International Insurance Traffic with Software Agents

Chris van Aart[1] and **Kris Van Marcke**[2] and **Ruurd Pels**[3] and **Jan Smulders**[3]

Abstract. This paper presents a business case on digital cross-border information flow within a European network of insurance companies. Underlying the business case is the multi-agent paradigm. The paper explores a structured approach for designing agent behavior based on the 5 Capabilities (5C) agent model. The central value of the 5C model is the conceptual separation of concerns. It breaks apart five different dimensions of capabilities upon which an intelligent agent must draw. The work shows that functional as well as technical constraints can be reflected in an intuitive manner along these five dimensions. The outcome of the business case was the KIR system, a network of information exchanging agents handling green card insurance traffic within Europe. The strength of the agent paradigm combined with the simplicity of the application design acted as an eye-opener; with two major consequences. First, the customer has given the green light to develop and implement the KIR system at European level. Second, the customer has become a strong believer in the value of solutions in the insurance domain based on intelligent agents.

1 INTRODUCTION

Green card traffic is the process where companies are exchanging data for handling car accidents involving parties from different countries. Every country has a national green card bureau responsible for handling international car accidents, which is delegated to commercial insurance companies. 17 insurance companies in Europe form an international network of claim handling business called Euphoria. Claim handling works as follows: suppose a Dutch driver gets involved in a car accident in Germany. The accident is reported to a German insurance company, in this case, R+V in Wiesbaden (D). To settle the case, R+V will contact its Dutch partner, i.e. Interpolis in Tilburg (NL). R+V will open a new file containing information related to the incident and the involved partner. For this R+V has to contact Interpolis to verify whether the Dutch party is insured and covered. After exchanging details of the accident, the two bureaus will settle the case, determining who has to pay the costs.

The European Commission has recently enacted the so called 4th guideline: *Fourth Motor Insurance Directive (Directive 2000/26/EC)* -operational from February 2003 that obliges all EU insurance companies to execute and settle insurance claim submissions within three months after the date of accident. If they do not, they receive a penalty as high as the total amount of the costs. Costs range from an average of 6,000 EURO for only material damage to 100,000 EURO for physical injury. The time needed and personnel costs involved are not calculated in these amounts. Interpolis is confronted with 3,500 cases involving international insurance takers each year. The average settlement time per case is approximate six months, involving four to six contacts at different times and dates between foreign insurance companies. The reasons for the settlement duration are due to the internal bureaucratic process of the insurance companies. Furthermore, humans do most of the information processing.

The problem is that all systems (back-offices) used by insurance companies are heterogeneous, in the sense that data is stored and used differently. Information between these companies is exchanged by hand, meaning that claim handlers within the network communicate largely by telephone, fax and mail. The first alternative suggested was to develop a central database in Brussels, where all companies upload their data, and where every company can retrieve data. One disadvantage of this approach is that every company has to make mappings from its back-office data to this central database including a synchronization mechanism. The largest objection however is that companies have access to data of other companies, which can be used for other purposes. For example, one company could start to contact customers of other companies, offering their services. Therefore, the system should offer an arms length relationship between the involved parties: *"you can ask me questions, but you cannot have access to my information"*. The second alternative was to give web-based access to every individual back-office. One problem here is that the definition of a single interface would lead to endless discussion on topics such as: in what language should it be? What functionality should it have? Furthermore, not all companies are able or willing to submit to a single technical implementation of the interface, and a lot of companies are not ready to be on the Web. The greatest drawback however would be that, although access to information is possible, the results still need to be transferred manually between back-offices. The third alternative was to take an agent based approach. Every company can connect to a network of information exchanging agents.

The KIR system[4] has been developed by Acklin using the agent metaphor, because it provides a natural and flexible way to reason about distributed heterogeneous components, processes and coordination [1]. Business logic is encoded into the agents, on the one hand to assure the most fluent throughput of the process at stake; and on the other hand to respect the main business requirement, i.e. confidentiality. Furthermore, an agent-based system is far easier to extend in terms of functionality than a classical solution.

