

Group Communication for Real-time Role Coordination and Ambient Intelligence

Paolo Busetta¹, Mattia Merzi¹, Silvia Rossi^{2,3}, and Massimo Zancanaro¹

¹ ITC-irst – Povo, Trento, Italy

{busetta,merzi,zancana}@itc.it

² Università di Trento – Povo, Trento, Italy

³ Istituto di Cibernetica, CNR – Pozzuoli, Napoli - Italy

silvia.rossi@dit.unitn.it

Abstract. We propose 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. Overhearing also allows the collection of contextual information without interfering with running systems. We introduce the concept of *implicit organization* for coordinating agents, outline a general architecture, describe some of the protocols in use in our applications (interactive museums), and report on some initial experimental results.

1 Introduction

In the context of the PEACH project,⁴ we are investigating the concept of “active museums”, i.e. interactive cultural information delivery within museums or archaeological sites. An active museum is a form of “active environment” [12], and can be seen as a large scale multi-user, multi-media, multi-modal system featuring “ambient intelligence” [6].⁵ Although many research projects are exploring the possibilities offered by Personal Digital Assistants as multimedia guides (among others, [13, 4, 9, 17]), there have been very few attempts to investigate the architectural issues emerging from the embedding of a mobile guide into an overall “intelligent museum”.

In a “intelligent museum” different devices, mobile or fixed, have to interact with an array of services. Not only the number of devices in the museum can’t be easily predicted, but also the available services may change over time. Moreover, it is desirable that services are provided in a user-adapted and context-adapted way. A layered architecture may not be able to support flexibility up to this degree.

In this paper, we adopt a *multi-agent* approach, where collaboration among agents (that is, distributed, autonomous components) is obtained via role-based, observable communication. This allows, from the one hand, to cater for continuous changes in the

⁴ <http://peach.itc.it>.

⁵ This work was supported by the PEACH and TICCA projects, funded by the Autonomous Province of Trento.

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, we introduce *implicit organizations*, which are 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, and supports negotiation and coordination within implicit organizations.

2 A multi-agent approach to integration and coordination

We call “agent” any distributed, autonomous, communicating component [16]. 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) [5] that has led to the development of various theories and systems (e.g., [14, 11, 15]). Others have focused on group planning (e.g., Partial Global Planning [7], Shared Plans [10]).

Our experimental agent communication infrastructure, called *LoudVoice*, is based on the concept of *channeled multicast* [1]. A channel is a stream of messages that can be listened to by many agents simultaneously, thus supporting *overhearing* [3]. 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 [8], 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. 4).

In simple terms, our approach consists in modelling the components of an active environment as agents, and having them communicating via LoudVoice rather than via point-to-point connections. 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 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.

In this work, we focus two broad classes of relationship: **redundancy**, where multiple agents support a certain role in an equivalent way; and, **partitioning**, where the services in the role are partitioned among the agents, i.e. a service request can be fulfilled by at most one of the agents. We don’t deal here with another interesting class of

relationships, **coalition**, where the services of the role can be provided by the group but there is no single agent that can do it by its own.

3 Agents and active environments

A typical problem we are facing in PEACH is the following. A visitor is supposed to receive a multimedia presentation on a particular subject, but (1) several agents are able to produce it with different capabilities (pictures + text or audio + video) and variable availabilities (CPU load or communication possibilities) and (2) the visitor is close to several actuators (screens, speakers) each able to “display” the presentation, and the visitor carries a PDA which is also able to display it, albeit in a more limited way. In addition, several other visitors are nearby potentially using the actuators, and of course if the visitor does not get a presentation after a few seconds she will leave.

At least five main roles can be identified in our museum scenario:

- the **user interfaces on mobile devices**: agents that play this role can: (i) sense infrared codes and communicate them to the rest of the system; (ii) play presentations (i.e., play audio files and display synchronized images), (iii) communicate to the rest of the system notable events concerning the presentations being played, e.g. start, end, stop or pause requested by the user;
- the **Physical Organization Knowledge Base**, able to map sensor codes to physical positions, and to provide a number of related reasoning services (e.g. retrieve the set of closest frescoes, etc.);
- the **User Modeler** that deals with storing, retrieving and reasoning on information related to the user;
- the **Input Analyzer**, which takes care of interpreting user movements in terms of inputs to the system;
- the **Presentation Planner** that assure the composition of a coherent presentation on a given artwork, tailored for a given visitor.

