An approach to the integration of peer-to-peer systems with active environments

Paolo Busetta, Mattia Merzi
ITC-irst – Povo, Trento, Italy
E-mail: {busetta,merzi}@itc.it

***Abstract*— We adopt a form of group communication, called *channeled multicast*, for active rooms and other scenarios featuring strict real-time requirements, inherently unreliable communication, and a continuously changing set of context-aware autonomous systems. In our approach, rooted in multi-agent and team programming, coordination and cooperation are supported via "social awareness" and overhearing; agents form *implicit organizations* by actively participating to playing a role on a channel. We present an approach to the integration of Internet-based peer-to-peer systems into active rooms by means of agents on *proxy devices* accessing the users' *home peers*; a home peer is a remote or delocalized repository of data and services. In addition to other advantages, our approach introduces a new flavor to the concept of neighborhood in P2P systems, reduces or even avoids the need for users to physically carry P2P-equipped mobile devices with them, and easily supports virtual environments created by the composition of smart rooms at different locations.**

I. Introduction

We are working on the application of agents (by which we mean distributed, autonomous components), interacting by means of group communication and overhearing, to "active environments" [1]. Our research focuses on the issues related to the integration of components, and to the adaptation to continuous changes in the environment [2], [3]. These may vary from simple faults (e.g. power and network failures), to the continuously changing availability of different types of devices, including mobile devices carried by people entering or leaving a space. We adopt a *multi-agent* approach, where collaboration among agents is obtained via role-based, observable communication. This allows, from the one hand, to cater for continuous changes in the number of components, for the collection of contextual information and for unforeseeable interactions among different types of components; on the other hand, it requires proper coordination among components able to play the same role. To this end, agents form *implicit organizations*, i.e. sets of agents that negotiate a policy for playing a role. A multicasting technique is used, which eases the problem of finding interfaces and computing elements based on their capabilities and physical distribution; it also supports negotiation and coordination within implicit organizations.

Somewhere else [4], we have shown how we apply our approach to "active museums", i.e. interactive cultural information delivery within museums, which can be seen as large scale multi-user, multi-media, multi-modal systems featuring "ambient intelligence" [5]. This work explores a different application scenario and architecture. In particular, we focus on the integration of Internet-based peer-to-peer systems with smart rooms in order to provide a collaborative work environment. We introduce the concept of "home peer", i.e. a remote or delocalized system that contains data and services of a specific person or group. A home peer roughly corresponds to what is meant by "peer" in sophisticated architectures such as JXTA [6]. We show how agents running on *proxy devices* (called *proxy agents* for short) provide the proper way for a user to access and share the facilities available on her home peer while in a specific environment. This architectural choice – which contrasts with the common assumption that the user carries with her a peer on a portable device – offers a number of advantages, including security (both for the user's peer, and for the host environment), ease of management of the peers (backups as well as shared services do not have to be managed directly by users), simplified interoperability problems (proxy agents have to access a few well-known services offered by home peers, but proxies acting in different environments can be extremely different in their look-and-feel, in their use of sensors and actuators, and in their functionality), and possibly the freedom for a user *not* to carry any mobile device with her (at most, a smart badge for authentication if no alternative systems are available) if the smart room provides to create and destroy proxy agents when users enter and leave it.

This paper is organized as follows. Next section gives an overview of our approach to agent coordination. Its underlying group communication technique is examined in more detail in Sec. III. Sec. IV shows how we apply our approach to active environments. Sec. V focuses on the concept of home peer, and how it relates to agent-based active environments; a practical example is illustrated in Sec. VI.

II. A multi-agent approach to integration and coordination

We call "agent" any distributed, autonomous, communicating component [7]. Various researchers have looked at groups (or *societies*) of agents as first-order entities with goals and other attributes of their own, rather than being just an aggregation of the knowledge and behavior of their members; coordination protocols are used for maintaining a group's attributes synchronized with those of its individual members. Some research has focused on *teamwork*, initiated by the classic work on joint attitudes (goals, intentions, and mutual beliefs) [8] that has lead to the development of various theories and systems (e.g., [9]–[11]). Others have focused on

group planning (e.g., Partial Global Planning [12], Shared Plans [13]).

