Research Report 10/97

HÖGSKOLAN KARLSKRONA RONNEBY

Agent Technology in Industrial Applications

by

Staffan Hägg

Department of
Computer Science and Business Administration
University of Karlskrona/Ronneby
S-372 25 Ronneby
Sweden

ISSN 1103-1581
ISRN HKR-RES—97/10—SE

Agent Technology in Industrial Applications
by Staffan Hägg

ISSN 1103-1581
ISRN HKR-RES—97/10—SE

Printed by Psilander Grafiska, Karlskrona 1997

AGENT TECHNOLOGY IN INDUSTRIAL APPLICATIONS

Staffan Hägg*

*Department of Computer Science, University of Karlskrona/Ronneby, S-372 25 Ronneby, Sweden, Staffan.Hagg@ide.hk-r.se, http://www.sikt.hk-r.se/~staffanh/

This research is sponsored by the Information/Society/Energy/System (ISES) project, which is a joint project between the University of Karlskrona/Ronneby, Sydkraft AB, Enersearch AB, IBM Utility, Preussen Electra AG, Electricité de France, ABB Network Partner, Ronneby Energy, and the City of Ronneby.

ABSTRACT

The agent metaphor has become increasingly popular in distributed computing systems. As often with a new paradigm, it is not very clear what the exact meaning is. The meaning and the development of the agent paradigm is therefore discussed in this paper. One motivation for using agents in distributed systems is that, due to the rapid technical development of computing and communication facilities, these systems become larger, more complex, and with more heterogeneous components. The change rate for such systems will also increase. Another, but related, motivation is that, in many cases, the systems are *open* and *emergent* in new respects: Different parties will be in control of parts of the total system, and the change of the system may not be altogether predictable. It is simply not possible to describe the total system from any one point of focus or at any one point in time, meaning that traditional methods for describing and implementing the systems are not sufficient.

Industrial applications (e.g. manufacturing systems, process control systems, or resource management systems) show many of the characteristics mentioned above. Specifically, there is a demand for integrating quite dispersed applications, for example systems for process control, market analysis, and economic management. There is also a stronger demand for communication and interacting with sub-contractors, partners, and customers. This is emphasized with the development of a common communication medium like the Internet.

In the ISES project, the agent paradigm is studied and developed for use in Power Distribution Automation. Examples of this work are given here, and the applicability of agent technology is shown for applications that concern Distribution Automation, Demand Side Management, and Home Automation. It is also argued for the applicability of agent technology in other types of industrial applications, based on similarities between problem domains.

INTRODUCTION

The rapid development in the computing and communication areas has opened for wide areas of new applications, and information processing and control systems may be distributed to a degree that was not imaginable just a few years ago. Also, demands from users are ever increasing, as human efficiency is more focused, both commercially, being an emphasized competition factor and, perhaps more surprisingly, in relation to people's leisure activities. Such increased demands relate to, for example, availability, response times, integration of dispersed applications (not only technically, but also semantically), "easy-to-use", and adaptability to individuals. In this paper we analyse the scope of this development in terms of how to describe and model such systems, especially for industrial applications, and we argue that the agent metaphor is well suited for this. After an overview of the ISES project, we analyse the basic problems. Then we discuss the agent metaphor and see how agent technology is used in the ISES project. We show an example from the project, describing a simulation of an energy saving application. The conclusions aim at showing that agent technology is well suited for many industrial applications, based on domain similarities.

THE ISES PROJECT

Power distribution has traditionally been focused on the task of delivering energy from a utility company to customers. This is true for a number of forms of energy, but it is especially significant for electricity distribution, electricity being produced more or less at the same time and in the same amount as customers demand it. However, the context wherein electricity is delivered is in the process of getting more complex for utilities and also for their customers. For utilities, it means offering new services to customers. It also includes new methods for optimizing the distribution process itself. Customers, on their part, can choose between suppliers. They can, and probably will, decide on supplier, not only from price, but from additional services, quality of information, etc. In the power industry [1], these applications are given considerable attention, an we may group them as follows:

- *Distribution Automation* (DA) are applications that automatize (and optimize) the distribution process.
- *Demand Side Management* (DSM) applications involve interaction with the customer.
- *Home Automation* (HA) applications integrate the utility's services with other in-house computing devices.

The present research is performed as part of the Information/Society/Energy/System (ISES) project whose purpose is to automatize and develop the process of distributing energy from producers to customers, and the contributors to the project are major actors on the European energy market, their suppliers of sub-systems, and community representatives[1].

