Energy Saving and Added Customer Value in Intelligent Buildings*

Magnus Boman[1,2], Paul Davidsson[1], Nikolaos Skarmeas[3], Keith Clark[3], and Rune Gustavsson[1]

(mab@dsv.su.se, pdv@ide.hk-r.se, nik@otenet.gr, klc@doc.ic.ac.uk, rgu@ide.hk-r.se)

[1]Department of Computer Science and Business Administration
University of Karlskrona/Ronneby
Soft Center, SE-372 25 Ronneby, Sweden

[2]Department of Computer and Systems Sciences
Stockholm University and the Royal Institute of Technology
Electrum 230, SE-164 40 Kista, Sweden

[3]Agent Solutions / Department of Computing, Imperial College
London, UK

The usefulness of agent technology in the domain of power distribution and building automation is investigated. A system consisting of a collection of software agents that monitor and control a small office building using the electrical devices present in the building has been developed. Communication between agents and devices is achieved via the existing power lines. The objectives of the application are both *energy saving* (by controlling lights, heating, ventilation, etc.) and *enhancement of customer value* (by taking into account the personal desiderata of the people in the building). The agent system has been designed to be easily configured and customised, in order to deal with different building environments. We are currently experimenting with simulations, and preparing for fielded experiments that will commence in Villa Wega (Ronneby, Sweden) where a large part of the required hardware is already in place.

1 Introduction

A main business process of the energy sector has been concerned with the production and distribution of KWh. Due to two main forces, de-regulation of the energy market and advancements in information technology, the energy market is now undergoing dramatic changes. A key technological driving force is that the power grid also can be used as a data communication channel with increasingly higher bandwidth, over one megabit per second on parts of the grid is now possible. Another key technological driving force is the rapid development of technologies supporting smart distributed systems communicating on the electric grid.

* The work described in this report is part of the Information/Society/Energy/System (ISES) project; see `www.enersearch.se/projects/`, sub-project #9: “Robust Distributed Decision Islands”.

In a de-regulated market a customer can more easily change supplier, so the utility companies have to compete with added value for the customer on top of the delivery of KWh. For example, in October 1997, the first petroleum company entered the electricity market in Sweden, with the intent to gain market shares with low prices and knowledge of how to compete in a similar market, viz. selling gasoline and oil to individual customers.

Since the price of KWh as such does not yield sufficient market differentiation, the ability to communicate on the electric grid and piggy-back services open up entirely new business opportunities for utilities [9]. In short, the energy sector is in a transformation from a KWh-based business to a service business based on KWh. It is common to group the new kinds of services as follows (cf. [11]):

- Distribution Automation (DA) applications, which automatise the distribution process itself, and
- Demand Side Management (DSM) applications, which involve interaction with the customer.

The ISES (Information/Society/Energy/System) project has as a goal to assess and demonstrate new business opportunities for future service-centric utilities. Some demonstrators take a multi-agent system (MAS) approach. For example, an extensive investigation of the DA/DSM application of power load management [14] has been made. In essence, load management is modelled as a computational market in a society of agents. The mechanisms of the computational market are implemented as very efficient algorithms on smart equipment, Homebots, communicating on the electric grid [1]. The Homebots form the kernel of the society of smart hardware devices at the Villa Wega test site in Ronneby, Sweden. It is a three story building equipped with devices for communicating on the electric grid.

In a MAS setting, we can model services as societies of agents providing the service at hand. These societies of agents are then implemented, in a structure preserving way [8] as a society, or distributed system, of smart equipment communicating on the electric grid. Integration of services is a challenge in itself. One must deal with issues like conflicting goals, robust behavior of the total system, and adaptation. In the power domain, AI methods have been used in several related applications [2][6]. The topic of this paper is to address some of these issues in a systematic way, with the ISES goals in mind.

The present application involves a MAS used for intelligent building control and relates to Building Automation, a special kind of DSM application which aims at integrating the utility's services with other in-building computing devices. The objectives of our application are both energy saving and enhancement of customer value through value-added services. *Energy saving* in this environment is realized by controlling lights, heating, ventilation, etc. Examples include lights that are automatically switched off, and the room temperature being lowered in an empty room. *Enhancement of customer value* is realized by the system by taking into account the desiderata of the people in the building, for instance, by adapting temperature and light intensity according to each person's personal preferences.

