An Agent Model for Collaborative Ubiquitous-Computing Environments

Marco P. Locatelli, Marco Loregian, Giuseppe Vizzari
Department of Informatics, Systems and Communication
University of Milano-Bicocca,
viale Sarca 336, 20126 Milano (Italy).
{locatelli,loregian,vizzari}@disco.unimib.it

ABSTRACT

Besides technological challenges, there is a growing interest on how pervasive computing can deeply affect the way in which people interact and accomplish collaborative tasks. Context awareness plays a crucial role for communities, where the degree of participation of a single user dynamically changes in relation to the distance from the place where practices occur and in relation with the ability of gathering the right information at the right time, and in a proper way. In this paper, we adopt an interpretation of pervasive computing as an environment, populated by interconnected smart devices, where users can interact with each other, individually or in groups, to accomplish collaborative tasks. In other words, we aim at modeling, designing and realizing Collaborative Ubiquitous Environments (CUEs). We present an agent based model, balancing the management of such a complex scenario between agents and a structured multifaceted environment.

1. INTRODUCTION

Since Weiser's seminal work [23], the challenges to pervasive computing have changed thanks to the rapid advances in electronics, distributed systems engineering, and mobile computing in general [20]. In particular, several middleware technologies have been proposed as the glue to keep devices connected in a seamless way, and to allow people to take advantage from *cooperating devices*. In spite of a multiplication of technological approaches [14] — to optimize network performance, reliability, and so on — less attention has been paid on how pervasive computing platforms can adapt to the way in which people act, so to reach an optimization in terms of targeted practices and behaviors.

In this work, we adopt an interpretation of a pervasive computing scenario as an environment, populated by interconnected active devices, where users can interact with each other, individually or in groups, to accomplish *collaborative tasks* [4]. In other words, we aim at modeling, designing and realizing Collaborative Ubiquitous Environments (CUEs) [11].

Among the many challenges that these scenarios pose to several research communities in the Computer Science and Engineering areas [25], we focus on the fact that the relevant entities in an pervasive computing system must be able to opportunistically take advantage of mutual interaction to obtain information or to exploit specific services, in order to ultimately satisfy users' needs. These entities should thus be considered as highly autonomous components driven by specific individual motivations, and capable of establishing high-level interactions with each other. These considerations have lead to consider the agent-based and multi-agent systems (MAS) paradigm as a suitable and effective approach to the modeling, design and engineering of CUEs.

In particular, the approach described in this paper is focused on the definition of proper structures and mechanisms for *environments* [24] that can provide agents with information, through their perceptions, and with means of action and interaction. An environment of this kind can be used to structure and organize a system in order to achieve the desired overall resulting behaviour by means of *simple* agent specifications and architectures.

In our perspective, a proper CUE can be enacted only if the constitutive entities are:

- *cooperative*: i.e., able to operate in strict collaboration by coordinating their actions according to specific patterns of interaction. These patterns define *must policies*, supplying coordination patterns to decide who has to be involved in a task and with which role. Members have to respect them under any condition;
- *context-aware*: i.e., able to perceive and share information about the context so to adapt themselves and to collaborate in a loosely coupled manner – without requiring direct contact or communication. This side of interaction defines opportunities rather than strict rules, *may policies*, supplying information to the involved entities in a less prescriptive way, as their execution is decided by evaluating awareness information and members' state.

These two aspects of a CUE must be properly integrated in order to allow the composing parts of the system in choosing and enacting the most suitable actions to be carried out in order to exploit their contextual situation to fulfill users' needs. To support cooperation among entities we provide them both with an environment to *coordinate* their behaviors and with a separate environment to share and perceive *awareness* information. In this work, we consider the agents' environment not only as a necessary element of the MAS model, but also as an exploitable abstraction supporting the

Jung, Michel, Ricci & Petta (eds.): *AT2AI-6 Working Notes, From Agent Theory to Agent Implementation, 6th Int. Workshop,* May 13, 2008, AAMAS 2008, Estoril, Portugal, EU.