PROBLEM ANALYSIS

As a result of the development mentioned earlier, demands on the software systems are increased, and new problems are introduced. We think the problems listed in Figure 1 are real in many of

1 See the affiliation section on the first page.

today's and tomorrow's distributed systems, and we argue for the need of a new paradigm for describing and realizing them, as traditional methods have problems with meeting the demands. Items (1) and (2) say that when a system becomes large or complex enough, they simply become unmanageable with tools like class libraries, data dictionaries, etc. Item (3) says that when system components are very heterogeneous, it gets less realistic to translate between specific representations; a generic method for handling different representations will be more desired. Item (4) means that unlike traditional systems, new systems will more often be controlled by many parties. Different organizations own their parts of the total system and the system is not even in principal describable from any one point of focus, neither in design nor in run-time. Item (5) means, in combination with item (4), that these systems may be subject to unpredictable and rapid change. The system is not describable at any one point in time, and methods for incremental development are needed. The last item is a generalization of items (4) and (5) (and, to a lesser extent, also of the former items). It says that as the focus point, from where the whole system can be observed (grounded), is removed, a semantic problem is introduced[1]. There exists no way of guaranteeing that the meaning of references are the same throughout the system, and the consequences of this may be fatal. This is more elaborated upon in the next section.

(1) Size
(2) Complexity
(3) Heterogeneity
(4) Control
(5) Change
(6) Semantics

Figure 1. Basic problems

THE AGENT METAPHOR

The agent concept has become very popular but it is not equally well-defined. It is used to describe anything from autonomous, DAI inspired, robots or processes that can act "intelligently" in a partly known environment, to quite simple pieces of code moving across the Internet. Here, we shall concentrate on some aspects of agenthood that maps well onto our previous problem description.

First, agents may be used to cope with the problems of size and complexity. Traditionally, addressing is dependent on fixed addresses. This approach may then be loosened with the help of name servers, yellow pages, etc., but these have to be updated which can take a lot of effort. When a message is to be reached by one or a few of a large number of potential receivers, it would be nice to have an access method where the sender of a message did not have to know the locations (or, perhaps, not even the names) of the receivers. Such approaches are suggested by, for example, Hägg and Ygge [2] and McCabe and Clark [3] using semantic or content-based addressing. In these examples, messages are delivered to agents who have a specific interest in those messages. The content of a message is considered, as well as both the static properties of agents and their states.

The semantic problem of a system without a single point of observation, or grounding, is thoroughly described by Werner [4]. He even means that this is the starting point for what may be called Multi-Agent System research. The situation is shown in Figure 2. In (a) there is a central observation point, at least at design time. Black agents (ellipses) means that they are observed (and, thereby, given the

1 Agents with identical descriptions of an object, relation, etc. may have different models. If an agent tries to solve this by communicating a description of its model in a higher order language, this description may then have different models for different agents, and so on ad infinitum. This is a classical problem in linguistics and philosophy, as well as in computer science research.

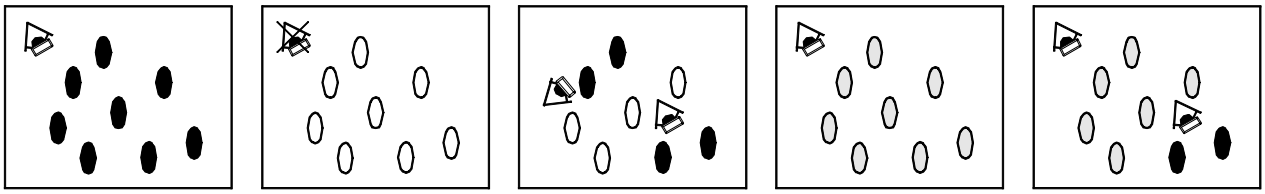

Figure 2. The semantic problem of a Multi-Agent System

same semantics, i.e they are semantically grounded). In (b) the observation point is removed. This may lead to misconceptions between agents. There is no common meaning of objects, functions, etc. This is, as previously mentioned, unsolvable in the general case. In (c) local observation points are entered, allowing for semantically well-defined sub-systems. This could be, for example, a banking system on the Internet. Agents are grey in (d). It means that they have some commonly defined semantics as basic notions of social behaviour. Much research is devoted to these questions (see e.g. Castelfranchi and Conte [5]). It includes rules for agreement, joint action, etc. However, applications also require more specific definitions, as in (e) where the approaches from (c) and (d) are combined.

Finally, the heterogeneity problem is addressed with, for example, the KQML [6] and KIF [7] languages. They allow for information sharing between agents by means of defining a common ontology for interaction purposes. Heterogeneous sub-systems may cooperate on a peer-to-peer basis.