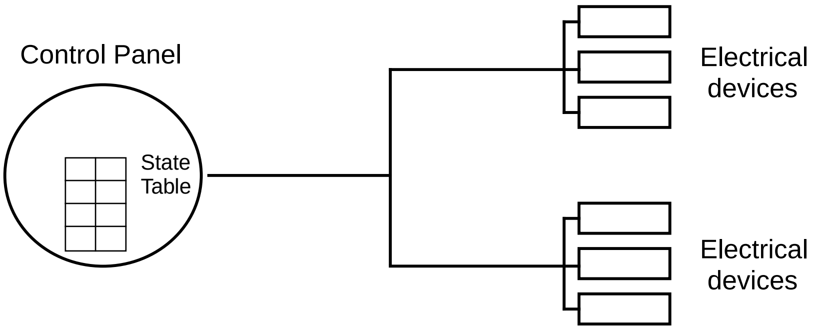

Figure 1. The hardware infrastructure.

The MAS consists of a collection of software agents that monitor and control an office building using the electrical devices already present in the building. Different agents control different parts, as well as different aspects of the environmental conditions, of the building. Other agents represent the persons in the building in order maintain their preferences concerning temperature, light intensity, etc. The goal is to make the system transparent to the people in the building in the sense that they do not have to interact with the system in any laborious manner. By using an active badge system, it automatically detects in which room each person is at any moment and adapts the conditions in the room according to that person's preferences.

The use of agent technology provides a decentralised solution with a number of advantages, like scalability and re-configurability (both dynamic and static). New agents can dynamically enter the scenario and start participating in the operation of the system without bootstrapping. Also, agents can be easily customized by adding new control policies or have old ones modified. The current version of the software consists of the agent-based control software and a simulation of the building environment. However, the intention is to test the system on an actual building and soon fielded experiments in Villa Wega will commence. As a large part of the required hardware is already in place in this building, the platform has been designed to be easily interfaced with the existing hardware.

The following section will describe the underlying infrastructure of the simulation environment and its interface to the MAS. Sections 3 and 4 will detail the MAS used to control the building devices, and in the last section we conclude with some comments on the advantages of using an agent-oriented approach, and pointers to future work.

2 An Intelligent Building Application

A building contains a number of electrical devices that constitute an important part of the infrastructure of the building. At the Villa Wega test site, the interaction with the devices at the hardware level is facilitated by an infrastructure based on LonWorks technology (cf. `www.echelon.com/`). Its conceptual structure is depicted in Figure 1.

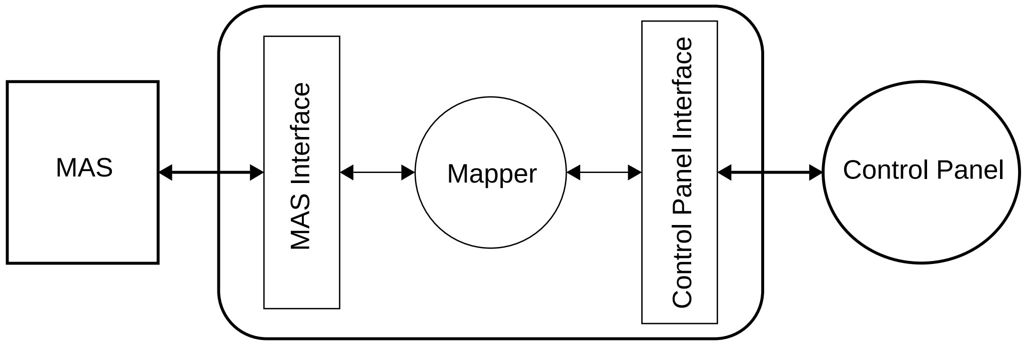

Figure 2. The control panel interface.

Each electrical device in the system is connected via special purpose hardware to the LonWorks system, allowing the exchange of information over the electrical network using the LonTalk protocol. In order to simplify the communication between the MAS and the LonWorks hardware, all the information received from the devices is recorded on a blackboard-like entity called the control panel. This information reflects the state of the devices (and therefore the state of the environment) and is stored in a special attribute-value table. However, it would be possible to let the agents of the MAS communicate directly with the hardware.