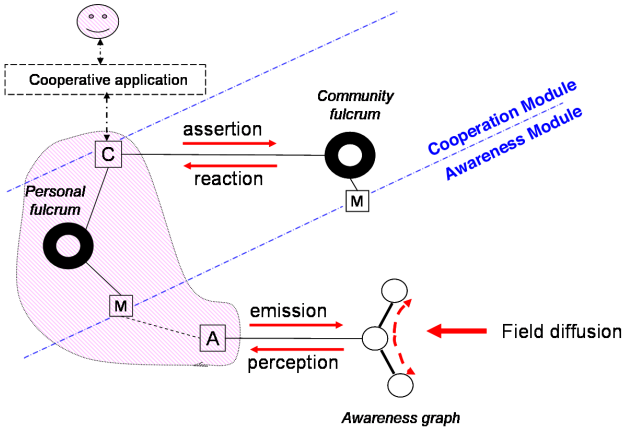

Figure 1: Elements of the reference model.

definition of suitable solutions to face the requirements of CUEs.

The main aim of this paper is to illustrate how a MAS model provide support for the *design* of CUE; the reference model (CASMAS, Community-Aware Multi-Agents Situated Systems [3]) is presented in the following Section. It must be stressed the fact that the model is aimed at the realization of systems that are focused on a given community, encapsulating thus its coordination patterns and the context-awareness policies. These systems are thus aimed at supporting a relatively small number of users (e.g. the persons working and studying in a Department of a University, the different workers of a firm) and technological appliances supporting their activities. The details of the language defined to specify the behaviours of agents in CASMAS is introduced in Section 3. Section 4 presents related work, and concluding remarks end the paper.

2. THE CASMAS MODEL

Ubiquitous-computing systems can shape up as constellations of dynamically defined and interacting *communities* of human and technological *entities*. In the model we are presenting, each entity is represented by two sets of agents belonging to two separate classes — grouped in two logical *modules*, — defined according to roles: agents can either enact *cooperation* mechanisms (within the communities to which entities belong), or supply the entities with information related to context *awareness* [1].

2.1 Cooperation module

Even if CASMAS (Community-Aware Multi-Agents Situated Systems) is designed to provide support to communities of persons, the communities of agents that compose an instance of CASMAS do not try to mimic human behaviors and social organizations. Agents are organized in communities with the purpose of autonomously coordinating their activities and acting to satisfy users' needs. In other words, even if a device is personal and somehow used to qualify a person, its behavior does not directly reflect the behavior of any person: devices interact with each other by means of agents gathering in autonomous communities and enacting (agent-processable community) policies that describe users' needs.

Communities of entities are established in the cooperation module by aggregation points called *community fulcra*. Community fulcra are designed to contain characteristic information: entities gather around them to exploit such information and contribute to the definition of a community. Entities connect to community fulcra by means of cooperation agents (**C-agents**) (Fig. 1).

Entities own a distinct C-agent for each of the communities in which they participate: when created, all C-agents are provided with generic inferential capabilities and with a set of entity-specific statements and rules. By connecting to a community fulcrum, C-agents standing for different entities can share information and acquire community-specific behaviors that are either defined at design time or injected in the fulcrum by other entities.

A *statement* is the unit of information on which an agent reasons and it is characterized by an owner, a list of receivers, and an access control to allow modification only by the owner.

Information (asserted as statements by agents) are shared, and behaviors (rules, also in the form of statements, that express how to infer from other statements) can be asserted in the fulcra and acquired by other agents (Sect. 3). Intelligent behaviors are thus deployed as a result of distributed inferential capabilities of the interconnected computational sites. The blackboard approach makes the environment very flexible with respect to dynamic situations — e.g., entities joining and leaving communities — since variations of interaction patterns among entities can be dynamically managed through mechanisms allowing for the (de)activation of behaviors.

In addition, each entity owns a *personal fulcrum* for its inner coordination (each C-agent is also connected to this fulcrum) and, in case of human entities, for the coordination of the various technological entities that refer to the same person.

The behaviors of C-agents are influenced by the degree of participation of entities in communities, according to additional context information that is supplied by the awareness module. In other words, when a specific application domain is modeled, it is necessary to define which factors concur in the characterization of an entity with respect to a community and the context. The degree of participation is then evaluated by weighting such information according to a domain-specific function.

2.2 Awareness module

The context awareness facet of an ubiquitous computing environment can encompass several different aspects, such as the physical position of an entity as well as its logical (sometimes social) role, such as the responsibility in an organization or a project.

Aim of the **awareness module** is to manage this kind of information, and entities are therein represented by specific awareness agents (**A-agents**).

