Community Computing Model Supporting Community Situation Based Strict Cooperation and Conflict Resolution*

Youna Jung, Jungtae Lee, and Minkoo Kim

Graduate School of Information and Communication Engineering, Ajou University, Suwon, Republic of Korea, 443-749
{serazade,jungtae,minkoo}@ajou.ac.kr

Abstract. Community computing is a new computing environment where ubiquitous services are provided by cooperation between existing smart object. In these days, it is studied by many researchers but works on community computing are still at an early phase. To design and describe cooperation effectively, in this paper, we propose the community situation based cooperation model. In addition, we introduce conflict resolution scheme for community computing. Consequently, we propose the community computing model supporting the community situation based cooperation and conflicts resolution. Case studies are also tried to examine the proposed community computing model.

1 Introduction

In recent years, 'Community Computing' has been suggested as a new technical environment. For an instance, Jonathan Murray, Microsoft's chief technology officer for the EMEA (Europe, Middle East and Africa) region, expressed his vision about community computing in InfoWorld magazine [1]. He said, "We are moving from a world where we just have my own personal device that runs my own applications to a new world where we are sharing other device's computing capacity and resources". Microsoft Company called such a new environment as community computing. In fact, the idea is not totally new. By several projects such as PICO [2] and GAIA [3], community concept had been introduced. Yet despite these interests, a number of fundamental questions about community computing remained unanswered. In particular, a formal model and development process for community computing system were not well defined. In order to find an answer, we have researched on the model for community computing and a development process to generate a community computing application system. As a progress, we proposed an early version of community computing model and a development process using MDA approach [4] .

However, in the previous model, there is no cooperation model. At that time, we didn't have an idea to abstract cooperation, thus we just described a specific

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

procedure of cooperation just like pseudo codes in protocol description part. Such description style didn't help to intuitively design cooperation of community at all. By the difficulties in cooperation design, we have tried to find a cooperation model for community computing. In addition, in the previous model, we assumed a conflict-free circumstance but, in practice, conflicts are exists. Accordingly, we have also tried to find a solution of conflicts.

In this paper, the major contribution is that we proposed the community situation based cooperation model for community computing. By the cooperation model, each member of a community cooperates with other members according to a community situation. Additionally, we analyzed conflicts which can be happened in community computing systems, and make up policies for conflict resolution. Using the cooperation model and policies, we improve the previous community computing model.

The rest of the paper is organized as follows. In section 2, we introduce some related works, and then we propose the community situation based cooperation model and conflicts resolution scheme in section 3. In Section 4, community computing models supporting the community situation based cooperation model and conflicts resolution scheme are proposed. Case studies are presented in section 5. Finally, section 6 is dedicated to the conclusion and future works.

2 Related Works

2.1 Previous Community Computing Model

In our work, community computing is a computing technology to offer ubiquitous services by exploiting the cooperation between smart objects. To design and develop a community computing system effectively, we surveyed several existing models. In particular, we concentrated on abstraction models for multi-agent systems such as GAIA [3] because of agent's flexible and autonomous problem solving behavior. However, the existing multi-agent based models focus on what agents are needed to satisfy the requirements of a system, while community computing focuses on how to meet requirements of a ubiquitous system using cooperation between given ubiquitous objects. In other middleware approach projects such as Active Space [5] and PICO [2], a vision is similar with ours but there are no formal models. Therefore, we proposed a community computing model [4] as an abstraction model for community computing systems. In the previous community computing model, a community computing system is abstracted as a society, and a society is composed of members and communities. A community, a proactive organization consisting of members, is represented by its goals, protocols, and necessary roles. A member is a ubiquitous object, which can play a certain role in a community. At the runtime, if a goal arises dynamically, ubiquitous objects are selected for each role then a community is instantiated. After creation of a community, each member cooperates with other members to attain a goal according to protocol description. When the goal is finally achieved, the community is disorganized.

2.2 Existing Cooperation Models

In the previous community computing model, cooperation between members is considered as a predefined procedure. It means that a designer should know which task should be executed in which order. However, in case of a huge and complex cooperation, it is not easy that a designer lay out a whole cooperation procedure in once. Therefore, we considered a cooperation model should be necessary to intuitively design the cooperation. To find an appropriate cooperation model, we surveyed existing cooperation models. In many systems, infrastructures, and cooperation models, cooperation is used and described.