Some of the devices are sensory and some are actuator devices. The *sensory devices* (read only) assumed are: temperature, light intensity, fire detector, presence (detects whether there is activity in a room in a room or not), and an active badge system. The active badge system [10] makes it possible to know which persons are in each room at any moment. The *actuator devices* differ from the sensory devices in that it is possible, besides reading the state of the device, to change the state of the device (in order to change the state of the building). The actuator devices in the current application are lamps, radiators, and generic mobile devices* that can be connected to an arbitrary electrical device, e.g., a coffee machine, or a personal computer. It is possible to switch on and off the device connected to the generic mobile device and to read its state.

These devices interact with, and are controlled by, the MAS. The devices provide input to the MAS (the sensory devices) and occasionally receive instructions from it (the actuator devices). The interaction is mediated by the control panel and its state table using an interface that translates messages originated from the MAS to commands understood by the LonWorks system and vice versa (see Figure 2). Since the languages used to implement the agent system are April and April++, the interface is using the April and LonWorks APIs.** The role of the interface is to receive messages from both ends and perform the appropriate translations.

---

* ARIGO Switch Station, for more information, see `www.arigo.de`.

** April [12] is a process-oriented language, supporting features like communication, concurrency, pattern matching, and mobile code. April++ [4][13] is a macro-based syntactic extension of this language that augments April's functionality with object-oriented and distributed database features.

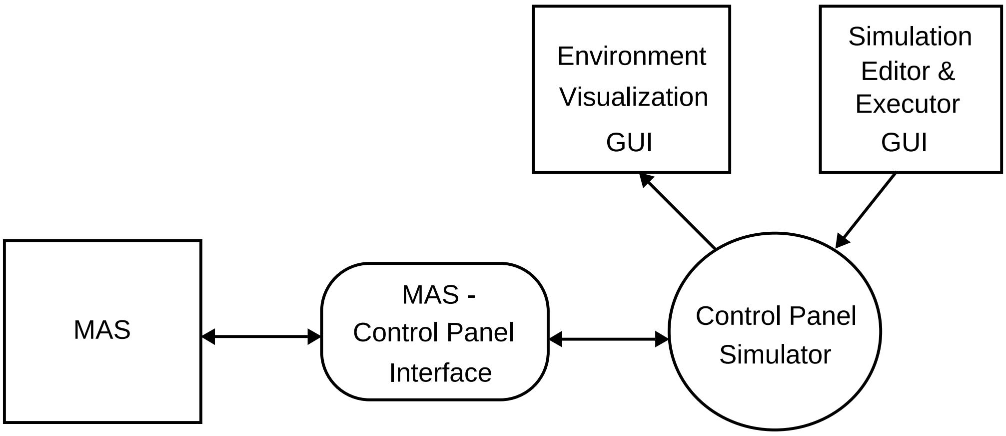

Figure 3. The simulation environment.

Currently, a simulation of the building environment is provided including a simulation of the control panel functionality. Thus, the MAS and the Interface communicates with the simulated building through the simulated control panel. The conceptual structure of the MAS and the simulation environment is depicted in Figure 3. Using this design we predict that it will be easy to interface the MAS with the actual LonWorks system of Villa Wega. The only modification necessary concerns the part in the interface that communicates with the control panel: it would have to talk to the LonWorks API.

In addition, a graphical user interface visualizing the building environment (and a simulation scenario editor and executor) has been implemented. Figure 4 provides a snapshot of the interface that visualizes the state of the building in terms of temperature, light intensity of the rooms, and the persons present in the rooms.

3 Agent-Based Building Control

During the design and implementation of the control software, a MAS approach was adopted. Agents correspond to different entities of the building, for example, to offices, meeting rooms, corridors, persons, and electrical devices. They are given a number of rules which express the desired control policies (constraints) on the building conditions. The occurrence of certain events inside the building (e.g., a person moving from one room to another) will generate messages to the MAS agents that will trigger some appropriate rule(s). The agents execute the rule(s), with the purpose to re-adjust the environmental conditions to some preferred set of values. The rule will cause a sequence of actions to be executed, which will involve communication between the agents of the system. For the format of the messages a KQML [7] like approach is followed.