This paper is organized as follows, Sec. 2 introduces the 5 Capabilities model. Sec. 3 discusses the approach to the business case followed by Sec. 4, which describes the resulting KIR system. Conclusions are drawn in Sec. 5.

[1] Department of Social Science Informatics, University of Amsterdam, Roetersstraat 15, 1018 WB Amsterdam, The Netherlands, Email: aart@swi.psy.uva.nl

[2] KPMG Consultants, Bourgetlaan 40, B-1130 Brussels, Belgium, Email: kris.vanmarcke@kpmg.be

[3] Acklin B.V., Parkstraat 1a, 4818 SJ Breda, the Netherlands, Email: info@acklin.nl

[4] KIR stands for KBC, Interpolis and R+V, respectively Belgium, Dutch and German insurance companies of the Euphoria network.

2 FIVE CAPABILITIES MODEL

The *5 Capabilities (5C) model* is a conceptual framework for analyzing and designing the capabilities and functionality of an intelligent agent [7]. It defines five dimensions of agent intelligence - using the notion of *separation of concerns* - where each dimension plays a role in the development of intelligent software agents. It is designed in order to understand and be able to explain the added value of agent technology to software engineers and business managers. Besides the use of the 5C model for the intelligence of an individual agent, it applies most notably to agents operating within a multi-agent system. One of the inspirations of this model is the metaphor of an individual agent as an information processing brain, where functionality emerges by combining different specialized elements, i.e. models that cooperate [4].

2.1 Five Dimensions of Agent Intelligence

An agent in the 5C model is separated into five distinct dimensions: communication, competence, self, planner and environment. The dimensions are framed into models. Each model is responsible for one particular kind of capability an agent requires by having specialized functions and knowledge. The *communication-model* is responsible for handling all interactions between agents and other systems. It has parsing and encoding methods using knowledge of message transportation, representation and interpretation. The *competence-model* contains the methods and knowledge that enables an agent to execute the tasks it is designed for. The *self-model* gives the agent an idea of what the agent is doing (e.g. what are its tasks, goals, jobs, states, competences, etc.). The *planner-model* enables an agent to autonomously decide how to spend its time. It contains various planning strategies for meeting the agent's goals. The *environment-model* finally, gives the agent a view on the world it operates in (e.g. which other agents and systems it can interact with). How each of the dimensions will eventually be given shape may vary a lot depending on the particular kind of agent or the particular application.

2.2 Internal Message Flows

Each model in the 5C model can be seen as a process collaborating with the other processes through internal messaging mechanisms, which are standardized cooperation patterns forming the glue between the models. The implementation of each of the five models is open to the particular application. To show how agent behavior is managed through the cooperation of the five internal models we describe a typical internal message flow. This flow is called the *Incoming-Questions-Flow* as illustrated in Fig. 1. It starts when the agent is requested - for example by another agent - to perform a task (e.g. to look up a file). When the agent receives a REQUEST-message, the following flow is started: `communication` is responsible for handling interaction between agents, receives the REQUEST-message. After checking the syntactical validity of the message and whether this agent is indeed the intended receiver, `communication` asks `environment` to verify whether this agent is authorized to ask questions. For this, `environment` maintains a list of authorized agent references. If `self` does not recognize the content of the message it will ask `communication` to return a FAILURE-message. Otherwise, it detects whether the message is a response to an existing conversation or not, based upon which it will either define a new job and give it to `planner` or ask `planner` to resume an existing job. When this is a new request, `planner` will create a new job and forward it to `competence`. `Competence` selects the appropriate method for the job and performs it. When the job finishes, `competence` will ask `communication` to reply with the appropriate result by starting the *Outgoing-Answers-Flow*. The *Outgoing-Questions-Flow* is for getting information, support or cooperation from other agents. Furthermore, the *Incoming-Answers-Flow*, starts after receiving an answer to a question the agent has sent before.