It is easy to see how the system would benefit in terms of efficiency and robustness by using a multi-agent, redundant approach to roles. For example, the Presentation Planner role could be played by two agents: a complex adaptive planner that composes tailored presentations, and a database that indexes “canned” presentation on the artworks. The coordination policy between the two could be implemented as a fixed scheduling as the following: if the complex planner is not able to build a presentation, the simpler one replaces it.

4 Coordinating an Organization

We propose a special form of team, called *implicit organization* [2], for the coordination of agents joining multicast channels to play a specific role. The organization is “implicit” in the sense that it does not need any explicit formation phase; any agent playing a certain role and listening messages directed to that role is automatically involved in the organization. A high-level formalization of the coordination within an

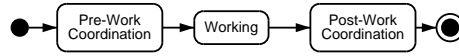


Fig. 1. Abstract coordination policy, UML state diagram

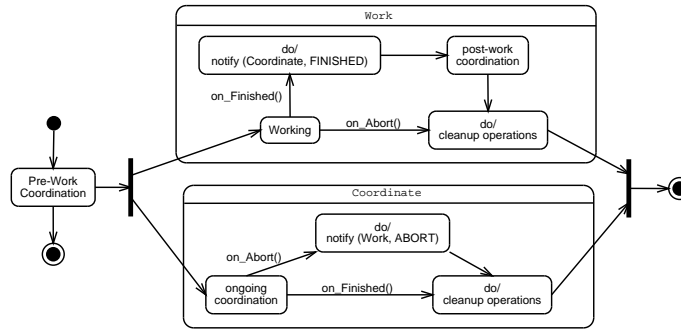


Fig. 2. Abstract behavior of a member of a goal-specific sub-team – UML state diagram

implicit organization needed to achieve a goal is provided in [2]. It prescribes two main steps: first, negotiating a coordination policy, if one has not been already established; second, in a policy-specific way, achieving the goal. In this section we focus on the process of coordination among the agents while the sub-process of selecting a policy will be discussed in the next section.

When an implicit organization receives a requests, its members with available resources form a sub-team. This happens silently, i.e. there is no explicit group formation message. Any coordination policy follows a straightforward three-phase schema (Fig.1). Before doing anything, the sub-team coordinates to form a joint intention on how to achieve the goal (Pre-Work Coordination). The agents in charge perform whatever action is required, including any necessary on-going coordination (Working). When finally everybody finishes, a Post-Work Coordination phase collects results and replies to the requester.

The abstract schema above is summarized by Figure 2 from the perspective of a single member of the sub-team. The Working phase of the abstract coordination policy here is expanded in two concurrent, cooperating state machines (Work and Coordinate), one performing actions toward the goal, the other coordinating with the other members of the sub-team. Observe that, after Pre-Work Coordination, an agent may immediately stop working; this is typically the case when it is no longer involved in achieving the goal.

So far, we have defined four basic organizational coordination policies for our domain. Variants and alternatives can be easily designed to meet different requirements, such as Quality of Service objectives, efficiency, and so on. Practical examples of their application will be shown later, in Sec. 6. We give here a quick outline of those policies; more details can be found in [2]. Since we use unreliable communication channels, the

corresponding protocols have been adjusted to take in account the chance of message loss and repetitions.

Our first basic policy is **Plain Competition**. It can be summarized as “everybody races against everybody else, and the first to finish wins”. It is by far the easiest policy of all: no pre-work nor post-work coordination is required, while the on-going coordination consists in overhearing the reply sent by who finishes first. The **Simple Collaboration** policy consists of a collaboration among all participants for synthesizing a common answer from the results obtained by each agent independently. This policy does not require any pre-work coordination; the on-going coordination consists in declaring what results have been achieved, or that work is still on-going; finally, the post-work coordination consists in having one agent (the first to finish) collecting answers from everybody else and sending the answer to the requester. Finally, we have designed a **Multicast Contract Net** policy, i.e. a variant of the well-known Contract Net Protocol, and a quite straightforward **Master-Slave** policy.

5 Negotiating a policy

An implicit organization establishes its own coordination policy by means of negotiation. In the following, we give an overview of a protocol designed for unreliable communication; see [2] for a formalization and the algorithm applied by each agent.