Figure 4. The Environmental Visualization GUI

3.1 Types of Agents

There are four main categories of agents in this application:

- *Personal Comfort (PC) agents*, which each corresponds to a particular person. It contains personal preferences and acts as a surrogate of the person in the multi-agent system trying to maximize customer value. Thus, the agent does not model the behavior of a person, rather it tries to act on that person's behalf, in her interest.
- *Room agents*, which each corresponds to and controls a particular room with the goal of saving as much energy as possible. Taking into account the preferences of the persons currently in the room, it decides what values of the environmental parameters, e.g., temperature and light, are appropriate.
- *Environmental Parameter (EP) agents*, which each monitors and controls a particular environmental parameter in a particular room. They have access to sensor and actuator devices for reading and changing the parameter. For instance, a temperature agent can read the temperature sensor and control the radiators in a room. The goal of an EP agent is to achieve and then keep the value of the parameter decided by the Room agent.
- *Badge System Agent (BSA)*, which keeps track of where in the building (in which room) each person, i.e. badge, is situated at any time.

As shown in Figure 5, the agents have been divided into three groups. This division has been done for both conceptual and administrational purposes. The PC agents may reside on the indi-

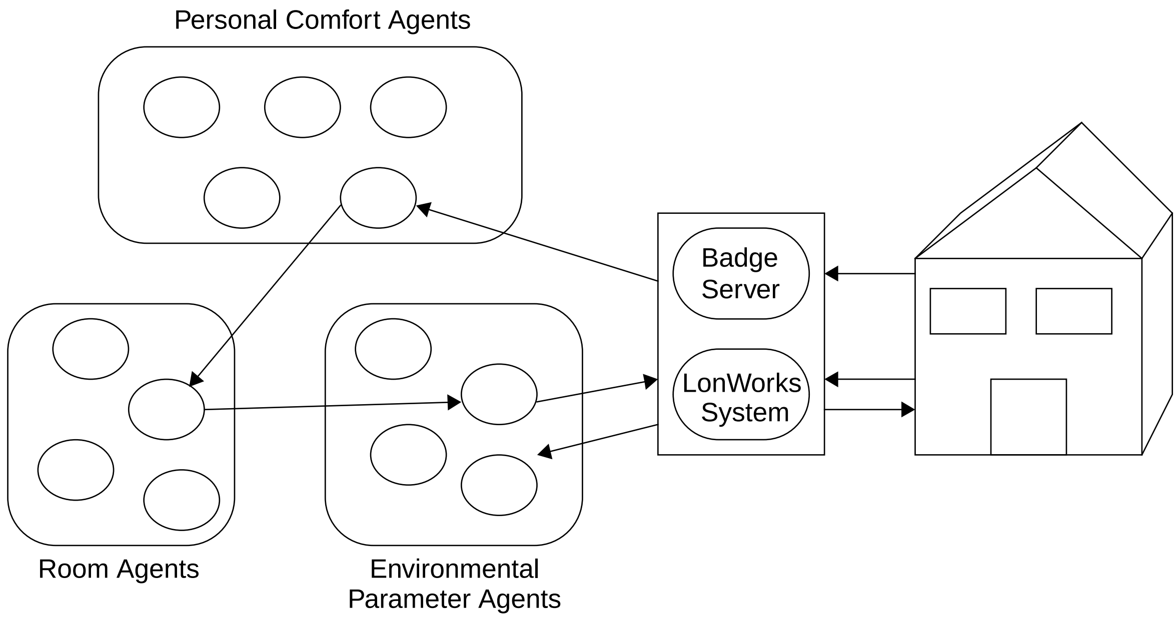

Figure 5. The multi-agent system.

viduals' desktop computers and interact locally with the corresponding person, e.g., in order to change the preferences. Normally, the preferences are set when the agent is initiated, i.e., when the person visits the building for the first time, and rarely changed. When initiated, the PC agents register with the BSA which maintains a data base of the PC agents and their associations to persons. Thus, the BSA, in addition to providing the interface between the badge system and the PC agents, plays the role of a directory. When a person movement is detected (movement from one room to another in the building or movement into/out of the building) the BSA informs the appropriate PC agent about this movement.

