach to the development process of the community computing systems, three different abstraction model is proposed: Community Computing Model (CCM) as the most high-level abstraction model, Platform Independent Community Computing Implementation Model (CIM-PI) as implementation model without platform specific features, and Platform Specific Community Computing Implementation Model (CIM-PS) as complete implementation model as real-world application.

Social concept of Community computing is similar to describing in this paper. But, not yet, context modeling and context-awareness do not provided in community computing model. Since context-awareness is essential element to realize ubiquitous computing system, community computing must meet the context. We believe that community computing can more powerful model to develop ubiquitous computing system with context-awareness using context ontology proposed in this paper.

7 Conclusion

In this paper, we describe context ontology for cooperation system in ubiquitous computing environment, and propose organizational structure and cooperation processes based on context ontology. Context ontology represents not only information of physical environments, but also cooperative or organizational information required when cooperating between members. By using this ontology, cooperation system can provide context model and context-awareness improving the performance of system. Based on our context ontology, we also present Organizational architecture composed of society, community and members. This architecture supports a dynamic way to solve unpredictable problem in ubiquitous environment. However, there are several issues that need to be considered in future works.

- Completion of context ontology and writing in OWL: current context ontology has some weakness for organizing physical ontologies with cooperative and social ontologies. And to develop real-world application with context ontology for sharing and reusing context, it is good to publish context ontology using recommended ontology language OWL.
- Explicit description of ontology reasoning: main component of proposed architecture is the reasoning engine supporting various ontological processes. As yet, we describe reasoning process implicitly, but for completion of context management and cooperation system, we need to explicit reasoning mechanism with well-performed reasoning engine.
- Practical uses with Community Computing: community computing introduced in previous section is good to be exploited context ontology. With context ontology, community computing will have ability to manage context, and then will solve context-related problems such as resolving conflict.

Acknowledgments. This research is supported by the ubiquitous Computing and Network (UCN) Project, the Ministry of Information and Communication (MIC) 21st Century Frontier R&D Program in Korea.

References

1. Anind K. Dey, "Understanding and Using Context", Personal and Ubiquitous Computing, Special Issue on Situated Interaction and Ubiquitous Computing, vol. 5(1), 2001
2. Paul Prekop and Mark Burnett, "Activities, Context and Ubiquitous Computing", Computer Communications, vol. 26, p.p.1168-1176, 2003
3. A. Schmidt, M. Beigl, and H.W. Gellersen, "There is more to Context than Location", Computers and Graphics, vol. 23, p.p.893-901, 1999
4. J. Ferber and O. Gutknecht, "A meta-model for the analysis and design of organization in multi-agent systems", In Proceedings of 3rd International Conference on Multi-agent Systems (ICMAS'98), 1998
5. J. Ferber, O. Gutknecht, F. Michel, "From Agents to Organizations : An Organizational View of Multi-agent Systems", In Proceedings of AOSE 2003, Australia, 2003

6. G. Cabri, L. Leonardi, F. Zambonelli, “A Framework for Flexible Role-based Interactions in Multi-agent System”, In Proceedings of the 2003 Conference on Cooperative Information Systems (CoopIS), Italy, 2003
7. G. Cabri, L. Ferrari, L. Leonardi. “Agent Role-based Collaboration and Coordination: a Survey About Existing Approaches”, In Proceedings of the 2004 IEEE systems, Man and Cybernetics Conference, Netherlands, 2004
8. M. Wooldridge, Nicholas R. J. “The Gaia Methodology for Agent-oriented Analysis and Design”, Autonomous Agents and Multi-Agent Systems, 3, p.p. 285-312, 2000
9. R. Jennings, et. al. “Developing Multiagent Systems: The Gaia Methodology”, ACM Transactions on Software Engineering and Methodology, 12, 3, p.p. 317-370, 2003
10. Xinjun Mao and Eric Yu, “Organizational and Social Concepts in Agent Oriented Software Engineering”, 5th International Workshop, AOSE 2004, New York, USA, July 19, 2004.
11. V. Dignum, J-J. Meyer, H. Weigand, and F. Dignum, “An organizational-oriented model for agent societies” In Proceedings of International Workshop on Regulated Agent-Based Social Systems: Theories and Applications (RASTA'02) at AAMAS'02, 2002.
12. Akio Sashima, Noriaki Izumi, Koichi Kurumatani, Yoshiyuki Kotani, “Towards Social Role-Aware Agents in Ubiquitous Computing”, Proceedings of ubicomp2005, Japan, 2005
13. T. Strang and C. Linnhoff-popien, “A Context Modeling Survey”, In Proceedings of 1st International workshop on Advanced Context modeling, Reasoning and Management UbiComp2004, 2004
14. T. Berners-Lee, J. Hendler and O. Lassila, “The Semantic Web”, Scientific American, p.p.35-43, 2001
15. Harry Chen, Tim Finin, and Anupam Joshi, "The SOUPA Ontology for Pervasive Computing", Ontologies for Agents: Theory and Experiences, Springer, July 2005
16. Xiao Hang Wang, Tao Gu, Da Qing Zhang, Hung Keng Pung, "Ontology Based Context Modeling and Reasoning using OWL", In Proceedings of Workshop on Context Modeling and Reasoning(CoMoRea 2004), In conjunction with the Second IEEE International Conference on Pervasive Computing and Communications (PerCom 2004), Orlando, Florida USA, March 2004
17. T. Gu, X. H. Wang, H. K. Pung, D. Q. Zhang. “An Ontology-based Context Model in Intelligent Environments”, In Proceedings of Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2004), pp. 270-275. San Diego, California, USA, January 2004.
18. Christopoulou E., Kameas A., "GAS Ontology: an ontology for collaboration among ubiquitous computing devices", Protégé special issue of the International Journal of Human – Computer Studies, 2004
19. Eleni Christopoulou, Christos Goumopoulos, Achilles Kameas, "Context-aware systems: An ontology-based context management and reasoning process for UbiComp applications", In Proceedings of the 2005 joint conference on Smart objects and ambient intelligence: innovative context-aware services: usages and technologies sOc-EUSAI '05, 2005
20. Youna Jung, Jungtae Lee, Minkoo Kim. “Multi-agent based Community Computing System Development with the Model Driven Architecture”, In Proceedings of 5th International Conference on Autonomous Agents and Multiagent Systems (AAMAS), Japan, 2006.