In summary, the negotiation protocol works in two phases: first, the set of *policy instances* common to all agents is computed; then, one is selected. A policy instance is a tuple with the name of a policy and ground values for all its parameters; for instance, $\langle \text{auction}, \text{Euro} \rangle$ represents an “auction” policy whose first parameter (presumably the currency) is “Euro”. In [2], we describe a language for expressing sets of policy instances in a compact way by means of constraints on parameters, including infinite or undetermined ranges. During the first phase of the negotiation, a group belief about the common policy instances is established. This can be easily achieved by having each agent sending information messages (INFORM) to the role on what it is able to support, and intersecting the contents of all received messages. The second phase consists in having one agent – called the *oracle* – selecting one policy among those common to everybody and notifying the organization of its decision, by means of a simple assertion message (ASSERT) sent to the role.

The oracle can be any agent, either a member or external to the organization (in the latter case, recall that it can overhear all messages sent on a channel, thus it can be informed on the common policies). It can apply whatever decision criteria it deems more appropriate – from a random choice, to a configuration rule, to inferring on previous policies based on machine learning, and so on. In this work, we do not elaborate on how the oracle is chosen, nor on its logic (but we will give some examples in Sec. 6). Our algorithm [2]: (1) allows for any external agent (such as a network monitor or an application agent interested in enforcing certain policies) to intervene just after the common policies have been established; (2) provides a default oracle election mechanism if an external one is not present; and, (3) handles conflicting oracles by forcing a re-negotiation.

To work in an unreliable environment, we added some special information and messages to our protocol. For instance, all messages contain a *Negotiation Sequence Number* (NSN), i.e. a unique identifier of a negotiation process that guarantees coherence of protocols and integrity of beliefs in the cases of message losses, network partitioning, and new agents joining mid-way a negotiation. A *policy reminder* message, consisting of an INFORM on what is believed to be the current policy, is periodically sent by each agent after the end of the negotiation, allowing recovery from the loss of the oracle announcement and consistency checking against other problems.

6 Practical examples of implicit organizations

Our first example is a group of *competing presentation planners*. As shown in Sec. 3, PresentationPlanners (*PP* for short) generate multi-media presentations on specific topics relevant to the context where a user, or a group of users, is. For instance, a user getting close to an art object should receive – depending on its interests, profile, previous presentations she received, etc. – a personalized presentation of the object itself, of its author, possibly of related objects in the same environment or other rooms. A typical interaction would look like the following (described using a simplified FIPA-like language):

```
REQUEST                                     DONE
From: VisitorAssistant033                  To: VisitorAssistant033
Receiver: PresentationPlanner              Content: done (
Content: preparePresentationForUser(621)   preparePresentationForUser(621),
                                           results (
                                           file(http://srv/pres1377.ram),
                                           bestResolution ( 800, 600 ),
                                           includeVideo ( true ),
                                           includeAudio ( true ) ) )
```

Typically, a *PP* works on a knowledge base containing text, audio and video tracks. When a request arrives, the *PP* collects data on the user and her contexts, e.g. by querying roles as *UserProfiler*, *RoomLayout*. Then, it queries its own knowledge base and, if information is available, it builds a multimedia presentation, by connecting audio and video tracks, generating audio via text-to-speech systems, and so on.

A *PP* is often the leader of its own team, formed by highly specialized agents. By contrast, it is unlikely that different *PPs* collaborate – sensibly merging multi-media presentations is a hard task even for a human. Observe that, in realistic situations, redundancy of *PPs* is a necessity, e.g. to handle the workload imposed by hundredths of simultaneous visitors. Redundancy can be obtained in various ways, for instance by putting identical *PPs* working on the same knowledge bases, or by specializing them by objects, or by rooms, or by user profiles.

Given the variety of possible configuration choices, the best policies for a *PP* are *Plain Competition* and *Multicast Contract Net* based on some quality parameter; it may also be that, in well controlled situations, a *Master* can be elected (or, more likely,

imposed by an oracle). Thus, the developer of a *PP* should enable a number of non-collaborative policies, which means specifying criteria for bidding, using negated enumerations to accept a Master-Slave policy but excluding its own *PP* from becoming a master, and so on.

Our second example of the use of implicit organizations consists in *choosing among multiple screens*. Active museums may have multiple places where to show things to visitors, e.g. large screens on different walls and the users' own PDAs. Location, but also quality and logical congruence with the rest of the visit, are all important choice factors. Also, not necessarily *one* screen is a good choice – for instance, a presentation to a group of people may be better shown on multiple places simultaneously.