The (spatial, social, organizational) environments in which entities are situated are modeled as awareness graphs, also called *topological spaces*, and A-agents are connected to their nodes, also called *sites*, accordingly.

Each community can refer to one or more graphs (one for each aspect), therefore, each entity needs an A-agent for each of them. Similarly, the same graph could support more than one community. However, different communities may

have different perceptions of the same aspect (e.g., different notions of distance in a spatial arrangement) thus requiring a more precise definition of interaction on awareness graphs.

The awareness module has been designed according to the perception-reaction paradigm [21]: the behaviors of A-agents are driven by the perception of signals, called *fields*, emitted by other agents and propagated across the awareness graph (e.g., to notify presence in a specific site). The propagation of fields is mediated in intensity according to a graph-specific and community-specific distance function evaluated in and between sites. When a field reaches a site its intensity is computed according to the weight of the arc connecting the current site to the one from which the field is coming, and then the field (if sufficiently intense) can be propagated to adjacent sites. If the field reaches a site where an A-agent is connected, and if it is relevant to the A-agent, the agent evaluates its local intensity according to a characteristic (*sensitivity*) function: possibly, the A-agent can then emit new fields as the result of inner computation, and reach other A-agents.

For example, the closeness of some entities can be considered a fundamental awareness information: A-agents can have a sensitivity function based on "intensity of the perceived presence field greater than x" where a high threshold can be set to perceive entities in the close neighborhood while a lower threshold can be set to perceive also entities further apart.

2.3 Integrating the two aspects

Additionally, A-agents can provide information for the cooperation module, to possibly trigger some reaction in the C-agents. To this aim, special agents bridging information between the two modules are defined: manager agents (**M-agents**) are specialized C-agents that can translate information expressed in terms of fields (exported by the A-agents) into statements that C-agents can understand (by publishing them into the personal fulcrum), and, in the other direction, import fields in A-agents when requests for propagation come from C-agents (e.g., when a device localizes an entity). Finally, each community fulcrum has an M-agent enforcing inter-entity coordination by causing the propagation of general community awareness information on awareness graphs (e.g., causing devices to join the community when they are needed), and by applying access policies and preventing unauthorized entities to join the community.

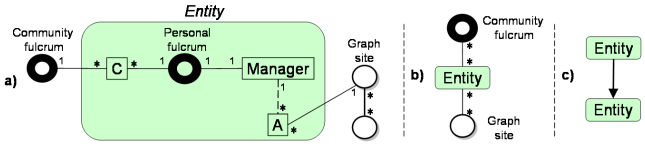

Figure 2: Notation used to represent the entities.

To better represent the instances of CASMAS (e.g., with respect to scenarios), the graphical notation presented in Fig. 2 was defined. Each entity — composed of the elements in a) — is drawn as a rounded rectangle, and can be linked to multiple fulcra and topological spaces. A C-agent is instantiated for each connection to a fulcrum, and an A-agent is instantiated for each connection to a topological space.

An entity such as a person's PDA can be connected to a fulcrum of an entity that represents a person, instead of being connected to a community fulcrum. The notation used to depict such situation is showed in c); in this case, a C-agent is created when an entity connects to another one.

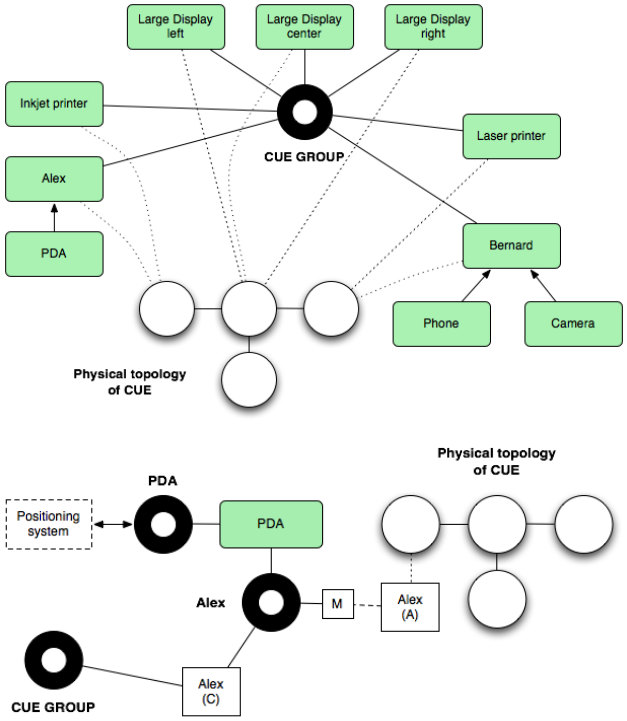

Figure 3: Overall and detailed views of the instantiation of CASMAS for the scenario.

