Distributed Shared Contexts

Rosa Alarcón[1], César Collazos[2], and Luis A. Guerrero[1]

[1] Department of Computer Science, Universidad de Chile,
P.O. Box 2777, Santiago, Chile
{ralarcon,luguerre}@dcc.uchile.cl

[2] Systems Department FIET, Universidad del Cauca,
Campus Tulcan, Popayán-Colombia
ccollazo@unicauca.edu.co

Abstract. Mobile solutions have gone beyond the role of personal tool to offer solutions in supporting coordinated work. Mobile workers shift constantly from individual to group work, access shared virtual environments from different devices and can create new elements when disconnected (annotations, appointments, etc.). However, current approaches focus either on individual or group work, as well as on the restrictions and affordances that mobile devices and mobility provides, but they do not address the huge heterogeneity, inconsistencies and complexity derived when both working modes are mixed. We are interested in the construction of virtual environments that can be accessed transparently anywhere, anytime on anything, but also exploit the advantages that mobility aware technologies provide. Context-based design facilitates the construction of such environments because they isolate and provide structure to the application layer facilitating adaptation to devices' characteristics, reducing the impact of devices heterogeneity and easing the shift among different working modes.

1 Introduction

When designing applications for mobile technology, the focus often relies on the restrictions and affordances that those devices provide, that is their need for dynamic reconfiguration, adaptivity, asynchronous interaction, context-awareness and lightweight middleware [1]. For instance, Personal Information Management (PIM) applications, which include calendars, to do's, notepads, e-mail alert among others; deal with issues like limited screen size, storage and memory and disconnected operations. Their main concern is data synchronization between the mobile device and the PIM system. Although first approaches conceived a PIM as a personal server running on user's desktop computer, current designs are based on Web servers and are integrated into enterprise's strategies. Now, personal information can be accessed from different devices (i.e. handhelds, data-enabled cellular phones, pagers, badges, etc.), and synchronization occurs not only between the PIM and each device, but also with organizations' databases, other networked applications and groupware.

Other approaches take advantage of different wireless networks like cellular, LAN, PAN (Personal Area Network) or BAN (Body Area Network) networks [2]. Although previous design restrictions applies in this kind of networks, users can update their versions more frequently and access huge databases on demand, or the information can be made available (pushed) to them proactively, based on their context. This area of research is know as context-aware computing [3], which are systems that examine and react to an individual's changing context. Typically "user context" is described in terms of the user's current location taking into account the *social* (at home, at the office, etc.[4], [5], [6]), *physical* (light, noise, etc.[7]) and *informational* (guided tours, tourist maps, etc. [8], [9]) characteristics associated to the user's current physical environment.

But mobile solutions have gone beyond the role of a personal tool to offer large-scale solutions in supporting coordinated work for teams, groups, and organizations. The adoption of mobile devices is most clearly seen in the professional world, for instance, IDC forecasts that the remote and mobile workforce will grow, in the US, to 47.1 million by the end of 2003 (37% of total workforce). Traditionally, work groups have been supported by collaborative systems that allow them sharing software artifacts, objects and self-representations enabling a virtual *shared environment.*

For instance, a mobile healthcare system supports workers' activity, but also information exchange, collaboration, coordination of activities and resources and decision-making [10]. Furthermore, based on the relevance of the events that occur in the shared environment and the current user context, group members can be contacted proactively (push) through different devices either fixed or mobile [11]. In both cases contextual information related to workgroup activities (i.e. activities' status, ongoing activities, changing artifacts, etc.) is presented to users so they can be aware of the status and progress of their work and can coordinate their future actions.

Mobile computing has become a useful and promising technology for supporting individuals, but as their potentiality is being acknowledged for professionals, it is creating new interaction paradigms for supporting work groups and organizations and adding also huge heterogeneity. However, it is very interesting that when designing context-aware systems, information must be adapted to user context; but when designing groupware systems, we are dealing with the provision of contextual information related to group activities, without taking into account users context when receiving the information; something similar occur with classic PIM systems.