When a PC agent is notified about a person movement, it informs the appropriate room agents, i.e., the agent of the room the person is leaving and the agent of the room the person is entering. The PC agent also provides the room agent with the personal preferences. The room agent decides, based on these preferences and on energy saving considerations, the new desired environmental conditions and pass them on to the EP agents. If necessary, the EP agents then send messages to the actuator devices via the interface described in the previous section. When the EP agents are started up, they register with the room agents they correspond to and also to a directory that the interface uses when sending messages to the agents of the MAS. Using this directory, the interface can find out exactly which EP agent it needs to talk to.

From the above description, the distributed nature of the application becomes apparent. We make no assumptions about the agents' locations in the network. Personal comfort agents can be distributed around the network in each person's desktop. The rest of the software can be distributed as well, with each of the other two groups residing on a different machine, and with EP agents actually residing in the hardware connected to the devices.

3.2 System Constraints

The system conforms to a number of general rules (constraints or decision policies) that are fed to the agents. Some examples are listed below:

- Every room with no persons in it must maintain some default environmental conditions, e.g., a certain temperature, or states of devices, e.g., radiator - off.
- For common rooms, like corridors, the temperature remains steady regardless of the persons in a room, and the light is turned on only when at least one person is in the room, otherwise it is turned off.
- When a particular person is in her office, the room agent must adapt temperature, light, etc. to her preferences, otherwise the default conditions are maintained. If an irrelevant person (i.e., another person than the ones that normally work in the office) enters that office, this does not affect the environmental conditions (except for that the light is turned on if the room was empty).
- For meeting rooms, the temperature condition is adjusted to the mean value of all the meeting participants, and the light intensity to the highest preference value.
- It must always be possible to over-rule the decisions of the agents in the MAS by physical interaction with the electrical equipment. For instance, even if an EP agent has decided that the light in a room should be on, it must be possible for a person to turn off the light using the switch in the actual room.
- The persons are also allowed to provide the PC agents with different preferences depending on the activities they are undertaking. For example, it is possible to specify different light or temperature conditions for working activities different from those for meeting activities.

These constraints are naturally not hard-wired, and are subject to constant change.

Usually, the goals of the room agents and the PC agents are conflicting: the room agents maximizing energy saving and the PC agents maximizing customer value. Another type of a conflicting goal situation would be the adjustment of temperature in a meeting room in which people with different preferences regarding temperature will meet. We are currently investigating the use of decision modules (sometimes called oracles, or decision machines) to address this problem with possible extensions of using the notions of group utility and norms for dealing with problems arising from agent negotiations [3].

4 The Agent Architecture

In general, each agent contains a number of components that perform a specific task, contributing to the overall functionality of the agent. For example, in a room agent, one component stores the rules that implement the control policies. Another component stores agent data, a third communicates with the devices in the room, etc. The approach described in [13] has been

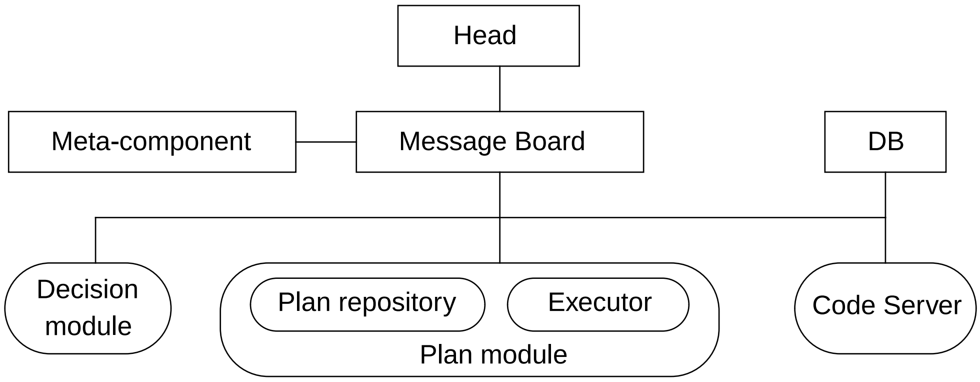

Figure 6. The room agent architecture.

adopted and a room agent in view of this architecture is depicted in Figure 6. According to this architecture a number of generic modules (rectangular boxes) are included in each agent by default but it is possible to add other domain dependent modules (rounded boxes).