Our experimental agent communication infrastructure, called *LoudVoice*, is based on the concept of *channeled multicast* [14]. A channel is a stream of messages that can be listened to by many agents simultaneously, thus supporting *overhearing* [15]. LoudVoice uses the fast but inherently unreliable IP multicast. Channels in LoudVoice can be easily discovered by their *themes*, that is, by the main subjects of conversation; a theme is just a string taken from an application-specific taxonomy of subjects, accessible as an XML file via its URL. Having discovered one or more channels, an agent can freely "listen to" and "speak" on them by means of FIPA-like messages [16], encoded as XML documents. The header of a message includes a performative (i.e., a message type such as "REQUEST", "QUERY", "INFORM", "DONE", "ASSERT"), its sender and one or more destinations; the latter can be agent identifiers, but any other expression – including group identifiers – is accepted (for instance, we use role names; see Sec. III).

In simple terms, our approach consists in modeling the components of an active environment as agents, and having them communicating via LoudVoice rather than via point-to-point connections. As shown in next section, agents may include systems embedded into hardware – sensors, or contextual interpreters of sensors data ("context widgets" and "context aggregators" using the terminology of [17]), actuators, output devices –, middleware components close to sensors and actuators, e.g. context interpreters and context aggregators, and sophisticated application components (presentation generators, user profilers, and so on).

Following a common convention in multi-agent systems, we define a *role* as a communication-based API, or abstract agent interface (AAI), i.e. one or more protocols aimed at obtaining a cohesive set of functions from an agent. Differently from traditional approaches, our goal is to distribute functionality on a group of agents without necessarily know which agents are or will be delivering them, and without involving mediators, so as to achieve a higher flexibility and adaptivity to continuously changing conditions. Our approach to this is by favoring role-to-role rather than agent-to-agent interactions, as described later. Of course, this has a significant impact on the way agents behave in their interactions, also because we want to allow all possible combinations of roles and agents. For instance, we can have roles provided in full by a single agent; roles only partially fulfilled by a group of agents; and so on.

Our current work focuses on the first two of the following broad classes of relationship among roles and agents:

- redundancy: multiple agents support a certain role in an equivalent way. That is, a request for obtaining a service from that role can be serviced by anyone of the agents;
- partitioning: the services in the role are partitioned among the agents; that is, a service request can be fulfilled by at most one of the agents. Note that partitioning is often based not on the type of service being requested, but on its parameters; e.g., two agents may maintain user profiles with the same schema but for different users;
- coalition: the services of the role can be provided by the group but there is no single agent that can do it by its own; thus, collaboration is necessary. This is where team-oriented programming is traditionally applied; refer to the literature mentioned above.

The group of agents entirely or partially fulfilling a role may change over time, with the consequent impact on the ability of the system of delivering a specific service. Consequently, it is necessary for an agent to be aware of who else is able to play its same roles in the current context, and coordinate accordingly.

We adopt the term *organization* from Tidhar [18], to refer to teams where explicit *command*, *control*, and *communication* relationships (concerning team goals, team intentions, and team beliefs respectively) are established among sub-teams. We call *implicit organization* a set of agents tuned on the same channel to play the same role and willing to coordinate their actions. The word "implicit" highlights the facts that there is no group formation phase (joining an organization is just a matter of tuning on a channel), and no name for it – the role and the channel uniquely identify the group, indeed. As said above, our current research is focusing on implicit organizations formed by agents all able to play the same role but possibly in different ways, i.e. we address the partitioning and redundancies cases mentioned in the list above.

Goals for an implicit organization are automatically established by requests and queries addressed to its role. In Tidhar's terms, this is to say that a command relationship is established between any agent performing a goal-establishing communicative action (the "commanding agent") and the implicit organization, whose consequence is that the latter is committed at achieving the goal. Section III below discusses the corresponding protocol.

An implicit organization is in charge of choosing its own control policy, which defines: (1) how a sub-team is formed within the organization in order to achieve a specific goal; and, (2) how the intentions of this sub-team are established. In our current work, a goal-specific sub-team is fixed to be simply *all agents that are willing to commit immediately at achieving the organizational goal at the time this is established*; i.e., there is no explicit sub-team formation, rather introspection by each agent to decide whether or not it has enough resources immediately available. A *coordination policy* established for the organization is then used within a sub-team to decide who actually works toward achieving its goal, and possibly to coordinate the agents if more than one is involved.

We assume that policies are well-known to agents (some are described in Section III); our computational model (described in [3]) assumes the existence of a library of application-independent policies. Thus, it is possible to refer to a policy simply by its name. A policy, however, may have parameters that are negotiated by the organization before use – for instance, the currency used for auctions, or the master agent in a master-slave environment. We call *policy instance* a tuple composed of a policy name and the ground values for its parameters.