Our aim is to support the development of shared applications that can be accessed anywhere, anytime on anything, but current approaches support either individuals or work groups and are designed separately or are considered merely extensions of each other, making it hard to design applications that can be easily adapted to different interaction modes (i.e. disconnected operations, mobile computing, nomadic computing, groupware applications, etc.).

In this paper, we propose a design based on multiple, possibly competing contexts (individual vs. shared) that represent users' activities, actions and status at a high level. The set of contexts are shared by people and agents, accessed from different devices and adapted to users' current context. By designing an application at a meta-level, the transition between different interaction modes is reduced but it is possible to provide richer contextual information. The strategy and the proposal draw from previous experiences in the subject, although our current approach is more general.

Section 2 discusses in detail the conceptual definition of context while section 3 presents the architecture proposed. Section 4 present an application based on the architecture of section 3. Finally, section 5 presents our conclusions.

2 Shared Contexts

Although the word "context" appears repeatedly in the area of context-aware computing there are not a consensual definition of its meaning. This is perfectly understandable as concept meaning is only valid in the scope of a determined context.

In broader terms context can be defined as "the interrelated conditions in which an event, action, etc. takes place"[1]. Because this definition is too general, researches have tried to determine which is the situation at hand and then, which conditions must be considered as part of its context (engineering point of view [12]).

2.1 Related Work

In linguistics and natural language research, context has been used as an interpretation medium for establishing the meaning of a sentence. For instance if we say: "I like to play with my sister", it is assumed that my sister and I are playing together and not that she is a toy. That is, social context narrows down the proper interpretation of an expression [13]. It is clear that a context can serve for restricting the solution's state space for problems related to automatic reasoning [12], or when huge amount of data is searched. For example, Web searching systems filter the information by estimating its relevance in determined contexts of interpretation such as pages' popularity (Altavista, Google), categories (Vivisimo), geographic zones (Lasoo), etc.

In the area of context-aware computing, context is usually related to the "computing" conditions in which users are immersed (network connectivity, communication bandwidth and nearby resources), their "personal and social" conditions (user's location, people nearby and social situation, for instance, Office Assistant [4], Conference Assistant [5], GeoNotes [6]), the "information" associated (guided tours, tourist maps, etc., for instance, CoolTown [8], Tourist Guide [9], Cyberguide[5]) or the "physical" conditions of users' surroundings (lighting, noise, etc., for instance, Tangible Bits [7]) [3]. Others, identifies *primary* (location, entity, activity and time) and *more complex* context derived from the primary ones [6].

Context is used for interpreting user's needs and intentions, and adapting the application (reconfiguration or information) or the environment itself (proximate selection, adaptable environments) according to them [3], [2]. Typically, context has been studied by considering the interaction between an individual and a context-aware or a knowledge-based system, but as professionals and organizations are adopting this technology, more work group applications for mobile platforms are being designed.

In groupware, an underlying mechanism, called "Awareness", renders the shared environments' state and changes to users [14], providing contextual information (where, when, what, who, how) so users *understand* how their actions fit in the group goals and can regulate their behavior [15], [16]. Contextual information is presented as visual, aural or physical gadgets (i.e. "new" icons in BSCW[2] systems). However, shared context is unstructured, implicit, created by users' actions and restricted by the application and this same approach is used when handheld devices are involved.

2.2 Distributed Shared Context

When users work together, they share a context that comprehends both, high level elements such as users' actions (and their consequences) and low level elements such as users' computing situation (and their restrictions). For instance, AwServer [11] is a multi-agent system that delivers contextual information about users' actions (high level of abstraction) trough different Internet-based channels *proactively*. Users are

[1] See Merriam-Webster Dictionary at http: // www.m-w.com

[2] For more information on BSCW see http: // bscw.fit.fraunhofer.de/

contacted through e-mail, private messaging systems and cellular phone or regular groupware widgets; depending on the relevance of the information, users' availability (i.e. from 8:00 to 19.00) and devices' characteristics (low level of abstraction).