Fig. 3 presents the details of a CASMAS instance supporting a CUE scenario modeled by means of the CASMAS approach. In particular, on the left there is an overall view of a set of CUE (domain) entities that are reified and associated to a CASMAS entity. It must be noted that different technological appliances are represented as entities (e.g. large displays, printers), as well as human actors. They are generally connected both to the workgroup community fulcrum and with the topological space of the building in which the CUE is situated. A *detailed view* of the same instantiation of CASMAS specific for Alex is presented on the right side of the figure: Alex's personal fulcrum, with the associated M-agent, a C-agent that links her to the Atelier community and an A-agent that localizes her on the topological space. Alex's PDA is modeled by its fulcrum and the C-agent linking the PDA to Alex's personal fulcrum.

This structure of the environment allows both the distribution of awareness information, on the topological representation of CUE's physical aspect and the sharing of statements guiding the coordination of activities of entities belonging to the community based in the CUE. The entities act as a bridge between the two aspects of the CUE, being able to perceive and generate awareness information, on one hand, and to exploit it for supporting the coordination of their activities on the other. In the following, the basic linguistic structures to define the behaviours and interactions of the different agents of the CASMAS model will be briefly presented.

3. CASMAS LANGUAGE

A simple language has been defined for CASMAS agents. Basic elements are called *primitives* and allow for simple operations. Primitives can be combined in more complex structures: *mechanisms* or *actions*. The former are employed by C-agents, the latter by A-agents.

A **primitive** implements an elementary function useful for the agents. The primitives used by the C-agents operate on statements and are: *assert*, *modify*, *retract* (erases a statement), *loadBehavior*, *translate*, and *propagate*. The primitives used by the A-agents are: *emit*, *export*, *transport* (moves the agent to another site), and *trigger* (changes the agent's state). These primitives are contextualized in the following where mechanisms and actions that apply them are presented.

3.1 Mechanisms

A **mechanism** is a set of (at least two) rules that implements a mode of interaction that can be adopted by the C-agents. In particular, a mechanism is composed by a rule to *execute* the operation(s) and a rule to *notify* when the mechanism can not be executed — no feedback is otherwise necessary.

A mechanism has the following (basic) form:

```
1  (defrule <execute-rule-name>
2    (<fulcrum-name>::allow (what <wish-statement-name>)
3      (state TRUE) (to ?entityId))
4    (<fulcrum-name>::wishTo (what <wish-statement-name>)
5      <slot>+)
6    <consistency-statement>+
7    =>
8    <operation>+)
9
10 (defrule <notify-rule-name>
11   (<fulcrum-name >::allow (what <wish-statement-name>)
12     (state FALSE) (to ?entityId))
13   (<fulcrum-name>::wishTo (what <wish-statement-name>)
14     <slot>+)
15   =>
16   (assert (_PERSONAL_::notifyFailure (what <wish-statement-name>)
17     [(on <statement-id>)])
```

The activation of the execution rule (line 1) and of the notification rule (10) is guarded by an `allow` statement (2 and 11) with the state set respectively to `TRUE` or `FALSE`, meaning that the entity can (or is forbidden to) execute the mechanism.

If the check is referred to the entity that is executing the mechanism, then the `?entityId` (3) should be changed to `?myId`; if it is referred to the requester, it should be changed to `?ownerId` (matched on the `owner` slot of the `wishTo` statement, that is a basic property of each statement). Moreover, both rules match a `wishTo` statement that expresses the request for activation of the mechanism by an entity. The execution rule can check other consistency conditions for the specific mechanism, for example, a device with a small screen can ensure a proper display only of short messages:

```
1  (defrule _COMMUNITY_::showMessage
2    (_COMMUNITY_::wishTo (owner ?applicantId)
3      (what "showMessage") (message ?message))
4    (_COMMUNITY_::allow (what "showMessage")
5      (state TRUE) (to ?applicantId))
6
7    (test (< ?message ?MAX_LENGTH))
8
9    ?display<-(display)
10   =>
11   (modify ?display (currentMessage ?message))
12 )
```

Mechanisms can be generally classified according to the purpose for which they are designed, i.e., according to the kind of interaction between agents they supply (Fig. 4):

Translation mechanisms are used by C-agents to move information from a source fulcrum to a target one. The information to be moved and the possible elaboration of such information (before it is entered in the target fulcrum) are expressed as statements, and the translation is enacted as an assertion in the target fulcrum.