One of these generic modules is the *head* that plays the role of the communication firewall of the agent. All the messages directed to the agent are sent to the head and they are subsequently forwarded to the internal modules. In this way external entities do not need direct access to the agent's modules. Also, a shared knowledge base (*DB*) is included that the modules can use to store shared information. This knowledge base makes use of the advanced distributed deductive capabilities that AprilQ offers [5]. The *meta-component* is used for administrative purposes during the addition and deletion of components. Finally, the communication between the components is facilitated by the *message board* which is the communication backbone of the agent. The message board provides routing of messages based on the symbolic names of the components, and content-based routing which uses active patterns [13] for processing the message content, in case the destination of the message is unknown. Therefore, components of the agent can route messages to another component by specifying its symbolic name as destination, or by not specifying any destination in the message in which case the pattern-based mechanism is used.

Depending on the situation, the agent needs to execute a sequence of actions, i.e., a plan. The *plan module* is responsible for maintaining such plans and consists of the *plan repository* and the *plan executor*. The plan repository is the component where the plan descriptions are stored. The descriptions involve the name of the actions involved in the plan, their temporal relationships, and descriptions of the information they manipulate. Requests for executing single plans are received by the plan executor. The executor will fetch the code that implements the specific actions from the *code server* and execute them. The code stored on a code server can be supplied on demand. An alternative approach would be to have the actions hard-wired in the decision module. Although our approach requires some additional communication among the components in order to retrieve the code, agents that adopt this approach can be configured more easily. New actions are simply added to the code server and new plans of actions can be added to the plan library maintained by the planning module. The communication between plan module, code executor, code server, and the decision module is achieved via the message board.

Exactly which plans to be executed is decided by the *decision module*. The room agent receives external events like “a person has entered the room”. Based on its current state and the external event it decides which course of action will be taken. For this purpose, the decision module maintains and uses a state transition table. Each entry in this table consists of a state, an event, a plan, and the new state (of the agent after the plan has been executed). When a new message arrives at the decision module, it searches the list of transitions for the entry matching the current state and event. Since we expect different decisions in the case of a meeting room and a personal office, the set of transitions is dependent on the type of room and each room agent loads the appropriate set of transitions when it is started up.

For example, assume that a person whose name is “Nikos” is entering a new room. For the room agent (whose name is *“Room_0_0”*), a message will arrive from Nikos’ personal agent and will inform the PC agent of the entrance of the new person:

```
(inform,  (From, “NikosAgent”),
          (To, “Room_0_0”),
          (content, (new_person, (“Nikos”,”NikosAgent”))))
```

This message will activate a plan in the room agent, named *person_enters_room* and which will cause the execution of the following actions:

(1) Get person preferences: retrieve the preferences from “NikosAgent”.

(2) Decide new conditions: Compute the new desired conditions based on the policies of the room.

(3) Inform the EP agents about these conditions.

To summarise, the plan module uses a library of plans and the decision module uses a library of state transitions depending on the type of the room. The actions to be executed are stored in the code server. This approach makes the agents highly modular. Different agents can be generated and programmed by simply changing the descriptions loaded. Their behaviour can be modified dynamically by changing these descriptions of the plans or the state transitions. No changes have to made in the core code of the agent.

Although only the structure of the room agent was detailed, a similar methodology was adopted for the development of the PC agents. The EP agents have got simpler structure and are implemented as April processes.

5 Conclusions and Future Work

We have given a high-level description of a current project aimed at investigating the usefulness of the agent metaphor and the notion of multi-agent systems for the design of control systems for intelligent buildings. The use of the agent approach was initially motivated by the close mapping that this approach offered between the entities of the application domain and the entities of the software. The concurrent non-deterministic nature of the activities inside the building was another factor that led to the development of concurrent autonomous entities. This aspect of the agent systems was fully exploited by the choice of a concurrent language for implementing the agents, viz. April++. Finally, the agent system as it was designed, allowed for the dynamic re-configuration of the agents, without any disruptions of the operation of the system. This is a useful feature when changes in the building infrastructure or of the persons in the building occur.