AGENT TECHNOLOGY IN THE ISES PROJECT

In the ISES project computing devices are communicating on the low-voltage grid. These devices are connected to temperature sensors, activity detectors, and consumption meters. They also control warm water heaters, radiators, and light switches, and they communicate with the utility's main computers. Our plan is to include communication with other in-house computing devices like an alarm system, an internet connection, and any equipment that can be supplied with an interface that at the minimum can function as an on/off switch (e.g. a vacuum cleaner, a coffee machine, or a toaster). It means that we have a huge infrastructure for distributed computing which may contain all the demands and problems listed in the problem analysis above. Our experiences from applying an agent-oriented approach include the following:

- Development of an agent architecture that uses semantic addressing for communication in an unpredictable and changing environment, Hägg and Ygge [2].
- Development of a system structure where sentinels guard specific functionality and guard against undesired states, Hägg [8].
- Implementation of remote meter reading where agents, connected to meters, send consumption information to centrally located systems, the IBM IDAM system [9].
- Designing a distributed load balancing algorithm as an electronic market place. The algorithm has extremely nice complexity properties, Ygge and Akkermans [10].

ENERGY SAVING, AN EXAMPLE FROM HOME AUTOMATION

The following example shows ongoing work with automatizing energy saving functions in an office building. The application is highly interesting from both an economic and an environmental point of

view, and it is performed within the ISES project. Initially we are working with a simulation of the house, before we make a test implementation in one of the university laboratories, scheduled to early 1998. Here we look at the simulation model.

Interface equipment

Input devices: room temperature sensor, I/O agent: <ID, type, Centigrade output[1]>; IR ("activity") sensor, I/O agent: <ID, type, binary output>.
Output device: radiator, I/O agent: <ID, type, binary input>.
Input/Output devices: light switch, I/O agent: <ID, type, binary input, binary output, binary state>; connector interface, I/O agent: <ID, type, binary input, binary output, binary state>.
Communication: RS-232C (operational), programming and testing interfaces.

Interface equipment are represented by simple agents, called I/O agents, that send and receive messages for status reporting and control. We are restricted to using quite small and simple agents here, as our equipment for this purpose are very limited in terms of memory, and we keep this restriction also for the simulation. The operational communication interface allows connection to other computers that can host more complex agents, programming and user interfaces, and bridges to other systems.

Basic entities

Rooms: room agent: <ID, type, set of types of I/O agents, internal state, set of goals, set of plans>.
Activities: activity agent: <ID, type, internal state, set of goals, set of plans>.

A room is a basic entity within the house. It can be an office, a conference room, a kitchen, a corridor, etc., and new types of rooms can be defined. Each room is represented by an agent. Goals for room agents are to contact activity agents when a specified input requires it and, thereafter, to set up new goals to maintain specific states, as required by activity agents. Plans are descriptions of sets of actions needed to perform in order to reach a goal.

An activity can be a meeting, office work, idle, cleaning, etc., and new types of activities can be defined. Each activity is represented by an agent, and activity agents are the only agents that have a notion of time (also sentinels, below). This agent, typically, holds goals that are related to energy saving. For example, an idle (i.e. an "idle" activity) agent requires a low room temperature and no lights, a cleaning agent requires all lights on but says nothing about temperature, a corridor should be dark when there is nobody in it or in the adjacent rooms, etc.

Auxiliary entities

Sentinels: <ID, type, internal state, set of goals, set of plans>.
Terminals: <ID, type, internal state, set of goals, set of plans>.

1 A temperature sensor is an input device to the system, but its I/O agent delivers an output signal. Consequently we use these, maybe confusing at first, notions of input and output. The same notions are used for all components.

A sentinel is an agent that may interact with any other agent. Its purpose is to guard certain functionality and prevent the system from entering undesired states, see Hägg [8]. It may take into consideration combination of states within different agents that are not obvious from any one of these. Terminals are agents that interface with external systems, for example a databases or a user interface.

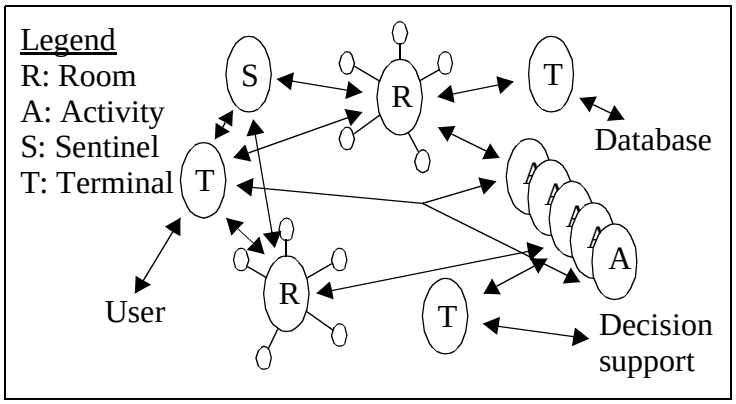

Figure 3. Sample agent interaction patterns.