Membership mechanisms are used by M-agents of personal fulcra to induce the acquisition of behaviors by the C-agents of the entity from the community to which they belong. In practice, M-agents assert a `memberOf` statement in the personal fulcrum; the statement quantifies the degree of participation of the entity to the community. Degree computation is based on awareness information perceived by the entity in relation to the specific community. Then, C-agents can activate the rule of the membership mechanism to load the behavior according to the degree of participation through the `loadBehavior` primitive. Membership mechanisms do not contain a `wishTo` statement since they only describe mandatory C-agent reactions to the presence of the `memberOf` statement.

Propagation mechanisms are used by M-agents — of any fulcrum — to respond to the willingness of C-agents to propagate awareness information within an awareness space by using the `propagate` primitive. Propagation mechanisms are in charge of the conversion between a statement and the field's format representing information perceived and to be propagated by the A-agents on the topological space.

Awareness mechanisms are used by M-agents to insert into their fulcrum the awareness information contained in the above mentioned fields; insertion is accomplished by means of the `assert` primitive. Awareness mechanisms are in charge of the conversion between the field's format and the `awareOf` statement representing awareness information in the target fulcrum.

3.2 Actions

The A-agents have a small set of **actions** that specify whether and how agents change their state or position, and propagate information on the topological space. *Trigger* and *transport* are the actions to change state and position, respectively.

Trigger defines how the perception of a field causes a change of state in the receiving agent, while transport defines how the perception of a field causes a change of position in the receiving agent. *Emit* is the action to propagate information on the topological space; this action can also be viewed as the activating part of a sort of asynchronous communication among agents. In fact, asynchronous interaction among agents takes place through a field emission–propagation–perception mechanism. An agent emits a field when its state is such that it can be source for it. Field values propagate throughout the space according to the diffusion function of the field. Field diffusion along the space allows the other agents to perceive it. Perception function, characterizing each agent type, defines the second side of an asynchronous interaction among agents: that is, the possible reception of broadcast messages conveyed through a field, if the sensitivity of the agent to the field is such that it can perceive it.

In CASMAS these actions are implemented as rules, and

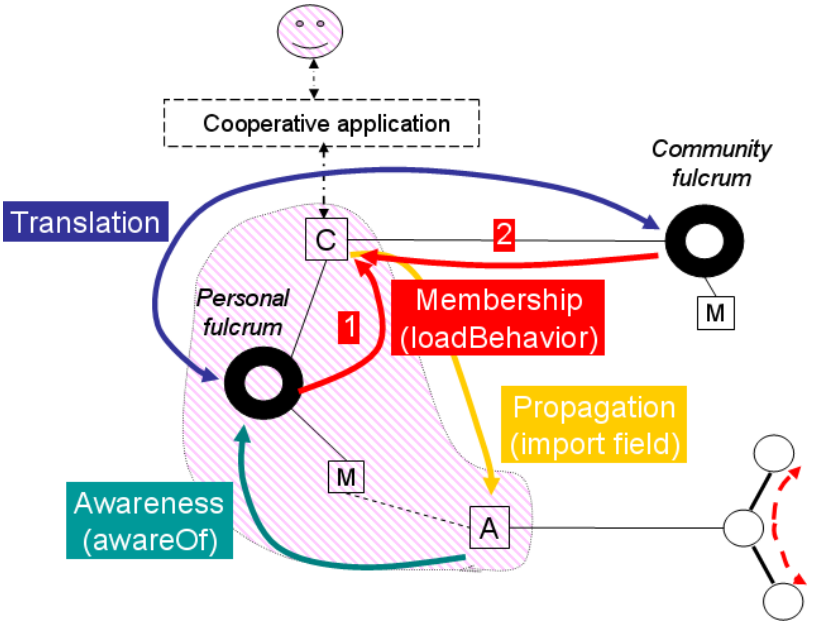

Figure 4: Mechanisms provided by CASMAS.

they can fire both on an `importedField` and on a field perceived from the site where the agent is situated. In the following, `_AW_` is where imported fields are asserted by the M-agent, `_SPACE_` is the topological space where the A-agent is situated.