First of all, in 1997, a refined formal cooperation model for ARCHON was proposed [6]. In this model, cooperation is represented just as a recipe, a set of predefined tasks. AGDRSCOM [7] is an agent cooperation model which member agents are able to adjust own cooperative tasks according to the changes of environment and the feedbacks from other members. In AGDRSCOM, an idea of adaptive cooperation is introduced but the detailed mean of adaption is not proposed. Cooperation is just represented as a programming element in a skill description. In the cooperation model of MAPFS [8], a cooperation process is also procedural described by actions and instructions. In 2006, Ji Gao is proposed the hybrid cooperation using recipes, policies, and advertisements [9]. In this model, policy is the obligations and restrictions that agents should comply to, and advertisement is the record of interests of other agents. However, the fundamental cooperation process is also represented by a recipe.

In most cooperation models, as you can see, cooperation is described as a predefined static pseudo program called as recipe, plan, or skill. In many systems, infrastructures, and cooperation models, means of realizing cooperation were introduced but the mean of designing cooperation itself was not concerned. Therefore, we arrived at a decision that we needed a new cooperation model to design cooperation of community intuitively.

3 Community Situation Based Cooperation Model

In order to design cooperation of community, we proposed the community situation based cooperation model as a new cooperation model, especially for community computing [10]. The idea is that cooperation is executed according to community situation. If community's situation is changed then tasks of each member are also changed. That is, tasks which each member should perform are decided by the community situations. At this time, the final situation of a community should be a goal achievement situation. Since the proposed cooperation model is based on the community situations, we define the community situation first.

3.1 Community Situation Model

In order to define the community situations, we proposed the community situation model. In this model, a community situation is determined by situations of specific members. At this time, a member situation is decided by attribute values of the member. The definition of a community situation is as follows.

In this version of community situation model, a community situation is represented as a logical association of attributes. However, the expression power of the community situation model can be improved. If the power of community situation model then the cooperation model is also improved.

Definition 1. Community Situation Model

```
<community-situation>::= <situation-name> = <community-situation>
<community-situation>:: = <single-community-situation> | <conjunctive-community-situation> |
      <disjunctive-community-situation>
<single-community-situation>::= [<quantifier>] <role-name>.<role-situation>
<quantifier>::= ∀ | ∃ , <role-name>::=<string>, <role-situation>::=<member-situation>
<conjunctive-community-situation>::=(<community-situation>AND<community-situation>)
<disjunctive-community-situation>::=(<community-situation>OR<community-situation>)
<member-situation>::= <single-member-situation> | <conjunctive-member-situation> |
     <disjunctive-member-situation>
<single-member-situation>::=<attribute>,<attribute>::=(<attribute-name> <operator><attribute-value>)
<attribute-name>::= <string> , <attribute-value>::= <value>
<operator>::= >| < | >= | <= | = |!= ,<value>::=<number>|<string>|<symbol>|TURE|FALSE
<conjunctive-member-situation>::=(<member-situation>AND<member-situation>)
<disjunctive-member-situation>::= (<member-situation> OR <member-situation>)
```

3.2 Community Situation Based Strict Cooperation Model

Using the proposed community situation model, we define the cooperation for community computing. Before the definition the community situation based cooperation model, let you know some promises of this model.

Assumptions. The community situation based cooperation model is founded on following strong promises.

1) Certainty of community situation
2) All members of a community are aware of community situations and know own tasks to do according to each community situation
3) In a community situation, each member can perform more tasks than one in sequential order
4) Although tasks of members are not completely finished in a community situation, community situation can be changed
5) Community situation is dynamically changed, but finally reached the situation of goal achievement

Definition 2. Community Situation based Cooperation Model

```
<community-cooperation>::= <goal-name> = {<cooperation-block>}+
<cooperation- block >::=<community-situation>=>{<role-task>}+,<role-name>::=<string>
<role-task>::=<role-name>:{<role-action-name>}+,<role-action-name>::= <string>
```

When a member performs own actions in a certain community situation, conflicts with other actions of the member or another member can occur. Such conflicting actions may be executed to play another role for a different community or be not finished in a previous community situation. In both case, we should resolve conflicts.

