Engineering Agent Societies: A Case Study in Smart Environments*

Alessandro Ricci
DEIS, Università di Bologna
Via Rasi e Spinelli 176
47023 Cesena (FC), Italy
aricci@deis.unibo.it

Andrea Omicini
DEIS, Università di Bologna
Via Rasi e Spinelli 176
47023 Cesena (FC), Italy
aomicini@deis.unibo.it

Enrico Denti
DEIS, Università di Bologna
Via Risorgimento 2
40136 Bologna, Italy
edenti@deis.unibo.it

Categories and Subject Descriptors

D.2.2 [**SOFTWARE ENGINEERING**]: Design Tools and Techniques; I.2.11 [**ARTIFICIAL INTELLIGENCE**]: Distributed Artificial Intelligence—*Multiagent systems, Coherence and coordination*

General Terms

Design

1. MAS FOR PERVASIVE COMPUTING

The technological progress – concerning chip density, processor speed and network bandwidth, to cite some – makes it possible to conceive new classes of applications and systems, which can be generally referred as *pervasive computing*. Intelligent / Smart Environments are prominent cases of pervasive computing, whose aim is to remodel the environments where people live and act, considering the new services (such as energy management, health care, entertainment) that can be provided by embedding (intelligent) software in network-enabled subsystems, such as sensors, actuators, mobile devices and general-purpose computers. In this way, some of the tasks currently performed by humans can be automated – possibly with improvements in efficiency or quality of service [2]. Pervasive computing scenarios account for two forms of intelligence: intelligence of the individual parts, which must autonomously execute their own specific tasks despite the openness and unpredictability of the environment; and social/collective intelligence, required to provide services that necessarily involve the coordinated interaction of the autonomous parts.

Since traditional software engineering approaches are known to be inadequate to face the complexity of such scenarios, new paradigms and models are required. Accordingly, approaches based on the *multiagent system* (MAS) paradigm provide the means to support the engineering process at the proper abstraction level, from analysis to deployment. In particular, the notion of agent provides the characteristics in terms of autonomy and pro-active behaviour to design and develop effectively individual parts; then, the notion of *society of agents* – which distinguishes multi-agent systems from simple aggregation of agents – is fundamental to engineer the *social aspects*, which concern social tasks, accounting for agent coordination and cooperation (the collective intelligence), organisational rules (norms) and environmental constraints.

The case study considered in this paper – the management of lights inside a building – is taken here as a simple representative of the aforementioned scenarios: our aim is to show the benefits of adopting methodologies and infrastructures based on the MAS paradigm that explicitly account for the social aspects as first class entities.

As a methodology, we adopt SODA [3], which provides a coherent conceptual framework and a comprehensive software engineering procedure that account for the analysis and design of agent societies and agent environments. As a coordination infrastructure, we adopt TuCSoN [5], which provides a run-time support and related IDE-tools to support the development and deployment stages. Both SODA and TuCSoN focus on inter-agent issues, and deeply rely on the notion of *coordination model* for this purpose. In the SODA design stage, coordination models and languages are taken as the sources of the abstractions and mechanisms required to engineer agent societies: social rules are designed as coordination laws and embedded into coordination media, and social infrastructures are built upon coordination systems. At the development stage, TuCSoN provides *tuple centres* [4] as coordination media, and the ReSpecT logic language to specify coordination laws, embedded in tuple centres for ruling agent interactions. At the deployment stage, tuple centres live in the TuCSoN infrastructure runtime, which provides (agent and human) tools to support their management, such as the dynamic re-programming of the behaviour to change coordination policies.

2. THE LIGHT MANAGEMENT CASE STUDY

The case study discussed here, inspired by [1], concerns the intelligent management of lights in a building. Despite its simplicity, this represents a broad class of applications, where autonomous and (possibly) mobile agents interact in an open environment, like the Internet. The intended scenario assumes that *visitors* of the building move from room

*This work has been partially supported by MIUR, and by Nokia Research Center, Burlington, MA, USA.

AAMAS'02 July 15-19, 2002, Bologna, Italy.

to room, and have each his/her own preference about the light intensity in each room. The application goal is to manage the room lights so as to automatically adapt their intensities to the visitors' preferences while respecting the existing constraints – and more generally to apply light policies that could change over the time. Such policies also reflect rules concerning the specific room interaction context – according to the room purposes –, as well as the global (building) rules – here related to energy saving. So, the intelligent environment is asked to mediate between the various light preferences when multiple visitors visit the same room, enforcing light management policies.