The `situated` fact reifies on which site an agent is situated. The match between the `siteId` of `field` and the one of `situated` ensures that the field is perceived from the site where the agent is situated.

```
(defrule transport
  (or
    (_AW_::importedField (type ?type) (intensity ?intensity)
      (data ?data) (to ?myId))
    (_SPACE_::field (site ?siteId) (type ?type)
      (intensity ?intensity) (data ?data))
  )
  [<conditions-on-field>]
  (_SPACE_::situated (agent ?myId) (onSite ?siteId))
  (test (neq ?siteId ?NEWSITE_ID))
  (state ?state)
  [<conditions-on-state>]
  =>0
  (transport ?NEWSITE_ID)
)
```

The `NEWSITE_ID` must be the site where the agent has to move in reaction to the perceived field. If necessary, the destination site can be computed instead of defined explicitly. The `transport` is executed only if the destination site differs from the current site where the agent is situated.

```
(defrule emit
  (or
    (_AW_::importedField (type ?type) (intensity ?intensity)
      (data ?data) (to ?myId))
    (_SPACE_::field (site ?siteId) (type ?type)
      (intensity ?intensity) (data ?data))
  )
  <conditions-on-field>
  (_SPACE_::situated (agent ?myId) (onSite ?siteId))
  (state ?state)
  [<conditions-on-state>]
  =>
  (emit "FIELD_TYPE" ?intensity [?data]))
```

In accordance with the model, the emit is possible only on the site where the agent is situated, so the site is not a parameter for `emit`. Moreover, the `data` parameter is used to associate an information to the field and it is optional.

```
(defrule trigger
  (or
    (_AW_::importedField (type ?type) (intensity ?intensity)
      (data ?data) (to ?myId))
    (_SPACE_::field (site ?siteId) (type ?type)
      (intensity ?intensity) (data ?data))
  )
  <conditions-on-field>
  (_SPACE_::situated (agent ?myId) (onSite ?siteId))
  (state ?state)
  [<conditions-on-state>]
  =>
  (trigger ?NEWSTATE)
)
```

If conditions are satisfied, a change of state is triggered.

4. RELATED WORK

The focus of this paper is on the design of pervasive computing environments, particularly with respect to communities of persons: such communities are first-class concepts both in CASMAS and in the realized supporting infrastructure and, therefore, they are present both at the model and the implementation level. From this point of view this work differs from other existing proposals that employ agents and agent–based infrastructures simply as a middleware for the design and implementation of pervasive computing systems (see, e.g., [8, 18]).

Moreover, CASMAS is a model to let software applications coordinate and share awareness information; it is not mainly intended to (re)implement software applications, although it is possible. Instead, approaches such as BEACH [22] or iROS [6] provide support to the development of software applications from data management up to user interfaces. Differently from these proposed approaches, CASMAS defines the concepts of community fulcrum, topological space,

and entity, and collaborative ubiquitous-computing environments are built on them.

CASMAS agents are programmed declaratively (by rules), and they can be classified as reactive agents [5]. Differently from the FIPA approach[1], that is based on the *direct* exchange of messages, and in general from approaches based on Agent Communication Languages (ACLs) [7], interactions among CASMAS agents are mediated by a space both in the coordination module, by exchanging information in fulcra, and in the awareness module, by propagating information on topological spaces.

The **coordination module** is naturally implemented through DJess because it allows to directly map the agents and the fulcra to the inferential systems and to the shared working memory, respectively. Moreover, it provides some useful features such as the sharing of rules that are fully exploited by CASMAS. Among other platforms for pervasive computing environments, the following experiences seem interesting for comparison:

The *Gaia* project is a leading effort in the field of pervasive computing systems. Gaia [19] is an operating systems for pervasive computing environments that are called Active Spaces. In an Active Space everything (users, actions, contexts) is managed by agents with different specifications, belonging to different classes — like CASMAS agents. A generic distinction defines Context Providers, Context Synthetizers and Context Consumers. In the last class are essentially all the application agents. As for CASMAS, this separation requires that each agent embeds some logic capability but information on context is replicated across several agents, and not shared to gather agents, also the characterization (i.e., the configuration) of similar entities is totally independent from an agent to another.