Again, context serves as an interpretation medium to infer information relevance (on the context of work at a high level) and users' disposition to be interrupted (on the context of users' devices capability and social situation at a lower level). Additionally, by having a context designed at a high level, it significantly facilitates the construction of applications that extend and ground context's semantics [18]. However, we argue that in [11] devices are merely seen as different interfaces of a unique virtual space through which the system propagates contextual information but, although it supports adaptation (as applications can react to contextual information), the workload (actions) created in portable devices is neglected.

Actions and conditions related to mobile devices (and hence, context) could be very different from those related to more powerful, but fixed devices. So, although some coherence in the interfaces for both systems is required, this does not implies that mobile applications are necessarily a smaller subgroup of fixed applications or just a reduced interface. For instance, when a group works together in a shared editor, users go through phases of *self-organization* (of ideas, thoughts and critics) and then, once they have a clear picture of their contribution they go to a *sharing* phase where they can argue and negotiate [19]. Clearly, although users share the same objects (i.e. a paper), their activities and are slightly different in both cases, that is, the context that they share includes isolated tasks as well.

This way, context is spread or distributed among different devices (and perhaps applications). From this perspective, a context-based design approach can ease the construction of applications where transition among different interaction modes (i.e. working isolated vs. working together) is facilitated for end-users and developers.

3 Proposed Architecture

Context must be represented through structures, schemata, scripts, rules, etc. Those representations capture the static aspect of context at design time; dynamic aspects are created when people or agents interact. We have designed a FIPA [19] compliant multi-agent system called "DSC Server" (Distributed Shared Context) where a set of agents, named "Context Managers" (CM), maintains the distributed shared context.

DSC Server is an extension of [11]. Users' actions are interpreted through a *work context* that must be instantiated in order to reflect the task at hand (called a *local vocabulary*). Both contexts are represented as ontologies (concepts, predicates agent actions) [20] implemented as java classes of JADE (see [21] for further information).

Events in a groupware application (Gevent) are sent to a Context Manager agent (CM), which infers its relevance in the work context; then, a User Context Agent (UCA) propagates the event to the user interface (classic feedback) or another group member (user 2 in Fig.1). Grayed components of the figure are fully described in [11] and [17] and are not the main concern of this paper. CM agents move to portable devices, when working in isolation, events correspond to a Local context (Levent).

DSC is designed in two conceptual layers (local vocabulary): one describing the *structure of the document* where users work on, and the other the set of possible user *actions* and the application *status* (considered as a Finite State Machine) (Fig.2).

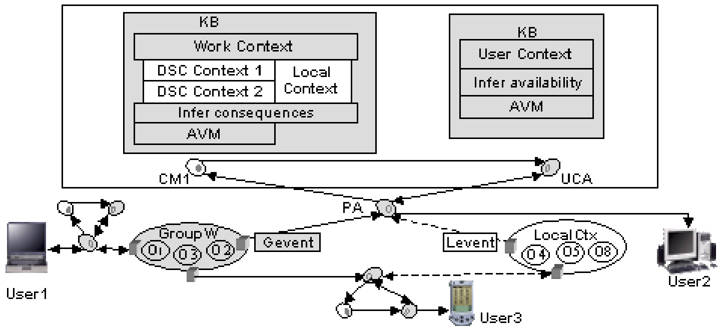

Fig. 1. System overview. Events occurred in a groupware application (*Gevent*) are propagated to *DESC* server through a *Proxy Agent* (PA). A *CM* agent is associated to each *user*. Local events (*Levent*) are propagated to the server as well