Both individual and social tasks must be explicitly modelled and designed: visitors have private goals – getting the best light setting according to their preferences – that they try to pursue by their individual tasks. In order to provide a valuable service to the whole society of visitors, an intelligent environment requires that the individual behaviours are coordinated.

3. ENGINEERING WITH SODA AND TuCSoN

Using the SODA methodology, individual and social tasks are explicitly described, identifying individual roles and group. As individual roles, we identify the *Visitor*, the *Light Master*, and the *Light Installer*. The Visitor role represents visitors moving from a room to another: its only task (responsibility) is to express a preference about the light intensity of a room. The task of the Light Master role is to change the light policies of the rooms when needed, while the role of Light Installer is to install / remove room lights when requested. As a social abstraction, we identify the *Room Group*, which is composed by all the Visitors currently in a room: its social task is to set the room light intensity according both to the current light policy of the room, and to the preferences of the current room Visitors. With respect to the resource model, the following types of resources can be identified: the *Building*, the *Rooms*, and the *Light Sources*. The main interaction protocol enables visitors to express their current preference about the desired light intensity in the point of the room where they are currently situated. The interaction rule for the Room Group takes care of setting the light intensity according to the current light management policy of the room. This involves collecting visitor preferences, applying the light policy currently associated to the room, and then accessing the corresponding light sources so as to set the light intensity.

At the design stage, individual and social tasks are assigned as responsibilities of agent classes and societies, onto which individual roles and group are mapped, respectively. `Visitor`, `Light Master` and `Light Installer` agent classes are identified directly from the related roles. The individual tasks identified in the analysis stage drive the design of the structure and behaviour of the agent classes. The Room Group is mapped onto the *Room* society abstraction, which is designed around the *Light* coordination medium. The *Light* coordination medium embeds the interaction rule about light management policies and enables the interaction protocols of the agent classes.

Application development and deployment are then made on a mobile agent platform based on the TuCSoN coordination infrastructure. TuCSoN allows the developer to keep the concerns about individual and social aspects separated also in the development stage: individual tasks and aspects assigned to agent classes in the design stage drive the development of single agents, while social tasks assigned to coordination abstractions drive the development of tuple centre behaviours.

From the topology viewpoint, one distinct TuCSoN node is provided for each room of the building. Agent classes are mapped onto TuCSoN mobile agents (such as `Visitor`, `LightManager`, and `LightInstaller`): there are different kinds of visitor agents, according to different ways of exploring the building. The *Light* coordination medium of each room is mapped directly onto a tuple centre named `Light`, located at the TuCSoN node of the corresponding room. The interaction rules associated to `Light`, which represent light management policies and agent coordination rules, are encoded in the ReSpecT language and used to program the behaviour of the `Light` tuple centre. The `Light` tuple centre enables and mediates also the interaction between agents and resources such as `LightSource` and `Room`, wrapped in components: this approach emphasises the role of local interaction spaces as agent interfaces to local resources, thus outlining the relevance of coordination media as *the* middleware enabling interactions, whatever they are.

Both *(i)* the uncoupled blackboard based communication model, and *(ii)* the ability of encapsulating social laws and constraints in coordination media (*objective* coordination) gave us several benefits: among the others, light policies and coordination laws could be easily changed and adapted at runtime, and new constraints added dynamically, according to the need, by changing `Light` tuple centres behaviour, supporting the openness of the system; Visitors interactions could be easily observed by inspecting `Light` tuple centres, enabling intelligent agents to reason about society evolution and to adapt light management policies accordingly; light policies and social rules governing Visitors' interactions are enforced prescriptively, despite of agent autonomy.

Future investigations will be devoted to the deployment of the SODA methodology and TuCSoN infrastructure in the design and development of more complete and demanding applications in the pervasive computing context, in particular based on wireless technology.

4. REFERENCES

[1] R. Gustavsson. Agents with power. *Communications of the ACM*, 42(3):41–47, Mar. 1999.

[2] V. Lesser, M. Atighetchi, B. Benyo, B. Horling, A. Raja, R. Vincent, T. Wagner, P. Xuan, and S. Zhang. The UMASS intelligent home project. In *3rd International Conference on Autonomous Agents (Agents '99)*, pages 291–298. ACM Press, 1999.

[3] A. Omicini. SODA: Societies and infrastructures in the analysis and design of agent-based systems. In P. Ciancarini and M. J. Wooldridge, editors, *Agent-Oriented Software Engineering*, volume 1957 of *LNCS*, pages 185–193. Springer-Verlag, 2001.

[4] A. Omicini and E. Denti. From tuple spaces to tuple centres. *Science of Computer Programming*, 41(3):277–294, Nov. 2001.

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