Finally, with respect to the **awareness module**, other approaches have been analyzed during our research, before basing the current implementation on DJess:

- a *framework for MMASS based applications* [2], supplies computational support to the abstractions and mechanisms defined by the model, and if properly integrated with a suitable middleware (the MAIS reflective architecture [17]) supporting network communication it can provide a reasonable support to the realization of the awareness model. However, this framework lacks the possibility of giving declarative descriptions of behaviors, a feature that is currently exploited to facilitate the task of deploying CASMAS modules;

- an *artifact–based approach* [15], supporting the design and engineering of MAS environment, through the adoption of TuCSoN tuple centres [16], could be adopted to represent the internal spatial structure of the topological layer as well as the various related mechanisms (e.g. field diffusion), realized in the form of reaction rules considering the TuCSoN [16] approach to artifact implementation. This approach, however, is mainly focused on the realization of infrastructures supporting agent interaction and coordination, while DJess also supports the realization of simple reactive agents and to describe their reactive behaviors in term of rules, and to distribute the execution of agents and topological spaces in a transparent way for the system designer and of course for the agents too;

- the *TOTA middleware* [13] offers a rich and sophisticated support to the design and engineering of Pervasive Computing applications exploiting the abstractions of agents and MAS environment. However, one of TOTA's most distinguishing features is the possibility to diffuse and keep updated context–awareness information in a dynamic environment, and in particular it offers the possibility to maintain the structure of Co-Fields over a changing network. In this case, however, the structure of the network is not very important in determining the context of a given node. Instead the CUE scenario generally requires the capability to integrate several topological spaces, related to different aspects of the CUE.

[1]Foundation for Intelligent Physical Agents (FIPA) standard specifications available at `http://www.fipa.org/repository/standardspecs.html`

5. CONCLUDING REMARKS

This paper has briefly presented an approach to the modeling and design of pervasive computing systems supporting the coordinated activities of members of communities. In particular, the CASMAS model was introduced as a support for the representation and management of both awareness and coordination facets of a CUE. The CASMAS conceptual model allows for the design of systems focusing more on collaborative tasks — application objectives — rather than on technological issues. The model was applied to represent and manage different types of CUES, in the healthcare context [10] and for the definition of smart environments in a educational facilities [9, 12]. Future works are aimed on one hand at a concrete experimentation of the approach and the supporting infrastructure in the above introduced contexts.

In addition to this form of evaluation of the adequacy of the modeling approach and the realized supporting infrastructure, we also working on instruments supporting a simple configuration of the CUE, its structure, rules and laws for the management of awareness and coordination policies. This aspect is particularly important due to the specific nature of this kind of application: in fact, even if a thorough analysis of methodological aspects of CUE definition must still be carried out, we envisage a specific role of CUE designer to manage the phases leading from an analysis of the specific application scenario (including the involved actors and devices, as well as the relevant facets of the specific notion of context), to the definition of a set of spatial and logical layers of representation, each one endowed with an internal structure, active entities and patterns of interaction among them.

6. REFERENCES

[1] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles. Towards a better understanding of context and context-awareness. In *HUC '99*, pages 304–307, London, UK, 1999. Springer-Verlag.

[2] S. Bandini, S. Manzoni, and G. Vizzari. Towards a platform for multilayered multi agent situated system based simulations: focusing on field diffusion. *Applied Artificial Intelligence*, 20(4–5):327–351, 2006.

[3] F. Cabitza, M. P. Locatelli, M. Sarini, and C. Simone. CASMAS: Supporting collaboration in pervasive

environments. In *Pervasive Computing and Communications, 2006. PerCom 2006. Fourth Annual IEEE International Conference on*, pages 286–295. IEEE, 2006.

[4] R. H. Campbell. Beyond global communications: the active world. In *PerCom*, page 211. IEEE Computer Society, 2005.

[5] A. Hector. A new classification scheme for software agents. In *ICITA '05: Proceedings of the Third International Conference on Information Technology and Applications (ICITA'05) Volume 2*, pages 191–196, Washington, DC, USA, 2005. IEEE Computer Society.

[6] B. Johanson, A. Fox, and T. Winograd. The interactive workspaces project: experiences with ubiquitous computing rooms. *IEEE Pervasive Computing*, 1(2):67–74, 2002.

[7] Y. Labrou, T. W. Finin, and Y. Peng. Agent communication languages: the current landscape. *IEEE Intelligent Systems*, 14(2):45–52, 1999.