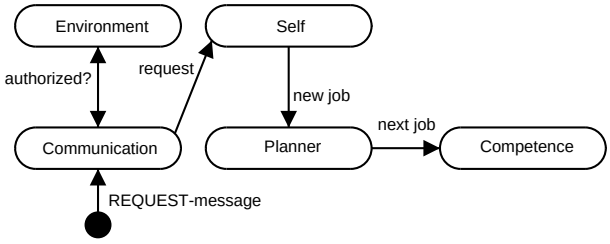

Figure 1. Incoming Questions Flow showing the cooperation between the five internal models of the 5C model when receiving a REQUEST-message.

2.3 5C Prototypes

The 5C model has been the design guide for the development of a series of intelligent agent application prototypes. The central value of the 5C model in these prototypes is the conceptual separation of concerns. The development of the applications showed that functional as well as technical constraints can be reflected in an intuitive manner along the five dimensions. Depending on the requirements of the application one can focus on each capability that needs attention, without loosing oneself in the complexity of the entire design. Prototypes that have been developed according to the 5C design guide include: *Supply in e-retail*: Special orders placed through internet to a retail-chain are delivered within hours to the nearest shop. The *Intelligent Freight Planner IFP*: Requested by the European Commission (DG7): a distributed transport planning application was implemented to help organizing intermodal freight transportation processes [6], and *International claim handling with the KIR system*: A multi-agent application that runs across different insurance companies to facilitate cross-border claim handling. The latter case is discussed in the next sections.

3 APPROACH

On the level of the business case we had to deal with the following functional constraints: (1) *No transparency in the market*, so it should not be possible that agents can query other agents' databases to retrieve data without a case, i.e. ensuring the arms length relationship; (2) The agents should have a high level of *robustness*, i.e. when the system or a part of the system should go down for some reason, it should go up without any problems, and go on with the tasks it was doing at the moment of the crash; and (3) The agents have to *operate within time windows* and should have *startup* and *shutdown procedures*. The back-office systems of Interpolis are operational from 6 a.m. to 11 p.m. from Monday till Saturday. All systems are started up and shut down via batch processes. The reason is that during the down period of the systems, maintenance can be performed, new systems can be installed and hardware can be replaced.

From a software engineering point of view we had to deal with the following technical/political constraints: (1) The agents should

work with *existing infrastructure* of Interpolis and its 16 partners in Euphoria; and (2) The agents can have *no direct access to the back-offices*. The IT-department of Interpolis did not want to have an exotic piece of software, like agents ”touch” its systems, because they do not have control over it.

To meet these concerns and the constraints of the business case we applied the following approach. First, the process wherein companies have to cooperate to handle car accident settlements was analyzed. Second, the activities that the agents should perform are mapped on agent behavior and agent communicative acts framed in agent collaboration diagrams. Third, an interface was designed to enable the agent to have (controlled) access to the required functionality, like finding, retrieving, creating and updating records in the database of the back-office. Finally, all pieces are put together in the KIR system which is presented in Sec. 4.