First of all, we defined that tasks of a member in a certain community situation are executed by one thread, thus we do not need to worry about conflicts on a thread. Accordingly, what we should consider is conflicts between threads. These conflicts are happened when a member cannot execute actions or when more than two members try to execute conflicting actions simultaneously. To handle such conflicts, we classify conflicting actions into two types, mutual exclusive conflict type and time dependent conflict type. In case of the mutual exclusive conflict type, if a conflict occurs then one among conflicting actions should be terminated. In case of the time dependent conflict type, one among conflicting actions should be executed first and then another action is executed. For handling conflicts in runtime, a community manager has an action-conflicts list about conflicts between own actions of a member or actions of different members. The list represents types of action conflicts. At this time, conflicts between same actions can be included in the list. For example, assume that a member performs action a_2 in community situation S_1. Then a situation is changed to S_2, although a_2 is not finished. After that, a situation is changed again to S_3 and the member should perform a_2 in a situation S_3. However, a_2 executed in previous situationS_1 is still operating.

Definition 3. Action-conflicts List

```
<action-conflict-list-in-community>::= {<mutual-exclusive-action-conflicts-in-community> |
    <time-dependent-action-conflicts-in-community>}*
<mutual-exclusive-action-conflicts-in-community-in-community>::= MEC(<role-name>.
    <remained-action-name>,<role-name>.<killed-action-name>)
<time-dependent-action-conflicts-in-community-in-community> ::=  TDC(<role-name>.
    <precedent-action-name>,<role-name>.<following-action-name>)
<remained-action-name>::= <action-name> , <killed-action-name>::= <action-name>
<precedent-action-name>::=<action-name>,<following-action-name>::=<action-name>
    <action-name>::= <string>
```

4 Community Computing Models with Community Situation Based Strict Cooperation

In order to design the community computing system, we had proposed the community computing model called as CCM. In addition, to systematically develop community computing systems, we had also proposed a development process [4]. Since the process is based on MDA (Model Driven Architecture) approach, we had generated more detailed models than CCM, CIM-PI (Platform Independent Community Computing Implementation Model) and CIM-PS (Platform Specific Community Computing Implementation Model). Using the proposed models and development process, we could create a community computing system fast and conveniently.

However, as we mentioned above, the previous models did not concern about cooperation model [4]. In those models, cooperation was just described like a procedural pseudo codes. Although the previous models aimed to abstract a cooperation system, an idea of cooperation was not involved. In order to make up the defect, we generated the community situation based cooperation model [10]. Thus, in this paper, we applied the cooperation model to the previous community computing models. As a result of that, we propose the improved community computing models, the community situation based community computing models. In this section, we introduce the community situation based CCM, the community situation based CIM-PI, and the community situation based CIM-PS. The differences from the previous models are as follows.

- Community situation based cooperation between members
- Conflict Resolution in a community computing system