Figure 3 shows an example with different agents and their interaction patterns. There are two room agents with connected[1] input/output agents. Each room agent interacts with one of a set of activity agents which support goals for that room with the current activity. One room agent reports to a database via a terminal, and an activity agent gets help from a decision support system via another terminal. A third terminal is used for user interaction. There is one sentinel. It may check that the temperature goals of the two room agents at no time differs more than a certain value, as the door between the two rooms is temporarily removed.

CONCLUSIONS

Though an application as the one above could be designed and implemented with other techniques than a multi-agent system, the agent oriented approach gives a number of advantages that, eventually, will be crucial for maintenance and further development:

- The multi-agent system allows for *incremental development*. Agents can be inserted, extracted, or modified, with minimal disturbance to the rest of the system. New functionality can be given a running system.
- With *semantic* or *content-based addressing* agents need not know all details for communication. For example, a room agent may not need to know about all I/O agents, their identities, nor even how many they are, in order to interact with them, as messages are routed to agents dependent on the message content.
- The level of interaction is important to the possibility of *integrating heterogeneous softwar*e with different architectures, internal representations, etc. Agents communicate on a high abstraction level, while other systems may be dependent on a common platform for object evaluation (Corba, Active-X, Java Beans, etc.), a common communication protocol (e.g. TCP/IP), or even lower level signalling. It is also important to the *integration of legacy software*. Genesereth and Ketchpel [11] address this issue, introducing the idea of agentification, and our terminal agents realize exactly this.
- The agent approach allows for including a number of techniques that are developed in this area of research: personal agents, multi-agent learning, etc.

1 These agents are not really connected to a room agent, but semantic addressing allows us to view them as connected.

Though agent technology is often associated with domains like personal interfaces, electronic commerce, and internet computing, quite a number of industrial projects related to, for example, manufacturing, transportation, and supervising and control, have adopted the agent metaphor. We find the reason for this being that such traditionally closed and clearly defined applications face the same demands and problems as those applications. They become larger and more complex, the demand for rapid change increases, and there is a pressure for integrating technical and administrative systems.

However, many domains may still be closed and well-defined. There may also exist well-proven systems for certain purposes, such as databases, industrial systems for real-time control, etc. These systems can very well be integrated with a multi-agent system by using the agentification technique.

REFERENCES

1. DA/DSM'96, *Proceedings of the DA/DSM Europe 96 Conference*, Oct. 8-10, Vienna, Austria, 1996. Available from PenWell Conferences & Exhibitions, PO Box 9402, 3506 GK Utrecht, The Netherlands.
2. Hägg S. and Ygge F., *Agent-Oriented Programming in Power Distribution Automation - An Architecture, a Language, and their Applicability*, Ph.L. Thesis, LUNFD6/(NFCS-3094)/1-180/(1995), Lund University, Sweden, May 1995.
3. 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.
4. Werner E., Logical Foundations of Distributed Artificial Intelligence, in O'Hare G. M. P. and Jennings N. R. (eds.), *Foundations of Distributed Artificial Intelligence*, John Wiley & Sons, ISBN 0-471-00675-0, 1996.
5. Castelfranchi C. and Conte R., Distributed Artificial Intelligence and Social Science: Critical Issues, in O'Hare G. M. P. and Jennings N. R. (eds.), *Foundations of Distributed Artificial Intelligence*, John Wiley & Sons, ISBN 0-471-00675-0, 1996.
6. Finin T., Fritzson R., and McKay D., et al., *An Overview of KQML: A Knowledge Query and Manipulation Language*, technical report, Department of Computer Science, University of Maryland, Baltimore County, USA, 1992.
7. Genesereth M. R., Fikes R. E., et al., Knowledge Interchange Format Version 3.0, Reference Manual, technical report Logic-92-1, Computer Science Department, Stanford University, 1992.
8. Hägg S., A Sentinel Approach to Fault Handling in Multi-Agent Systems, in *Proceedings of the Second Australian Workshop on Distributed AI*, in conjunction with the Fourth Pacific Rim International Conference on Artificial Intelligence (PRICAI'96), Cairns, Australia, August 27, 1996.
9. IBM IDAM V1 - IDAM Overview, *IBM Document IDAM-001-00*, available from IBM Sweden, IDAM department 5-40, PO Box 4104, S-203 12 Malmoe, Sweden.
10. Ygge F. and Akermans H., Power Load Management as a Computational Market, in *Proceedings of the Second International Conference on Multi-Agent Systems (ICMAS'96)*, Kyoto, Japan, pp. 393-400, AAAI Press, Menlo Park, CA, USA.
11. Genesereth M. R. and Ketchpel S. P., Software Agents, in *Communications of the ACM, Vol. 37*, pp. 48-53, 1994.