We are working on SmartBrowsers (*SB* for short). A *SB* is an agent able to show a multi-media presentation (video and audio) and aware of its position (which may be static, if its display is a wall screen, or mobile, if on board of a PDA). A typical interaction looks like the following:

```
REQUEST                                DONE
From: VisitorAssistant033              To: VisitorAssistant033
Receiver: SmartBrowser                 Content:
Content:                                done (
  showMultiMedia (                      showMultiMedia (),
    user (621),                          results ( completed ) )
    file (http://server/pres1377.ram),
    bestResolution ( 800, 600 ),
    includeVideo ( true ),
    includeAudio ( true ) )
```

SBs should accept a policy that allows a clear selection of one, or (better) a fixed number of agents at request time. Thus, *Plain Competition* and *Simple Collaboration* should be avoided; *Master-Slave* works, but seems unduly restrictive in a situation where *SBs* are context aware. We are working on a *Multicast Contract Net* policy where bids are computed as a single number combining screen resolution, distance from the user, impact on other people in the same room (e.g. when audio is involved); only *SBs* visible to the user from her current position, having all required capabilities, and not busy showing something else, can participate to the sub-team bidding for a multi media presentation.

7 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.

In the near future, we will focus on practical experimentation and application to our domain, i.e. multi-media, multi-modal cultural information delivery in active museums. The preliminary performance results shown above seem positive.

Longer term research should focus on two main areas: overhearing and computational models. We envision the creation of overhearers agents, whose goals include

supervising the behavior of implicit organizations, possibly deciding the “best” policy in a given environment, and providing context-sensitivity to applications.

References

1. 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*, pages 1280–1287. ACM Press, 2002.
2. P. Busetta, M. Merzi, S. Rossi, and F. Legras. Intra-role coordination using group communication: A preliminary report. In F. Dignum, editor, *Advances in Agent Communication*, LNAI. Springer Verlag (to Appear).
3. 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.
4. K. Cheverst, N. Davies, K. Mitchell, A. Friday, and C. Efstratiou. Developing a context-aware electronic tourist guide: Some issues and experiences. In *Proceedings of CHI 2000*, Amsterdam, 2000.
5. P. R. Cohen and H. J. Levesque. Teamwork. Technical Report 504, AI Center, SRI International, Menlo Park, CA, 1991.
6. K. Ducatel, M. Bogdanowicz, F. Scapolo, J. Leijten, and J-C. Burgelman. Scenarios for ambient intelligence in 2010. Technical report, Information Society Technologies Programme of the European Union Commission (IST), February 2001. <http://www.cordis.lu/ist/>.
7. E. H. Durfee and V. R. Lesser. Negotiating task decomposition and allocation using partial global planning. In L. Gasser and M. N. Huhns, editors, *Distributed Artificial Intelligence, Volume II*, chapter 3, pages 229–244. Morgan Kaufmann Publishers, 1989.
8. Foundation for Intelligent Physical Agents. FIPA Communicative Act Library Specification. <http://www.fipa.org/repository/cas.html>.
9. R. E. Grinter, P. M. Aoki, A. Hurst, M. H. Szymanski, J. D. Thornton, and A. Woodruff. Revisiting the visit: Understanding how technology can shape the museum visit. In *Proceedings of ACM Conf. on Computer Supported Cooperative Work*, New Orleans, LA, 2002.
10. Barbara Grosz, Luke Hunsberger, and Sarit Kraus. Planning and acting together. *AI Magazine*, pages 23–33, Winter 1999.
11. Sanjeev Kumar, Marcus J. Huber, and Philip R. Cohen. Representing and executing protocols as joint actions. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 543–550. ACM Press, 2002.
12. Joseph McCarthy. Active environments: Sensing and responding to groups of people. *Personal and Ubiquitous Computing*, 5(1), 2001. available at <http://www.inf.ethz.ch/vs/events/dag2001/>.
13. E. Not, D. Petrelli, O. Stock, C. Strapparava, and M. Zancanaro. The environment as a medium: Location-aware generation for cultural visitors. In *Proceedings of the workshop on Coherence in Generated Multimedia at INLG'2000*, Mitze Ramon, Israel, 2000.
14. 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.
15. M. Tambe. Towards flexible teamwork. *Journal of Artificial Intelligence Research*, 7:83–124, 1997.
16. Michael J. Wooldridge. Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, 10(2):115–152, February 1997.
17. M. Zancanaro, O. Stock, and I. Alfaro. Using cinematic techniques in a multimedia museum guide. In *Proceedings of Museums and the Web 2003*, Charlotte, NC, March 2003.