3.1 Green Card Traffic

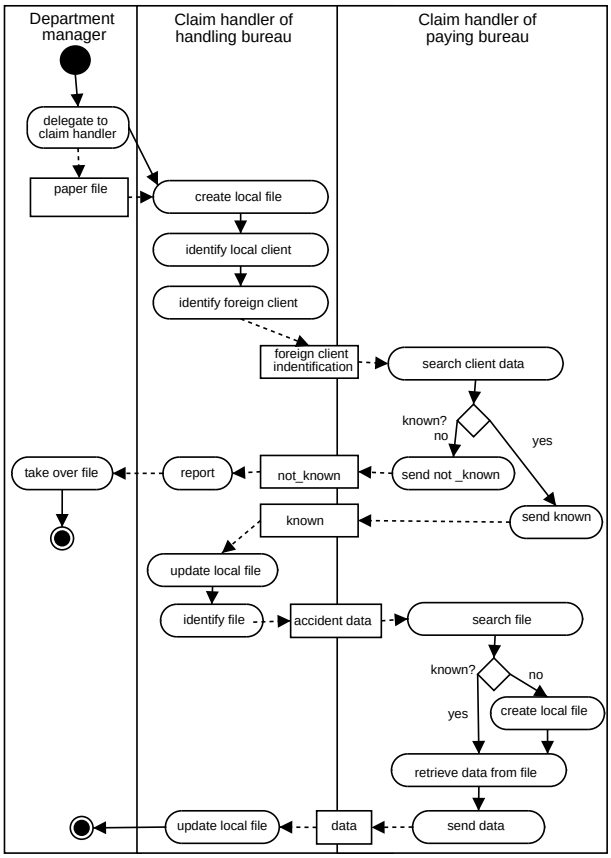

Figure 2. Green Card Traffic Process illustrated in UML activity Diagram. The swim lanes represent what actor does what jobs. The rounded boxes show the jobs (activities), the square boxes show the resources, the straight lines show the direction of the sequence flow (control) and dotted lines show the direction of the resource flow (data flow).

The interaction between insurance companies starts after the report of damage caused by a car accident involving parties from different countries. The assignment of the handling bureau and paying bureau role in case of an accident involving drivers from different countries, is done with the following rule. The company located in the country where an accident takes place, is the *handling bureau*. The company located in the country of the foreign driver, is the *paying bureau*. The handling bureau will start the settlement and will send/start handling traffic to the paying bureau. The other case is that the accident takes place in another country, where the company is the paying bureau and responds to the handling traffic with paying traffic.

The process starts when the manager of the foreign claims department receives a report via one of the various channels including call centers, mail or fax. This report includes basic information (license plate, policy number and date of the accident), damage forms, police reports and witness declarations. The manager will delegate it to one of the claim handlers. The claim handler will open a new file in the claims database and starts identifying the involved parties. First the local party will be identified, using the green card number and license plate number. Next the foreign party has to be identified, checking whether this party is known and whether the information is consistent. For that it will contact its foreign partner, i.e. the paying bureau, sending green card number and license plate number of their insurance taker. If the party is not know, the claim handler will report this back to the manager who will end this process and starts another process that we will not discuss here. If the party is known, the handler will update the local file and will ask its partner whether the case is known. If the case is known at the partner it will send the date and its local claim number (database primary key) including all dedicated information to the handler. If the case is not known, the partner will create a new file and will also send the new local claim number to the handler. A part of this process is illustrated in Fig. 2, drawn as a standard UML activity diagram.

3.2 Agent Collaboration

We mapped the green card traffic process on an agent collaboration diagram in AUML introduced by [5]. The Green Card operation are illustrated in Fig. 3 showing the handling and paying role and their pattern of interactions.

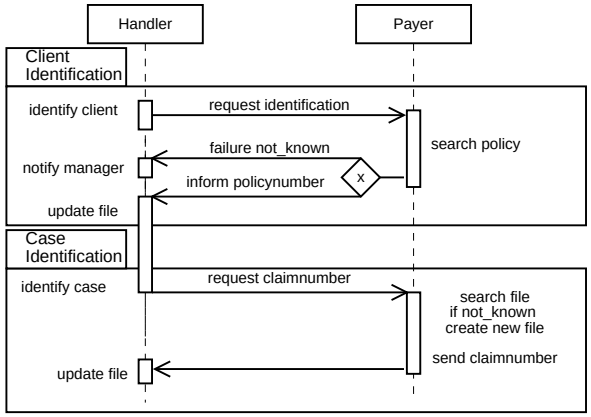

Figure 3. Green Card Traffic operation design in AUML collaboration diagram showing patterns of interaction with the operation. The dotted lines represent lifelines of position. The arrows show interactions and the packages show the applied agent interaction protocols.

Two packages show the two main processes: (1) client identification using green card number and license plate and (2) case identification, using policy number and local claim number. A package shows the applied agent interaction protocol (AIP) for enabling the cooperation between the agents, where an AIP describes communication patterns as an allowed sequence of messages between agents and

the constraints on the content of those messages. Here existing AIPs are placed in sequence to enable the process as described in Sec. 3.1. The AIPs used are all based on the FIPA REQUEST-protocol[5].