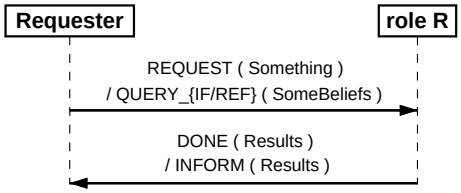

Fig. 1. A role-based interaction

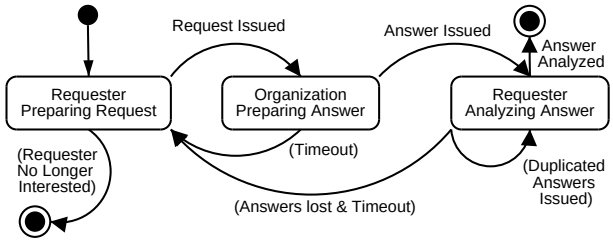

Fig. 2. Interacting with unreliable communication – a simplified protocol machine

Note that a goal-specific sub-team may well be empty, e.g. when all agents of the implicit organization are busy or simply no agent is part of the organization. With unreliable communication, this case is effectively indistinguishable from the loss of the goal-establishing message (unless overhearing is applied; this will be the objective of future works) or even from a very slow reaction; consequently, it must be properly managed by the commanding agent. These considerations have an important impact on the protocol between commanding agents and implicit organizations, as discussed in next section.

III. Role-based communication and implicit organizations

In principle, the interactions between commanding agents and implicit organizations are straightforward, and can be summarized in the simple UML sequence diagram of Fig. 1. A generic *Requester* agent addresses its request to a role *R* on the appropriate channel that it has discovered on LoudVoice; the corresponding implicit organization replies appropriately. However, unreliable channels, continuous changes in the number of agents in the organization, and strict real-time constraint, substantially complicate the picture. Fig. 2 is a finite state machine that captures, with some simplifications, the possible evolutions of the protocol. The events on top half represent the normal case - no message loss, a goal-specific sub-team achieves the goal. The events in brackets on the lower half represent degraded cases.

Consider, for instance, the cases where a request or its answer are lost, or no goal-specific sub-team can be formed. A timeout forces the commanding agent to reconsider whether its request is still relevant – e.g., the user's context has not changed – and, if so, whether to resend the original message. It follows that an implicit organization must be able to deal with message repetitions. This implies that agents should discard repeated messages or re-issue any answer already sent; also, the coordination policies should contain mechanisms that prevent confusion in the cases of partial deliveries or new agents joining the organization between two repetitions.

Similarly, the commanding agent has to deal with repeated answers, possibly caused by its own repeated requests. In the worst case, these repeated answers may even be different – e.g., because something has changed in the environment, a new agent has joined the organization, and so on. Rather than introducing middle-agents or making the organizational coordination protocols overly complicated to prevent this to happen, our current choice is to have the requester to consider as valid whatever answer it receives first, and ignore all others.

Not even to mention, the usability of the protocol presented above is limited to non-safety-critical applications, or at least to situations where it is possible to tolerate some level of uncertainty. This is definitely the case in our multi-media environments, for instance, where quality of communications is fairly high and the objective is nothing more critical than providing some guidance and context-sensitive cultural information to visitors of museums.

Observe that third parties overhearing a channel, in accordance to [15], may help in making the interaction with implicit organizations much more robust – for instance, by detecting some message losses, or whether a goal-specific sub-team has been established. We plan to explore some options in future work.

A high-level formalization of the coordination within an implicit organization needed to achieve a goal – that is, how the `Organization Preparing Answer` state of Fig. 2 is implemented – is provided in [2]. Two main phases are prescribed: first, a coordination policy is negotiated, if one has not already been established; second, in a policy-specific way, the goal is achieved. In turn, negotiation of the policy, also discussed in detail in [2] and [3], works in two phases: first, the set of *policy instances* common to all agents is computed; then, one is selected by an *oracle* agent, which may have been nominated from the outside or be heuristically chosen among the members of the implicit organization. We designed the policy negotiation protocol for unreliable multicast communication protocols.

To date, we have defined and described four basic, general-purpose organizational coordination policies that we use in our domain (described in [2]): Plain Competition (everybody races against everybody else), Simple Collaboration (everybody satisfies the goal, and one collects the results for all), Multicast Contract Net (a variant of the well known Contract Net), and Master-Slave (with an elected Master that decides for everybody). They all exploit, or work around limitations of unreliable multicast communication. Other policies may be easily added, possibly to satisfy application-specific requirements.