4.1 The Community Situation Based Community Computing Model

The community computing model, called as CCM, is the most high-level abstraction model for a community computing system. The objective of the CCM is to describe the requirements and the boundary of a system. In order to do, a community computing system is represented as a society, and a society consists of community types and member types. The major difference between the previous CCM and the community situation based CCM is in the cooperation description part. In the improved CCM, a cooperation of a community is represented by community situations and description about each role's tasks in a certain situation. In Fig.1, an example of the community situation based CCM is shown.

```
Society COEX_Mall {
 Community Type Description {
  Community Patrol_COEX{
   Role Patrol_Robot : 1 ~ 10 {
     Attribute:POWER={ON|OFF};
     LOCATION={location_type};MODE={BUZY|ORDINARY};
     Action:Area_Assign();Patrol();
     Cast : POWER=ON; LOCATION= IN.COEX_Mall;}
   Role Patrol_Manager : 1~ 2{
    Attribute:STATUS={ON DUTY| OFF DUTY};
      LOCATION={location_type};
    Action:Patrol_Management();
    Cast : STATUS=ON DUTY; LOCATION=IN.COEX_Mall;}
   Role Guide : 1~ 5{ ... }
   Goals Patrol_COEX(Patrol_Robot, Patrol_Manager, Guide){
     PATROL_AREA_ASSIGN: Areas of All patrol_ robots,
       guides, and patrol_manager is assinged
     PATROL_BEGIN: Start up a patrol service at COEX
    PATROL_END: Shut up a patrol service at COEX, and
     disorganize an instance of 'Patrol_COEX' community}
   Ontology : Patrol_COEX_Ontology; }
  Community Find_Person{
   Role Patrol_Robot: 1 ~ 10 {
    Attribute:POWER={ON|OFF};
LOCATION={locaton_type};
     Action: Area_Assign();Patrol();Broadcast_Info();
       Find_Person();Guide_To();
      Cast : POWER=ON; LOCATION= IN.COEX_Mall; }
    Role Guidian_of_Lost_Person: 1 {
     Attribute:LOCATION={locaton_type};
CONTACT={OMD_ID};
     Action:
    Cast:LOCATION= IN.COEX_Mall; }
    Role Guide : 1~ 5 {
      Attribute : STATUS={ON DUTY| OFF DUTY};
        LOCATION={location_type};
     Action:Patrol();Find_Person();Guide_To();Report_Police();
      Cast : STATUS=ON DUTY; LOCATION=IN.COEX_Mall;}
    Role Salesman: 1~1000{ ...}
    Goals Find_a_lost_person(Patrol_Robot,
      Guidian_of_Lost_Person, Guide, Salesman){
     READ_PROFILE_OF_PERSON:Read a profile of a lost per-
     son
     FIND_PERSON: Broadcast the profile, and try to find a
         person
     PERSON_FOUNDED : The person is founded
     PERSON_NOT_FOUNDED : The person is founded}
    Ontology : Patrol_COEX_Ontology; } }
 Member Type Description {
  Member Siociety_Member{
    Attribute : LOCATION={locaton_type};
    Cast : LOCATION= IN.COEX_Mall; } }
```