Communicative acts (CA) make up the process between a handling and paying bureau, and replace the interaction between humans by way of speech and handwriting into communication by agent as illustrated in Fig. 2. The idea is that the manager of the department delegates the two identification tasks to the agent instead of a claim handler. The handling and paying bureau are here called *handler* and *payer*. It starts with a REQUEST-message for identification of the client. The identification contains a license plate and green card number. The payer validates the identification and can response with: (1) a FAILURE-message containing *not_known*, which means that the client is not known. In many cases this is caused by typical errors such as data entry errors in the license plate or policy number as fed by the paper reports; or (2) a INFORM-message containing a policy number, meaning that the payer has identified the local insurance taker. Then a REQUEST-message for identification of the claim is to be send, with this the handler asks for all known data from the file of the payer. The payer can response with: a INFORM-message containing a policy number, which means that the payer has either created a new file with the data and locally know data, or has already created a file for this case. The latter can happen when the insured party has already registered this accident, before the handler asks for it. In both cases the payer will send a claim number, which is the key to the file of the accident.

3.3 Interface To Back-Office

There are a number of ways to give access to legacy systems. The main concern of the insurance companies is the security of their data and the stability of their back-offices. For that we build a transducer in the notion of [3]. This transducer maps instructions from both agent to the back-office and results from the back-office to the agent. This approach has the advantage that the agent does not have knowledge of the back-office, only of the transducer. A more important advantage is that when the agent has to give access to other back-offices, only the back-office side of the transducer has to be altered. The transducer at Interpolis has access to a separate database where instructions and reports are written and read by the agent. In this way the agent only has indirect access to the back-office, which in the case of Interpolis is build in Powerbuilder using a Sybase database.

The transducer between the agent and the back-office enables the agent to execute a number of actions: (1) `validate identification` using green card number and license plate number, (2) `search file` by policy number, (3) `create file` using policy and license plate number, (4) `retrieve data from file` using policy number, and (5) `update file` using policy number and received data.

4 KIR SYSTEM

The integration of the agents and the transducer results in the following architecture as illustrated in Fig. 4. As shown every insurance company has installed one agent with a specialized transducer that is able to route handling traffic and paying traffic. This architecture provides not only a communication medium for the designer and builder at Interpolis and Acklin, but also the end user of the system, the foreign claims department of Interpolis. The metaphor of actual communicating and reasoning entities using back-offices (also called virtual employees) helped to explain the architecture of the system. Furthermore, it aided in showing how and where design choices were made in respect to the functional and technical constraints.

The KIR agent is a specialization of the 5C model and its accompanied message flows as described in Sec. 2. The communication model is instructed with the AIPs as described in Sec. 3.2 meaning that it validates incoming messages from other agents. If it does not understand or expect a message, it will send a NOT_UNDERSTOOD or FAILURE-message back to the sender. The self-model has knowledge of the two roles of handling bureau and paying bureau. The environment-model holds a list of agents, i.e. the agents of 17 partners in the Euphoria network that are authorized to interact with the KIR agent. Furthermore the competence-model has access to the transducer, from where it can retrieve and update information from the database of the back-office. The *Incoming-Question-Flow* and *Outgoing-Answer-Flow* are specialized into paying traffic. As soon as the KIR agent receives a REQUEST-message it knows that its role is payer in this conversation and will react accordingly. The *Outgoing-Question-Flow* and *Incoming-Answer-Flow* are specialized into handling traffic.

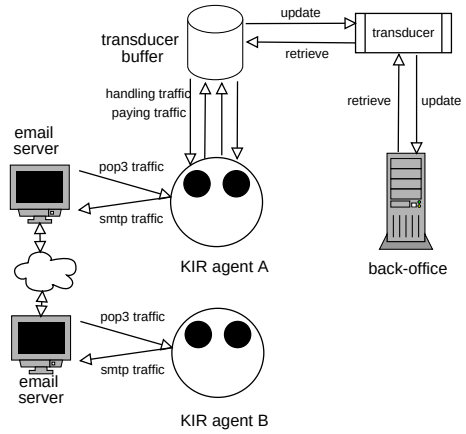

Figure 4. The KIR architecture, showing two KIR agents and the coupling to the mail server and the transducer.