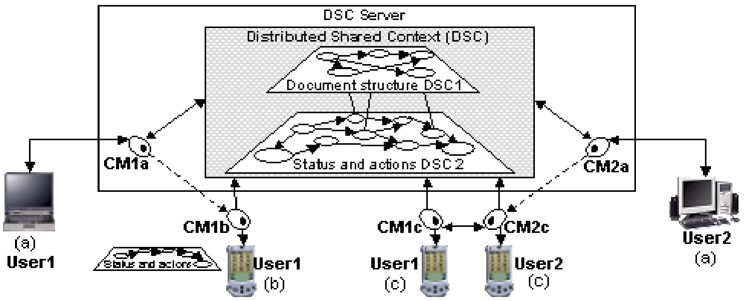

Fig. 2. System architecture. A *CM* agent is associated to each *user*. Agents maintain a *distributed shared context*, and support 3 *interaction modes*, which are classic groupware interaction (a), isolated work (b) and collaborative interaction based on mobile computing

3.1 Interaction Modes

Classic groupware interaction. In fig.2 (a), user 1 (represented by a laptop) and user 2 (represented by a desktop computer) work collaboratively in a classic groupware interaction (i.e. both users work together editing a document at the office). Every action performed in the application is sent to the agent associated to each user (CM1a and CM2a, respectively), which in turn updates the DSC.

Isolated work. When user 1 moves to his handheld (that is, stop working on his laptop and logs in-to his handheld), the agent moves to the handheld (CM1b) and, if there is not a wireless network available, the agent request the user to synchronize the application so that, a part of the context that is suited to the handheld is downloaded. Now, user 1 is able to work alone on his mobile device independently and other users can be aware that user 1 is on his moving and working alone (i.e. commenting a document when he is in the subway). See fig.2(b).

Wireless support. User 2 also moves to his handheld and walks around his office with wireless network support (fig. 2(c)). The agent moves also to his handheld (CM2c) and user 2 can continue his work. However, as we argued before, the possible actions that a user can perform in a handheld are different than those actions that could be executed in a desktop computer. But, similarly to (b) interaction, user 2 now can access only some part of the context and his interface adapts itself to the possibilities (i.e. present less or different menu items).

Finally when user 1 finally arrives to users' 1 office and acknowledges the existence of wireless network support, the agent (CM1b) asks user 1 if he likes to exploit the network and if so, CM1b synchronizes his actions with the DSC (updates DSC concepts at layer 1 and 2, personal activities are synchronized as local context for each user). If conflictive actions have occurred (another user has modified also the same paragraph), a new version for the document item is created (a new version of the paragraph). Later on, the user must negotiate with his co-author the final version.

Once the agent has updated the DSC, user 1 works now with the support of the wireless network (c). It is also possible that user 1 and 2 meet (resource discovery) and decide to work together on his handhelds (i.e. they could be at the cafeteria).

3.2 Agents' Structure

Users interact with the DSC Server through a CM agent. These agents restrict the global DSC according to the current restrictions and opportunities that devices provide, but also capture actions that users perform when working isolated. The set of actions and states that reflect the isolated working mode is called "Local context" and in addition to document structural units (DSC layer 1) is used to construct a local Knowledge Base for the CM agent (Fig.3).

Although it is possible for agents to reason based on this KB, up today we have not designed specialized reasoning tasks that support isolated work, but we plan to do so as future work in order to determine the impact of devices restriction (limited storage, battery, etc.). When synchronization is required only actions and concepts related to DSC1 are uploaded to the DSC server. We have implemented an interface mapping function as well; with this approach interfaces' elements like menu items are generated automatically according to the local context (the set of possible actions).

Contexts based design makes possible to integrate different devices (and their interfaces) straightforwardly. However, since contexts are represented as ontologies (concepts, predicates and agents actions), the main drawback is that it requires extra effort from developers, which can be demanded if they are not used to model domains in those terms. On the other hand, end users can switch from isolated to shared work smoothly as well as from different devices and applications: they can switch between a java editor, a web-based schedulers or a web-based project manager designed to share the application layer at a conceptual level, that is their context ontologies.