Plain Competition and Simple Collaboration are suitable to tasks which can be performed in parallel. In our smart virtual meeting room, described in Sec. VI, both policies are applicable to the case of distributed search. Multicast Contract Net and Master-Slave can be applied pretty much everywhere, but they are best suited to situations where it is wasteful or harmful to have multiple agents working in parallel. In

our museum environment, we apply Multicast Contract Net to so-called *presentation planners* creating context-dependent multimedia presentations. In a virtual meeting room, the same policy can be used to select the screen where to show the GUIs of the agents; the screens are under control of the so-called *smart browsers* (Sec. VI), which render the output of an application after bidding against each other with values depending on the quality of the output devices they control and the current user position.

IV. Agents and active environments

Generally speaking, active environments have some characteristics that make them substantially different from traditional computing and HCIs:

- Multiple users may be in a single place, interacting with different applications simultaneously. Context is shared among applications simultaneously active for the same user (typically, one interacting with the user, the others in background overhearing and interacting only when appropriate, for instance to provide additional information or when explicitly or implicitly invoked).
- The set of users changes dynamically over time. For example, people may enter or exit a smart room as they please.
- Users are unaware that the environment is formed by many components. They are also not interested in the internal mechanisms used to provide services. Therefore, they interact with the environment as if it were a single, monolithic system.
- However, services are provided by a variable set of components that can join and leave the environment (plug-and-play as well as mobile computing), and can be running anywhere (from a local computer to anywhere on the Internet).
- Services provided by components can (partially) overlap; therefore these components need to coordinate in order to decide, for instance, who provides a specific service, and how.

Active environments may differ for many reasons, the most obvious being the type and quantity of sensors and actuators being used. Less obvious but not less important is the functionality provided by each environment, which changes the users expectations and their behavior.

Our first example is PEACH [19] (`http://peach.itc.it`), which aims at creating "active museums", i.e. smart environments featuring multimodal I/O where a visitor would be able to interact with her environment via various sensors and effectors, screens, handheld devices, etc. This application domain provides a very challenging environment: agents can come and go dynamically, visitors move about the museum (potentially carrying handheld devices), communication media are heterogeneous and unreliable (a mix of wired and wireless networks like WiFi or Bluetooth are likely). An important consideration is that, if the system does not react timely to the movements and interests of the visitors, they will simply ignore it or, worse, will become annoyed.

By contrast, in this work we explore a *collaborative workspace* scenario, such as a *smart virtual meeting room*. This kind of room is virtual in the sense that it may connect more than one physical space or even isolated users and it is possible for a physical space or user to be part of more than one virtual room simultaneously. The major differences between this type of smart room and an active museum are two. First, the stress moves from designing a system driving a relatively passive visitor and providing information with relatively little or implicit user input, to supporting a group of active participants that need to perform tasks in collaboration, thus providing much and varied inputs (from speech and text to drawings, documents and actions on a shared workspace) while the room obeys to orders rather than drive the user. Second, while museum visitors are not willing to wait for a slow system to react, a participant to a smart room wants the system to help and expects to have to wait for certain operations to be performed (such as accessing information on the Internet or doing a complex computation). We anticipate that a smart virtual meeting room runs many more complex applications – but it is also more stable – than an active museum, since people are focused on one or a few specific goals, stay for longer periods and do more sophisticated tasks. Before describing our rooms, we need to introduce the concept of *home peer*.

V. From "take your peer along with you" to "access your peer from anywhere"

It is commonly expected, both in industry and research, that peer-to-peer systems and active environments will somehow merge; in this perspective, for instance, JXTA [6] has special extensions to support very small devices[1]. This expectation is justified by two main assumptions: (1) portable devices are becoming smaller and more powerful by the day, thus they will contain more and more data and useful processing capabilities (to the point of representing the entire electronic personality of their owner) and will become the privileged access interface to any form of virtual world, including the Internet and smart rooms; (2) peer-to-peer, as a paradigm, is much more suitable to real world communication needs than a centralized, client-server approach to the management of personal data and services. Moreover, a peer-to-peer approach to networking, based on discovery of neighbors and with no reliance on a central service of any kind, is commonly adopted to create sensor networks and to connect device components, for instance headsets and controls of mobile phones and home theater equipment, so it is already at the foundations of forms of active rooms. The long-term vision is to enable something similar to the iRoom [20]: heterogeneous portable devices in the same room spontaneously form a peer-to-peer network to exchange data, to enable collaborative work, and to exploit whatever facility is available locally (wall screens, microphones, and so on).