Although the system offers a testbed for testing some ideas and examining the feasibility of an agent-based approach a number of improvements could be made. We are currently working on integrating the PC-agents with personal agendas in order to make better decisions. For example, if the agent knows that the person will be missing from her office to attend a one-hour meeting, it can inform both the office room agent and the meeting room agent about this fact. In this way, we increase the degree of pro-active behaviour of the agents of the MAS. We are also implementing more complex functionalities, e.g., when a person enters the building in the morning, her monitor is switched on, as is the coffee machine.

When the simulations have been fully evaluated and the MAS optimized accordingly, the next step of the project will be to make the actual transition from simulation environment to physical implementation. A further step would be to combine the services of our system with the services of the load balancing system mentioned in the introduction. We then have to design the interface between the multi-agent systems and the physical devices of the building to use and reuse the same data from the control panel, but for different services.

We are also investigating robustness issues. For instance, we are working on ways to store the agent state persistently and in the case of system or agent crash to be able to re-launch the agents in the state they had just before their abnormal termination. Finally, we will address issues of different types of conflict resolution, as suggested in Section 3 above.

Acknowledgements

The authors would like to thank Christoffer Dahlblom, Martin Fredriksson and Mikael Svahnberg who implemented the simulation environment, and to Marko Krejic and Richard Krejstrup for helping us with hardware-related issues (all are M.Sc. students at the University of Karlskrona/Ronneby).

References

[1] Akkermans H., Ygge F., and Gustavsson R., “HOMEBOTS: Intelligent Decentralized Services for Energy Management”, *Fourth International Symposium on the Management of Industrial and Corporate Knowledge (ISMICK’96)*, 1996.

[2] Bann J. J., Irisarri, G. D., Mokhtari S., Kirschen D.S. and Mille, B. N., "Integrating AI Applications in an Energy Management System", *IEEE Expert 12(6)*, pp. 53-59, 1997.

[3] Boman M., “Norms in Artificial Decision Making”, *Artificial Intelligence and Law* (in press).

[4] Clark K. L., Skarmeas N. and McCabe F. G, “Agents as Objects with Knowledge base state”, *ICMAS’96*, pp. 33-40, 1996.

[5] Clark K. L. and Skarmeas N., “A Harness Language for Cooperative Information Systems”, *Cooperative Information Systems: Trends and Directions*, M. P. Papazoglou and G. Schlageter (eds.), Academic Press, 1998.

[6] Dabbaghchi I, Christie R. D., Rosenwald G. W. and Liu C. "AI Application Areas in Power Systems", *IEEE Expert 12(1)*, pp. 58-66, 1997.

[7] Finin T., Fritzson R., McKay D. et al., “KQML as an Agent Communication Language”, CIKM'94, pp.456-463, ACM Press, 1994.

[8] Gustavsson R., “Multi Agent Systems as Open Societies”, *Intelligent Agents - Theories, Architectures, and Languages (ATAL’97)*, Springer Verlag, 1997. (In press)

[9] Gustavsson R., “Requirements on Information Systems as Business Enablers”, *DA/DSM Europe DistribuTECH’97*, PennWell, 1997.

[10] Harter A and Hopper A, “A Distributed Location System for the Active Office”, *IEEE Network 8(1)*, 1994.

[11] Hägg S. “Agent Technology in Industrial Applications”, *Proceedings of the Australia-Pacific Forum on Intelligent Processing and Manufacturing of Materials (IPMM’97)*, 1997. (In press)

[12] McCabe F. G. and Clark K. L., “April: Agent Process Interaction Language”, in Wooldridge M. J. and Jennings N. R. (eds.), *Intelligent Agents, Lecture Notes in Artificial Intelligence, 890*, pp. 324-340, Springer-Verlag, 1995.

[13] Skarmeas N., “Agents as Objects with Knowledge Base State”, PhD Thesis, Imperial College, Department of Computing, January, 1997.

[14] Ygge F. and Akkermans H., “Power Load Management as a Computational Market”, *ICMAS’96*, pp. 393-400. AAAI Press, 1996.