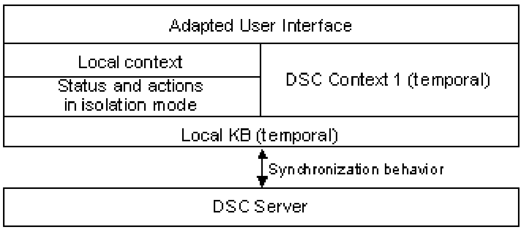

Fig. 3. Agents' behavior. A *CM* agent *adapts* the *user interface* according to the *local context* and *synchronizes* the local changes with the *Distributed Shared Context*

4 An Example Application

D-Write (Distributed Writing) is a collaborative writing tool based on DSC Server, designed to support three interaction scenarios: a user management component, a Web editing module and a PDA text-editing module. For implementing the Web and PDA version, it is only necessary to design the Distributed Shared Context (DSC) in both layers: Document structure (fig. 4(a)), and status and actions (fig. 4(b)).

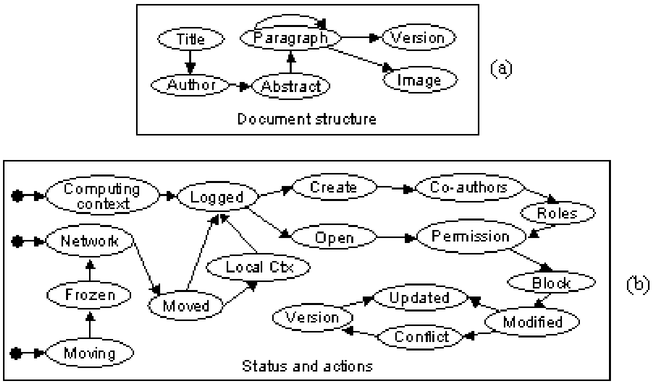

Fig. 4. Distributed Shared Context. Application context is described through *document structure* and the set of possible *states and actions* available for each device

In fig. 4(a), a D-Write document contains a title, an author, an abstract and a series of paragraphs, which can contain images and can have associated a version, those are represented as concepts and stored in a KB (JESS[3] facts). The minimal unit (atom) is a paragraph. For the sake of simplicity a partial view of the possible states is shown in Fig.4(b). In the figure, the nodes denote the states while the arcs the actions that cause a state change; again for simplicity arcs are not labeled in the figure.

As we can see, when users start the application (i.e. from a PDA or making a request to a Web site), the application recognizes its computing context, that is, whether it is running on a PDA or in a Web client. Depending on the computing context other procedures are available for the user; those are: authentication, document manipulation (create or open documents, determine co-authors and their roles and hence the set of permissions assigned to the user, modifying documents and handling conflicts and versioning), moving to another device and downloading a restricted context (a subset).

The user interface is adapted according to the devices type: functions available for the device are mapped from the context available for the device. For instance in fig. 5(a), we present the Web version and in fig. 5(b), the PDA version. Both interface designs are similar although the PDA module is smaller and specific. Entrance to the system is done through the main page. Here, basic data for connection to the server is initialized. Co-authors may create new documents or open previous ones (fig. 4(b)). Shared documents may have several versions, which can be navigated by a co-author, however, even though personal annotations keep versions as well, only its owner can

[3] JESS is the Java Expert System Shell. See http://herzberg.ca.sandia.gov/jess/ for more information.

access them and because they are stored in DSC Server, user could work isolated whenever they require it in any device (perhaps her notebook, at home).