We think that this vision has to be tempered with some realistic considerations. Personal devices can be lost, stolen, destroyed, and are hard to backup (even if many types of synchronization software exists) – it would be unwise for anybody to trust all their data on one of them. Personal devices

[1] JXTA for J2ME, `http://jxme.jxta.org`.

are incompatible with each other – it will still take a long time before a universal data format emerges, not to mention lock-in market strategies, so migrating from one device to another is a very unpleasant experience. Full discovery and adaptation to local conditions (e.g., sensors, actuators, applications, but also protocols and software versions) is still a long term goal. Privacy, authentication, and authorization of "foreign" mobile devices entering an active room, in particular when wireless communication is in place, are complex issues, and a universal approach is not likely to emerge for a while. Finally, the assumption of "one user == one device" is flawed – a person has different interests and goals (touristic rather than professional, for instance), and often she represents or needs to access data and services of her group (company, co-workers, social club, whatever) rather than her personal stuff: so, how many devices should she carry – one for each role she can play? or, how does she partition data on a single one that she carries everywhere? how does she keep the data continuously aligned with others of the same group?

We propose an alternative approach, which tackles some of those issues and does a better use of the increasing availability of fast Internet access and, in future, of grids of data and computers. We assume that access to the Internet is always possible from an active room, possibly through security gateways (e.g. via so-called *rendez-vous* nodes in JXTA, which allow controlled, HTTP-based tunneling from within an intranet to the external world). In short, rather than forcing the user to carry her own device, we envisage that the user is given one, called *proxy device*, when entering a smart environment (such as an office building of a corporation), to be used to access the local facilities.

A proxy device can be any kind of portable digital device that supports wireless communication and a location sensor (a palm top computer, a smart phone, even a system embedded in a digital wristwatch, the choice being dictated by application considerations); as a matter of fact, the proxy device could even substitute identification badges currently carried by visitors and employees. The right authorizations for the proxy device are easily set up, and the device can be trusted by the host network. Agents on the proxy device (*proxy agents* for short) perform their work using the data and services on the user's *home peer*. The latter is a system – such as a JXTA peer – on which the user has configured a set of services and data for access from a remote place. The home peer may be hosted anywhere on the Internet, or even be completely delocalized on a computing grid.[2]

The home peer is more than a repository of personal data – for instance, it may be participating to one or more peer-to-peer applications, such as distributed knowledge management systems like KEx [21], that do not require a continuous user supervision to operate and may be running 24 hours by 7 days per week. A user may have more than one home peer, possibly shared with somebody else (e.g., a department's peer containing all papers produced by its researchers). The user gives a home peer address and her credential (which may have an expiration date, limitations on the data that can be distributed, and any other techniques to prevent passwords to be stolen, privacy intrusions, etc.) to her proxy agents. Compared to the complexity of individual and group interactions that local and proxy agents perform within an active environment, the set of services provided to the proxy agents by a home peer is relatively limited and relatively easy to standardize ("get" and "put" documents, "read" and "modify" an agenda, "perform" some operations on the underlying peer-to-peer network, and so on). Ultimately, a home peer could appear to the proxy agents as a container of user-owned Web services defined with WSDL[3] and exchanging XML documents with SOAP[4], so providing a standardized and uniform access to all its capabilities.

In the case study presented later, proxy agents, in addition to acting as user assistants for accessing the home peers, can create a trusted sub-network (implemented, e.g., as a JXTA group) among these peers; this allows typical peer-to-peer services (such as file exchanges, distributed searches, and other application-specific operations) to be performed among all and only the peers of the users of the same active environment. This technique effectively provides a variant of the concept of "neighborhood" in P2P networks – rather than a matter of network configuration or physical proximity among peers (such as being on the same LAN or in the range of a wireless network like Bluetooth), in our approach two home peers are neighbors as long as their owners are part of the same *virtual space*.

A final observation is that a proxy device may be not necessary, in theory – in an environment fully equipped with localization sensors and multimedia, multimodal I/O components, proxy agents could be created on a server anywhere and interact with their human owners by means of whatever device is available locally, rather than being carried around by users. In addition to technological issues that make this scenario difficult to realize in the short term, other considerations (such as privacy, convenience of use of portable interfaces, the need to keep a virtual presence even if the user is temporarily away from a room) may lead to preferring proxy devices anyway.

VI. Smart virtual meeting rooms

Our case study (under development at the time of writing) can be seen as an example of Computer Supported Cooperative Work (CSCW). Our *smart virtual meeting rooms* (VMRs for short) differ from real meeting rooms because they do not necessarily correspond to a space delimited by concrete walls. A VMR consists of a cooperating set of tools and their controlling agents, even if their users may be physically spread over two or more locations. A *virtual room janitor* agent, in charge of managing one VMR, has the goals of: 1) creating one or more LoudVoice channels dedicated to the communication among the participating agents; 2) determining who is part of its VMR with some criteria specified by its creator (for instance, the people physically present in one or more rooms, or nominated users); and, 3) inviting the

[2]Of course, nothing prevents that the home peer is on a portable system that the user carries with her, and that becomes part of the host network via DHCP (Dynamic Host Configuration Protocol, `http://www.dhcp.org/`).