[8] T. C. Lech and L. W. M. Wienhofen. AmbieAgents: a scalable infrastructure for mobile and context-aware information services. In *4rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2005), July 25-29, 2005, Utrecht, The Netherlands*, pages 625–631. ACM Press, 2005.

[9] M. P. Locatelli and M. Loregian. Active coordination artifacts in collaborative ubiquitous-computing environments. In B. Schiele, A. K. Dey, H. Gellersen, B. E. R. de Ruyter, M. Tscheligi, R. Wichert, E. H. L. Aarts, and A. P. Buchmann, editors, *AmI*, volume 4794 of *Lecture Notes in Computer Science*, pages 177–194. Springer, 2007.

[10] M. P. Locatelli and C. Simone. Supporting care networks through an ubiquitous collaborative environment. In C. Nugent and J. Augusto, editors, *Smart Homes and Beyond*, volume 19 of *Assistive Technology Research*. IOS Press, 2006.

[11] M. P. Locatelli and G. Vizzari. Awareness in collaborative ubiquitous environments: the multilayered multi-agent system approach. *ACM Transactions on Autonomous and Adaptive Systems*, 2(4), 2007.

[12] M. P. Locatelli and G. Vizzari. Environment support to the management of context awareness information. In D. Weyns, S. Brueckner, and Y. Demazeau, editors, *Proceedings of Engineering Environment-Mediated Multiagent Systems 2007*, pages 162–169, 2007.

[13] M. Mamei and F. Zambonelli. Programming pervasive and mobile computing applications with the TOTA middleware. In *2nd IEEE International Conference on Pervasive Computing and Communication (Percom2004)*, pages 263–273. IEEE Computer Society, 2004.

[14] C. Mascolo, L. Capra, and W. Emmerich. Middleware for mobile computing (a survey). In E. Gregori, G. Anastasi, and S. Basagni, editors, *Neworking 2002 Tutorial Papers*, volume 2497 of *lncs*, pages 20–58, 2002.

[15] A. Omicini, A. Ricci, M. Viroli, C. Castelfranchi, and L. Tummolini. Coordination artifacts: environment-based coordination for intelligent agents. In N. R. Jennings, C. Sierra, L. Sonenberg, and M. Tambe, editors, *3rd international Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004)*, pages 286–293. ACM Press, 2004.

[16] A. Omicini and F. Zambonelli. Coordination for Internet application development. *Autonomous Agents and Multi-Agent Systems*, 2(3):251–269, Sept. 1999. Special Issue: Coordination Mechanisms for Web Agents.

[17] B. Pernici, editor. *Mobile Information Systems: Infrastructure and Design for Adaptivity and Flexibility*. Springer, 2006.

[18] M. D. Rodríguez, J. Favela, A. Preciado, and A. Vizcaíno. Agent-based ambient intelligence for healthcare. *AI Communications*, 18(3):201–216, 2005.

[19] M. Román, C. Hess, R. Cerqueira, A. Ranganathan, R. H. Campbell, and K. Nahrstedt. Gaia: a middleware platform for active spaces. *SIGMOBILE Mob. Comput. Commun. Rev.*, 6(4):65–67, 2002.

[20] M. Satyanarayanan. Pervasive computing: Vision and challenges. *IEEE Personal Communications*, pages 10–17, Aug. 2001.

[21] C. Simone and S. Bandini. Integrating awareness in cooperative applications through the reaction-diffusion metaphor. *Computer Supported Cooperative Work*, 11(3-4):495–530, 2002.

[22] P. Tandler. The beach application model and software framework for synchronous collaboration in ubiquitous computing environments. *Journal of Systems and Software*, 69(3):267–296, 2004.

[23] M. Weiser. The computer for the 21st century. In *Scientific American*, volume 265 of *3*, pages 94–104, 1991.

[24] D. Weyns, A. Omicini, and J. Odell. Environment as a first class abstraction in multiagent systems. *Autonomous Agents Multi-Agent Systems*, 14(1):5–30, 2007.

[25] F. Zambonelli and H. V. D. Parunak. Signs of a revolution in computer science and software engineering. In *Proceedings of Engineering Societies in the Agents World III (ESAW2002)*, volume 2577 of *Lecture Notes in Computer Science*, pages 13–28. Springer–Verlag, 2002.