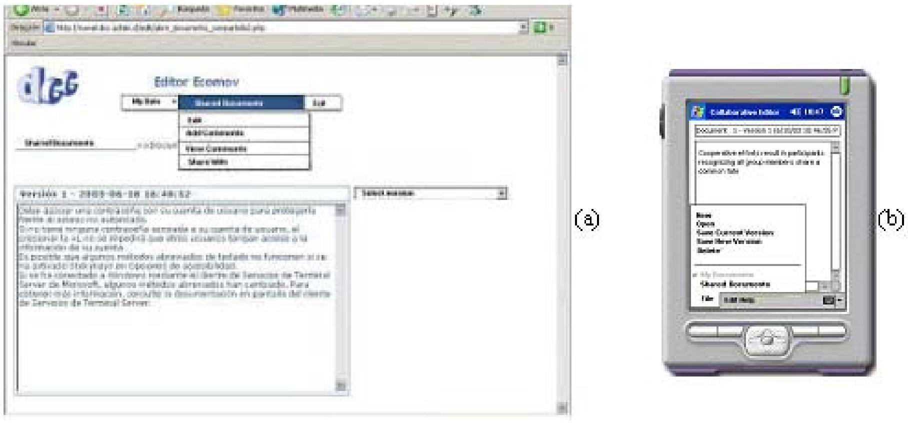

Fig. 5. D-Write editor allow users to edit a shared document from a Web browser and a PDA editing module. The PDA allows making personal *annotations* as well

5 Conclusions

The aim of this paper is to understand the distinction between personal and public artifacts and how a context-based design can facilitate the construction of environments that can be accessed anywhere, anytime on anything. By modeling the set of possible actions available for a particular device, we maximize its particular strengths and capabilities. By designing applications at a high level of abstraction, adaptation different modes of interaction and device's restrictions are eased. This recommendation mirrors Norman's advocating of information appliances [22].

In humans, awareness is a complex mental state through which individuals understands their immediate context and its changes and can regulate their actions and acquire new information [23]. Although not fully understood, it seams that we associate actions with multiple contexts at a time. For an observer, a user action conveys much information depending on the action itself, the context where it occurs and the observers' own experiences.

In [11] users' actions are interpreted on the context of work and practical inferences are made (responsibility, failure of expectations, etc.), however, when interactions are more flexible and are not restricted by work constraints (i.e. collaborative creation of a document, artistic work, etc.) the relations are relaxed so much that the inferences are not very interesting (perhaps because work context does not apply anymore). Because we are interested in more relaxed environments (such as learning environments), in this paper, we want to explore a more general approach.

The approach allows us to build applications that can be accessed from different devices, adapt to them and fully exploits their capabilities. However some questions arise. For instance, it can be argued that modeling every possible user action is not possible. This concern is closely related with the dynamic aspect of context: contexts change over time. It is possible that processing power required to keep KBs of different context could be a issue, up today we have not built more complex examples (like a shared scheduler or a project manager), but we are planning to do so, in order to get an insight on developers' extra effort required, as well as technology restrictions. On the plus side, as agents' mobility and synchronization tasks are encapsulated in the DSC-Server, an application can be accessed for different devices that we could not anticipated before.

Today, PDAs are perceived as highly personal devices containing specialized person information, while single display groupware are highly public devices containing specialized group information. To become an information appliance they need to share this information. The problem is how people move their personal artifacts (created on their PDAs) into the public domain (manipulated on the single display groupware) and back again. Our main goal is to understand how people distinguish between these personal and public artifacts, its consequences when designing CSCW tools and how to present context-based relevant information. Application design at a high level of abstraction eases to achieve these objectives.

Acknowledgements

This work was partially supported by the Chilean government funds MECESUP, under grant N° UCH019, and FONDECYT, under grant N° 1030959. We would like to thank the support of the Colombian program Colciencias.