[3]Web Services Definition Language, `http://www.w3.org/TR/wsdl`.

[4]Simple Object Access Protocol, `http://www.w3.org/TR/SOAP/`.

users' *virtual room participant* agents by sending a message on the appropriate LoudVoice channel. There is one instance of participant agent per user, typically running on portable devices (or "personal digital assistants", PDAs for short). A participant starts the required application agents after receiving an invitation by a janitor, shut them down when its user leaves the VMR, and handles the case of participation to multiple VMRs[5].

In its basic instance, a VMR may remind of existing systems such as the iRoom [20]: a room equipped with wall screens, sensors of various kind (including speech and gesture recognition systems), and wireless networks used by the PDAs running CSCW tools. The latter include a simple shared textual blackboard and a shared graphical space[6], whose output can be seen both on PDAs and on wall screens, possibly in different formats and sizes. These shared tools are implemented (or controlled) by agents, running either on the PDAs or in back-room computers, whose goals include negotiating who controls the input ("floor control" in CSCW terms) and who keeps the master copy of the shared objects; they form implicit organizations whose coordination policy is typically either Multicast Contract Net or Master-Slave (Sec. III).

As said above, a VMR may include multiple rooms or nominated users independently of the latters' locations. In addition to the tools already mentioned, distributed participants communicate by means of standard technologies, such as phone or video conference systems. A simple "business card keeper" agent exchanges user information with its peers on the LoudVoice channel dedicated to its VMR, and shows to its user who are the other participants, who enters and who leaves, and how to talk with them if in a different room.

The ability of a virtual room of encompassing multiple physical spaces, and the possibility for a user of participating to multiple rooms simultaneously distinguish our approach from what is commonly taken by middleware for active spaces (see for instance Gaia [23]), where there is no distinction between physical and logical proximity.

The home peer concept described in the previous section will be tested by developing a few other tools, implemented as proxy agents of the peer-to-peer KEx search engine [21]. For instance, *citation finder* agents – one per user – will handle different types of distributed searches on their users' KEx repositories of documents and so-called "contexts". Searches will be performed on behalf of any user or agent (including those controlling the CSCW tools) querying the entire VMR. A Plain Competition policy (Sec. III) will be used to coordinate the organizations of citation finders when implementing a search type such as "first matching document/context"; Simple Collaboration is better suited for searches such as "all matching documents/contexts". A citation finder can also work standalone for its user and delegate its KEx peer to search in the home peers of the other users involved in the same VMR (i.e., the other VMR participants becomes the so-called "targets" of a KEx query, thus exploiting the so-called "context matching" capability). Another example is *temporary federations*, created for all users participating to a VMR by *federating agents*. A KEx federation is a set of peers sharing a common context (i.e. a semantic index), which can be used to share documents among all participants. The federating agents interact among themselves and with their users in order to "map" the KEx peers' contexts (i.e. identifying their semantic relationships), to build the temporary federation context, and to allow users to navigate the resulting aggregate repository of the federated peers and to exchange documents.

Finally, we are defining a *strategy game* to be played by teams of human and artificial agents; goals of this game are to show the advantage of participating to multiple VMRs simultaneously, and to research on team intention recognition via overhearing. Indeed, each team will coordinate in its own private VMR, while the game is played in a VMR shared by all teams. We plan to develop artificial players, that will decide their own strategies in coordination with their team and against the others.

For testing, analysis and demonstration, we are simulating our VMRs by means of a multi-user 3D navigator, similar to common multi-user games [24]. A user connects to its graphic engine to navigate in the virtual rooms; in turns, users are represented as avatars, so participants can "see" each other. We developed a special purpose client of the 3D engine that can be extended to simulate any type of sensor; currently, we are simulating two types of localization systems for the PDAs carried by users with them during a navigation[7]. The first determines a PDA x, y coordinates on a plane[8]. The second senses the PDA direction (North, East, South, West, or midway two cardinal points)[9]. This second type of sensors is present only in part of the environment, so there are areas where only the PDA location is known but not its direction; this limitation has to be handled by our *direction-giving* agents, whose goal is to help a user reaching a given room.