Fig. 1. An example of community situation based CCM

4.2 The Community Situation Based Platform Independent Community Implementation Model Supporting Conflict Resolution

CIM-PI is a more detailed model than CCM. Its objective is to describe the implementation using given ubiquitous objects without knowledge of specific platforms. In order to do, in CIM-PI, descriptions of society, community types, and member types are more expanded and detailed. First of all, in the community type description, mapping information between role and member types is added to represent which member types can play which role. Secondly, description of cooperation is detailed. In particular, tasks to be executed by a member shaped up as a sequence of actions of the member, and the definition of community situations is also specified. In third, conditions of community creation are described. To initiate a community instance, two ways are allowed: a member requests an initiation to a society manager or a community manager requests an initiation as a part of cooperation. Finally, policies are added to manage conflicts during the lifetime of a community. In the present version, member casting policy, member secession policy, and action conflicts list (see Definition.3) are defined. The member casting policy represents a rule about member selection such as distant dependent casting or response-time dependent casting. In the member secession policy, treatments for sudden secession of a member are specified. For examples, if a member disappears, then we can initialize a cooperation process with a new member, continue cooperation with a new, or terminate the cooperation. In the member type description part, all member types are described and hierarchy of member types is also defined using the *extends* keyword. In addition, member situations are specified as a logical association of attribute's values. Finally, policies for a member are also described. When a member performs tasks to play one or more than one role, conflicts between own tasks can occur. To resolve such conflicts, we define an action conflicts list (see Definition.3) and represent it in member policy description. In society description, society policy is additionally described. In society policy description, precedence of communities and exclusive communities are defined. When a society manager takes more than one requests for community creation, these policies are used to select one. In Fig.2, an example of the community situation based CIM-PI is shown.

```
Society COEX_Mall {
Community Type Description {
 Community Patrol_COEX{.....................}
 Community Find_Person{
  Role Patrol_Robot: 1 ~ 10 {   ............}
 Role-MemberType Mapping {
  Patrol_Robot:ARGUS; Guidian_of_Lost_Person:
  Human; Guide:Guide;Salesman:Human; }
 Goals Find_a_lost_person(Patrol_Robot,Guidian_of_Lost_
        Person, Guide, Resident)
  {FIND_PERSON_REQUEST=>
    Patrol_Robot : Read_Personal_Profile(); Broadcast_Info
     (∀Patrol_Robot  and ∀Guide and ∀Resident, “Find a
     person”, profile);;
   FIND_PERSON=>
     Patrol_Robot : Find_Person(profile);
     Guide : Find_Person(profile);
     Salesman : Find_Person(profile);
   PERSON_FOUNDED=>
     Patrol_Robot and Guide and Salesman :
        Announce(∀Patrol_Robot  and ∀Guide and
        ∀Resident, “Person is founded”, location);
         Guide_To(founded person, information office);;
   PERSON_NOT_FOUNDED=>
      Patrol_Robot and Guide and Resident :
      Announce(“Person isn't founded”, ∀Patrol_Robot);;
     Guide: Report_Police(“ lost person”, profile);;}
 Community Situation {
   FIND_PERSON_REQUEST={
       Patrol_Robot.TAKE_REQUEST_FIND_PERSON};
   FIND_PERSON={Patrol_Robot.FIND_PERSON};
   PERSON_FOUNDED={∃(Patrol_Robot and Guide
      and Resident.PERSON_FOUNDED};
   PERSON_NOT_FOUNDED={NOT ∃Patrol_Robot.
      PERSON_FOUNDED=YES) };}
 Community Creation {
  ByMember: {ARGUS.TAKE_REQUEST_FIND_PERSON;}
   By Community: }
 Community Policy {
   Member Casting Policy {
      Patrol_Robot: distant-dependent; Salesman:  distant-
      dependent; Guide:  distance-dependent;  }
   Sudden Secession of Member {
      Patrol _Robot: Continue with a new; Salesman: Continue
      with a new; Guidian_of_Lost_Person: Initialize with a new;
      Guide:  Continue with a new; }
   Action Conflicts List={ MEC(Report_Police(“ lost person”,
      profile),Find_Person(profile)); }}
Ontology : Patrol_COEX_Ontology;  } }
Member Type Description {
 Member COEX_MallTIZEN {
  Attribute : LOCATION=IN.COEX_Mall; }
 Member Animate Object extends COEX_MallTIZEN {  .........}
 Member ARGUS extends Robot {
   Attribute : MODEL=STRING; FIND_PERSON={YES|NO};
    TAKE_REQUEST_FIND_PERSON={YES|NO};
    PERSON_FOUNDED={YES|NO};
   Actions Area_Assign(COEX_Mall, Patrol_Robot);
    Patrol(COEX_Mall); END_Patrol();
    Read_Personal_Profile(); Broadcast_Info(
    ∀Patrol_Robot  and ∀Guide and ∀Resident,
    “Find a person”, profile); Find_Person(profile);
    Announce(∀Patrol_Robot  and ∀Guide and
    ∀Resident, “Person is founded”, location);
    Guide_To(founded person, information office);
    Announce(“Person is not founded”, ∀Patrol_Robot);
   Member Situation {
    TAKE_REQUEST_FIND_PERSON:
       TAKE_REQUEST_FIND_PERSON=YES;
    FIND_PERSON:FIND_PERSON:
       FIND_PERSON:FIND_PERSON=YES;
    PERSON_FOUNDED:PERSON_FOUNDED:
       PERSON_FOUNDED:PERSON_FOUNDED=YES;}
   Member Policy {
    Exclusive Actions={ MEC(Patrol(COEX_Mall), END_Patrol_
    Service()); } } }    ...  }
Society Policy {
  Community Precedence {
   High_Priority: Find_Person
   Medium_Priority: Patrol_COEX, Sell_Product
   Low_Priority:}
 Exclusive Community = { } }  }
```