References

1. Gaddah, A. and Kunz, T.: A survey of middleware paradigms for mobile computing, Technical Report SCE-03-16, Department of Systems and Computer Engineering, Carleton University, Ottawa, Canada, July 2003.
2. Chen, G. and Kotz, D.: A Survey of Context-Aware Mobile Computing Research. Technical Report Dartmouth Computer Science Technical Report TR2000-381, 2000.
3. Schilit, B. N., Adams, N. and Want, R.: Context-Aware Computing Application. Proc. of IEEE Workshop on Mobile Computing System and Application, IEEE Computer Society Press (1994) 85-90. December 1994.
4. Yan, H., Selker, T.: Context-aware office assistant. In: Proceedings of the 2000 Int. Conference on Intelligent User Interfaces, New Orleans. ACM Press. (2000) 276-279.
5. Persson, P.: Social Ubiquitous Computing. In CHI 2001 Building the Ubiquitous Computing User Experience, Position Paper.
6. Dey, A., Abowd, G., Salber, D.: A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. Human Computer Interaction Vol.16. Special Issue on Context-Aware Computing. (2001).
7. Ishii, H., Ullmer, B.: Tangible Bits: Towards seamless interfaces between people, bits and atoms. In Proceedings of the CHI '97 Conference on Human Factors in Computing Systems. New York, NY: ACM Press. (1997) 234-241.
8. Barton, J., Kindberg, T.: The Cooltown Experience. In CHI 2001 Building the Ubiquitous Computing User Experience, Position Paper.
9. Davies, N., Mitchell, K., Cheverest, K., Blair, G., Developing a Context Sensitive Tourist Guide. In: First Workshop on Human Computer Interaction with Mobile Devices.
10. Muñoz, M. A., Gonzalez, V. M., Rodríguez, M. and Favela, J.: Supporting Context-Aware Collaboration in a Hospital: An Ethnographic Informed Design. CRIWG 2003: 330-344.
11. Alarcón, R. and Fuller, D. A.: Intelligent Awareness in Support of Collaborative Virtual Work Groups. In Proc. of CRIWG'2002. Lecture Notes in Computer Science, Springer-Verlag, Berlin (2002) pp.168-188.

12. Brézillon, P. and Abu-Hakima, S.: Using knowledge in its context: Report on the IJCAI-93 Workshop. The AI Magazine, (1995) 16(1) pp. 87-91.
13. Leech, G.: Semantics: The Study of Meaning. Harmondsworth, UK: Penguin, (1981).
14. Sohlenkamp, M.: Supporting group awareness in Multi-User Environments through Perceptualization. GMD Research Series (1999) N°6 Zugl.: Padderborn, Univ. Diss.
15. Dourish, P. and Belloti, V.: Awareness and coordination in shared workspaces. In Proc. Of CSCW'92, (1992) pp.107-114.
16. Fussell, S., Kraut, R., Lerch, F., Scherlis, W., McNally, N. and Cadiz, J.: Coordination, overload and team performance: Effects of team communication strategies. In Proc. Of CSCW'98 (1998), pp. 275-284.
17. Alarcón, R. and Fuller, D. A.: Application Design Based on Work Ontology and an Agent Based Awareness Server. In Proc. of CRIWG'2003. Lecture Notes in Computer Science, Springer-Verlag, Berlin (2003) pp.: 314-329.
18. Burg, B.: Foundation for Intelligent Physical Agents. Official FIPA presentation, Lausanne, February 2002. See http://www.fipa.org for further details.
19. DeSanctis, G.: Shifting Foundations in Group Support System Research. In Group Support Systems – New Perspectives, L. Jessup, J. Valacich (eds.), MacMillan Pub. Co., New York, (1993), pp. 97-111.
20. Gruber, T. R.: Toward principles for the design of ontologies used for knowledge sharing. In Guarino, N. y Poli, R. (ed.), Formal Ontology in Conceptual Analysis and Knowledge Representation, Kluwer Academic Publishers, Deventer, The Netherlands.
21. JADE is the Java Agent DEvelopment Framework (http://jade.tilab.com/).
22. Neisser, U.: Cognition and Reality. Freeman, W. H. (ed), San Francisco (1976).
23. Norman, D. A.: The Invisible Computer. MIT Press (1998).