Sensor agents send their changes of state as events on a dedicated LoudVoice channel. These events are elaborated by a set of *user tracker* agents, which fuse the different sensor data (possibly compensating for missing or inconsistent information), keep the current user positions and a history of their movements, and are able to convert the current coordinates into a variety of reference systems, such as rooms, logical grids, distances from known objects – adding or removing an application-specific reference system is just a matter of starting or stopping a specialized user tracker. The organization of user trackers, whose default coordination policy is Plain Competition (Sec. III), notifies location events and can answer

[5]In the initial version, this will be done in collaboration with the window manager of the PDA (running Linux), which creates a different virtual desktop for each VMR, on which the participant agent starts an instance of each application agent type.

[6]We are designing a multi-user version of a Web design tool called Denim [22]

[7]The PDA themselves can either be simulated, by means of a simulator running on the same system used for navigation, or be real; in this case, they do not need to have on board any sensor.

[8]In a closed environment, this is typically based on the triangulation of WiFi signals. One is being developed in IRST in the scope of the Wilma project, `http://www.wilmaproject.org/`. Other location sensors, based on ultrasounds and image processing, are being explored within the scope of PEACH.

[9]In PEACH, this sensor is implemented as a triangulation system that intercepts signals from infrared beamers spread around the environment.

to queries on another channel dedicated to user locations.

Location information within a VMR will be exploited to enable the automatic switch of the main user interfaces to the "best" system available in a given context. Whenever a PDA is left for some time very close to and facing a personal computer – thus in a position that makes using the PC instead of the PDA convenient for its user – the GUIs of the agents participating to this VMR "migrate" to the PC. Conversely, when the PDA is moved away, the same GUIs move back to the latter. This functionality is managed by *smart browser* agents, running on all involved devices, which follow the user movements and negotiate (typically with a Multicast Contract Net policy) who "owns" the main I/O device of a specific VMR for a specific user[10].

VII. Conclusions and Future Works

We proposed *implicit organizations* for the coordination of agents able to play the same role, possibly in different ways, exploiting group communication and overhearing in environments where messages may be occasionally lost and agents can come and go very quickly. Some preliminaries performance results can be found in [26].

We are currently focusing on practical experimentation and application to different domains. The first is multi-media, multi-modal cultural information delivery in active museums. The second are smart offices; we have presented the smart virtual rooms on which we are currently working.

We have introduced the concept of home peer to represent a permanent, always on-line repository of documents and services for a user or a group. We argued that, rather than "carrying" a peer with them in smart environments, users should access their peer by means of application-specific, context-sensitive proxy agents.

Acknowledgments

This work has been supported by the PEACH and TICCA projects, funded by the Autonomous Province of Trento. We thank Silvia Rossi and Gaetano Calabrese for their help in defining the smart virtual meeting room case study.

References

[1] J. McCarthy, "Active environments: Sensing and responding to groups of people," *Personal and Ubiquitous Computing*, vol. 5, no. 1, 2001, available at http://www.inf.ethz.ch/vs/events/dag2001/.

[2] P. Busetta, M. Merzi, S. Rossi, and F. Legras, "Intra-Role Coordination Using Group Communication: A Preliminary Report," in *Proceedings of the second workshop on ACL and conversation policies at the Second international joint conference on Autonomous agents and multiagent systems (AAMAS 2003)*. ACM Press, 2003.

[3] ——, "Real-time Role Coordination For Ambient Intelligence," in *Proceedings of the workshop on Representations and Approaches for time-critical decentralized Resource/Role/Task allocation at the Second international joint conference on Autonomous agents and multiagent systems (AAMAS 2003)*. ACM Press, 2003.

[10] In our initial implementation, smart browsers control multiple VNC [25] clients each, which show the output of X11 servers detached from a real screen. The applications using these servers are totally unaware of which physical device is being used at any time. In future, we plan to explore a more sophisticated technique, which goal is to allow application agents to be involved in what is going on and possibly adapt as necessary.

[4] P. Busetta, M. Merzi, S. Rossi, and M. Zancanaro, "Group communication for real-time role coordination and ambient intelligence," 2003, submitted to UbiComp 2003.

[5] K. Ducatel, M. Bogdanowicz, F. Scapolo, J. Leijten, and J.-C. Burgelman, "Scenarios for ambient intelligence in 2010," Information Society Technologies Programme of the European Union Commission (IST), Tech. Rep., Feb. 2001, http://www.cordis.lu/ist/.

[6] "Project JXTA," http://www.jxta.org/.

[7] M. J. Wooldridge, "Intelligent agents: Theory and practice," *The Knowledge Engineering Review*, vol. 10, no. 2, pp. 115–152, February 1997.