Fig. 2. An example of community situation based CIM-PI

4.3 The Community Situation Based Platform Specific Community Implementation Model Supporting Conflict Resolution

In a community situation based CIM-PS, combines the description in the CIM-PI with the details that specify how that system uses a particular platform. In improved CIM-PS, descriptions about attribute acquisition, action mapping, and member configuration are added. In attribute acquisition part, we describe where values of each attribute derived from. The source of attribute values can be a kind of sensor or action. In action mapping description, we describe how to realize actions of members. In case of using existing programmed objects, we should make a connection between actions in model and programmed actions in an existing object. On the other hand, in case that we should program a ubiquitous member object, we use action names in models to program a member. In member configuration part, components of each member are described.

```
Society COEX_Mall {
Community Type Description { ... }
Member Type Description { ...
 Member ARGUS extends Robot {
   Attribute : ...
   Actions  ...
   Member Situation {.................}
   Member Configuration={
    Vision_Sonsor_v3; Samsung_Location_Sensor_v1;}
   Attribute Acquisition {
    TAKE_REQUEST_FIND_PERSON:Vision_Sonsor_v3;}
   Action Mapping {
    Area_Assign(COEX_Mall, Patrol_Robot):Set_patrol_
    range(location); Patrol(COEX_Mall):CyberCap(patrol);
   END_Patrol_Service():CyberCap(patrolstop);
      Read_Personal_Profile():Read_RFID(person_RFID);
       Broadcast_Info(∀Patrol_Robot  and ∀Guide and
      ∀Resident, “Find a person”, profile): BroadCast
      (towhom, msg); Find_Person(profile):Search_Obj(Info);
      Announce(∀Patrol_Robot  and ∀Guide and
      ∀Resident, “Person is founded”, location):Notify
      (towhom,msg);Guide_To(founded person, information
      office):GuideServie(who,where);Announce(“Person is
      not founded”,  ∀Patrol_Robot):Notify(msg,towhom);}
    Member Policy {
     Exclusive Actions={
       (Patrol(COEX_Mall), END_Patrol_Service()); } } }
   ...}
Society Policy { ...  }
```

Fig. 3. An example of community situation based CIM-PS

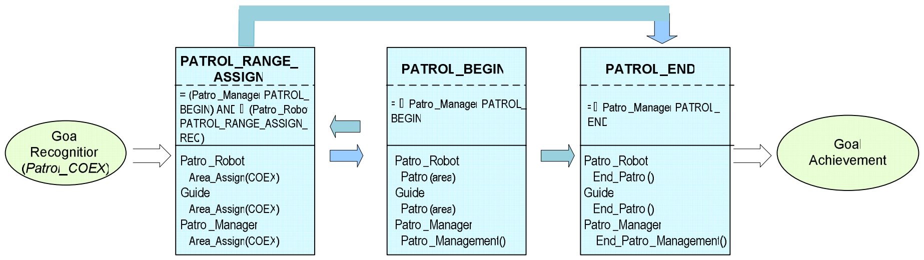
5 Case Study

In a huge shopping mall, several robots exist. These robots have various functionalities such as move, vision sensing, alarm, voice recognition, information search, and so on. At the ordinary time, each robot offers its own services such as guide service or information presentation. Sometime, robots compose a community to achieve a community's goal. Each robot can take multiple roles, depending on its ability.

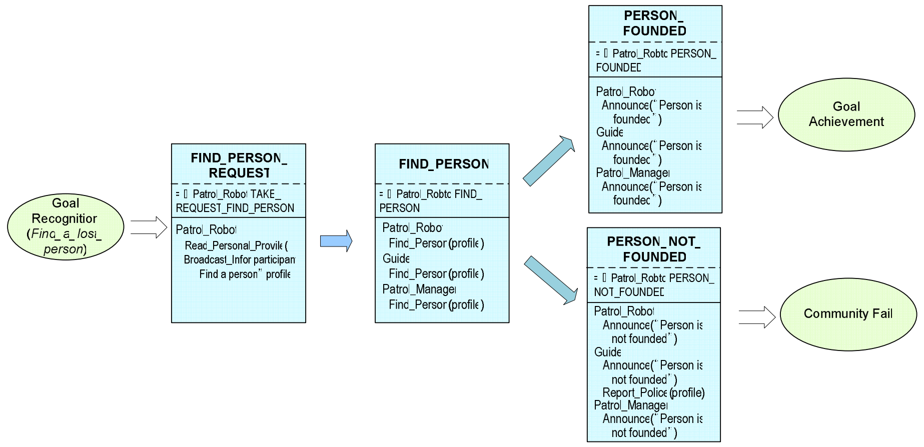
Level-1 Cooperation: When a shopping mall is opened, a *Patrol_COEX* community is initiated by casing all robots and guides. Area to patrol is assigned to all robots and guides as soon as a community is created, then each robot and guide patrols assigned area. When a robot or guide cannot patrol because of too much load or sudden interruptions, they request to reassign.