[8] P. R. Cohen and H. J. Levesque, "Teamwork," AI Center, SRI International, Menlo Park, CA, Tech. Rep. 504, 1991. [Online]. Available: citeseer.nj.nec.com/article/cohen91teamwork.html

[9] I. Smith, P. Cohen, J. Bradshaw, M. Greaves, and H. Holmback, "Designing conversation policies using joint intention theory," in *Proceedings of Third International Conference on Multi-Agent Systems (ICMAS98)*, 1998.

[10] S. Kumar, M. J. Huber, and P. R. Cohen, "Representing and executing protocols as joint actions," in *Proceedings of the first international joint conference on Autonomous agents and multiagent systems*. ACM Press, 2002, pp. 543–550.

[11] M. Tambe, "Towards flexible teamwork," *Journal of Artificial Intelligence Research*, vol. 7, pp. 83–124, 1997.

[12] E. H. Durfee and V. R. Lesser, "Negotiating task decomposition and allocation using partial global planning," in *Distributed Artificial Intelligence, Volume II*, L. Gasser and M. N. Huhns, Eds. Morgan Kaufmann Publishers, 1989, ch. 3, pp. 229–244.

[13] B. Grosz, L. Hunsberger, and S. Kraus, "Planning and acting together," *AI Magazine*, pp. 23–33, Winter 1999.

[14] P. Busetta, A. Doná, and M. Nori, "Channeled multicast for group communications," in *Proceedings of the first international joint conference on Autonomous agents and multiagent systems*. ACM Press, 2002, pp. 1280–1287.

[15] P. Busetta, L. Serafini, D. Singh, and F. Zini, "Extending Multi-Agent Cooperation by Overhearing," in *Proceedings of the Sixth Int. Conf. on Cooperative Information Systems (CoopIS 2001)*, Trento, Italy, 2001.

[16] Foundation for Intelligent Physical Agents, "FIPA Communicative Act Library Specification," http://www.fipa.org/repository/cas.html.

[17] A. K. Dey, "Providing architectural support for building context-aware applications," Ph.D. dissertation, College of Computing, Georgia Institute of Technology, Georgia, USA, 2000.

[18] G. Tidhar, "Organization-oriented systems: Theory and practice," Ph.D. dissertation, Dep. of Computer Science and Software Engineering, The University of Melbourne, Australia, 1999.

[19] O. Stock and M. Zancanaro, "Intelligent Interactive Information Presentation for Cultural Tourism," in *Proc. of the International CLASS Workshop on Natural Intelligent and Effective Interaction in Multimodal Dialogue Systems*, Copenhagen, Denmark, 28-29 June 2002.

[20] B. Johanson, A. Fox, and T. Winograd, "The interactive workspaces project: Experiences with ubiquitous computing rooms," *IEEE Pervasive Computing Magazine*, vol. 1, no. 2, April-June 2002.

[21] M. Bonifacio, P. Bouquet, G. Mameli, and M. Nori, "Peer - Mediated Distributed Knowledge Management," in *Proceedings of AAAI Spring Symposium on Agent Mediated Knowledge Management (AMKM'03)*, Stanford University, Stanford CA, USA, 2003.

[22] J. Lin, M. W. Newman, J. I. Hong, and J. A. Landay, "Denim: An informal tool for early stage web site design." in *Extended Abstracts of Human Factors in Computing Systems: CHI 2001, Seattle, WA*, March 31-April 5 2001, pp. 205–206.

[23] M. Roman, C. Hess, R. Cerqueira, A. Ranganathan, R. Campbell, and K. Nahrstedt, "A middleware infrastructure for active spaces," *IEEE Pervasive Computing Magazine*, vol. 1, no. 4, October-December 2002.

[24] F. Bertamini, R. Brunelli, O. Lanz, A. Roat, A. Santuari, F. Tobia, and Q. Xu, "Olympus: an ambient intelligence architecture on the verge of reality," in *Proceedings of ICIAP 2003 – 12th International Conference on Image Analysis and Processing*, Mantova, Italy, September 2003, (To be published).

[25] T. Richardson, Q. Stafford-Fraser, K. R. Wood, and A. Hopper, "Virtual network computing," *IEEE Internet Computing*, vol. 2, no. 1, pp. 33–38, Jan/Feb 1998.

[26] P. Busetta, M. Merzi, S. Rossi, and F. Legras, "Intra-role coordination using channeled multicast," ITC-IRST, Trento, Italy, Tech. Rep. 0303-02, 2003.