Level-2 Cooperation: When a robot is on patrol as a member of a *Patrol_COEX* community, the robot is asked for finding a lost child by child's mother. The robot generates *TAKE_REQUEST_FIND_PERSON* member situation, and then requests a creation of *Find_Person* community to a society manager. The society manager, which supervises the COEX-Mall Community computing system, creates a community manager for *Find_Person* community, and then the community manger initiates a

Find_Person community by casting necessary members. The robot taking a request sends child's profile to all robots, guides, and salesman in COEX-Mall. After robots get the profile, they start to find the child while patrolling. At this time, each robot takes at least two roles in *Patrol_COEX* community and *Find_Person* community.



a) Community Situation based Cooperation for *Patrol_COEX* community


Goal Recognition (Find_a_lost_person)
FIND_PERSON_REQUEST
= Patrol_Robot TAKE_REQUEST_FIND_PERSON
Patrol_Robot
Read_Personal_Profile()
Broadcast_Info participant "Find a person" profile
FIND_PERSON
= Patrol_Robot FIND_PERSON
Patrol_Robot
Find_Person(profile)
Guide
Find_Person(profile)
Patrol_Manager
Find_Person(profile)
PERSON_FOUNDED
= Patrol_Robot PERSON_FOUNDED
Patrol_Robot
Announce("Person is founded")
Guide
Announce("Person is founded")
Patrol_Manager
Announce("Person is founded")
Goal Achievement
PERSON_NOT_FOUNDED
= Patrol_Robot PERSON_NOT_FOUNDED
Patrol_Robot
Announce("Person is not founded")
Guide
Announce("Person is not founded")
Report_Police(profile)
Patrol_Manager
Announce("Person is not founded")
Community Fail

b) Community Situation based Cooperation for *Find_Person* community

Fig. 4. Community Situation based Cooperation for level-1 case and level-2 case

6 Conclusion

In this paper, we proposed the community situation based cooperation model and conflict resolution scheme for community computing. Using the cooperation model and policies, we improved the previous community computing model. In addition, to examine the improved community computing model, we introduced case studies. However, our proposal leaves space for further works as follows.

- Improvement of power of community situation based cooperation model and situation model. – This version of community situation based cooperation model is based on strict assumptions.

- Improvement of conflict resolution scheme
- Various case studies

References

1. Blau, J.: Microsoft: Community computing is on the way. InfoWorld Magazine (November 22, 2005), http://www.infoworld.com/article/05/11/22/HNcommunitycomputing_1.html
2. Kumar, M., et al.: PICO: A Middleware Framework for Pervasive Computing. Pervasive Computing 1268-1536, 72–79 (2003)
3. Jennings, R., et al.: Developing Multiagent Systems: The Gaia Methodology. ACM Transactions on Software Engineering and Methodology 12(3), 317–370 (2003)
4. Youna, J., Jungtae, L., Minkoo, K.: Multi-agent based Community Computing System Development with the Model Driven Architecture. In: Proc. of Fifth International Joint conference on Autonomous Agents and Multiagent Systems, May 12, 2006, pp. 1329–1331 (2006)
5. Román, M., Campbell, R.H.: GAIA: Enabling Active Spaces. In: 9th ACM SIGOPS European Workshop, Kolding, Denmark, pp. 229–234 (2000)
6. Brazier, F.M.T., Jonker, C.M., et al.: Formalization of a cooperation model based on joint intentions. In: Tambe, M., Müller, J., Wooldridge, M.J. (eds.) Intelligent Agents II - Agent Theories, Architectures, and Languages. LNCS, vol. 1037, pp. 141–155. Springer, Heidelberg (1996)
7. Hua, C., Gao, J., et al.: AGDRSCOM: A complicated Dynamic Real-time Strong Cooperation System Model. In: Proc. of the Second International Conf. on Machine Learning and Cybernetics, pp. 318–323 (November 2003)
8. Perez, M.S., Sanchez, A., et al.: Cooperation Model of a Multiagent Parallel File System for Clusters. In: Proc. of IEEE International Symposium on Cluster Computing and the Grid, pp. 595–601 (2004)
9. Guo, H., Gao, J., et al.: Recipe, Policy and Self-Organizing: A Hybrid Collaboration Approach for Agent-based Cooperative Design. In: Proc. of the 10th International Conf. on computer Supported Cooperative Work in Design (2006)
10. Jung, Y., Lee, J., Kim, M.: Community Situation based Strict Cooperation Model for Cooperative Ubiquitous Systems. Journal of Convergence Information Technology 2(1) (2007)