4.1 Operationalization

The IT-department of Interpolis developed the transducer between the database of Interpolis and the agent within 30 days. The KIR agent was built in Java in less than 60 days. The models are implemented as Java thread objects with asynchronous mail box semantics meaning that every model is a separate computational process with own control and has a mail box with which it communicates with other processes [2]. The mailboxes are implemented as databases with records representing incoming messages. Besides that every variable is also stored in a model state database. Forwarding a message from one model to another means that the sending model adds a record to the mailbox database of the receiving model. This ensures that when the agent or a model (i.e. Java thread) goes down, the state can be restored. Every action of a model is logged in a log file, for both maintenance reasons and tracking and tracing of flows within the agent. The use of databases ensures robustness and enables the requested daily startups and shutdowns. The arms length relationship constraint is handled in the environment-model and the self-model.

[5] http://www.fipa.org/repository/ips.html

The environment-model will filter out non-authorized messages, using a (hard coded) list of authorized agent email addresses. The self-model has a set of rules for alerting a claim handler, such as *"when an agent will ask for more than ten cases in the hour"* for identifying the querying of the database without an actual case, *"when an agent ask for more than three not existing cases or policies"* and *"when an agent send a messages that cannot be read"*, for identifying possible hackers.

The insurance companies made the choice for e-mail as means of message transport between the agents. The reason was that e-mail functionality is present at all companies. Another option was the use of middleware, such as CORBA, but the state of the technology at several insurance companies prevented this. The format of the message is in a frame based like syntax for expressing feature-value pairs. Strict security is not applied to keep the hurdle of implementing the agency at a minimum. The system is relatively secure through the strict application of the format of subject and content, the small number of e-mail addresses in the system and the fact that return addresses in e-mails are not used.

The final implementation of the KIR system did not use an existing agent framework, such as JADE[6], mainly because a large number of features in JADE were unnecessary for the KIR system. The functionality was too specific to use a general framework. Furthermore, existing frameworks at the time were not industry-strength. This means that there were no mechanisms for handling robustness, startups and shutdowns, and logging. Many of the existing frameworks rely on synchronous connections over TCP/IP. Insurance companies are very hesitant to let third parties make such connections from outside their internal networks, and some might even not be able to do so. The least common denominator used by every partner in the agency is e-mail. The introduction of the KIR system, immediately resulted in a work pressure release of three people at Interpolis and reduced the process of identification of client and claim from 6 months to 2 minutes. The 2 minutes here are an estimate of the time needed to send a number an agent messages using email. The bottleneck is the 'polling' time of the involved email servers and the availability and latency of intermediate mail services.

4.2 Extension

The separation of concern principle introduced in Sec. 2 appeals when we further refine and extend the application. For instance, the EU directive dictates that the procedures should not exceed the period of 3 months. If we want the agents to act consciously according to this principle, we need to refine the agent by providing it with more explicit goals (i.e. the goal to succeed with a procedure within 3 months) and the ability to reflect upon its own performance (i.e. to be conscious about how well it achieves its goals). Both functions can be realized by refining the agent's self-model. If we also want the agent to be able to act when it observes that its goal is not met, we can then again extend the competence-model to give the agent alternative methods for executing a job more rapidly. If alternatively we want to extend the payer role such that they can distinguish between different types of handlers, for instance handlers from insurance companies with whom there exists a special agreement to handle the claim in a less complicated manner, we can refine the environment-model of the agent in order for it to be able to make this discrimination. In that case we also need to refine the agent's competence-model as it needs to have knowledge of the alternative handling procedures as well. When adding an extra service (task or goal) to the agent, only the competence and self-model have to be changed.

[6] http://jade.cselt.it

5 CONCLUSION

This work presented how an industry-strength system can be analyzed and designed using the 5C model and techniques from the agent paradigm, taking into account functional and technical constraints. The 5C model enables the designer of a software agent to focus on aspects of the software agent's intelligence separate from the rest of the agent's behavior and implementation. The KIR system showed that for the design of the two agent roles - the payer and the handler - the differences in competence is solely located in the competence-model and self-model; all of the other models are identical. In order to install an agent at a insurance company, only the transducer to the back-office had to configured so that the agent has access to the right functionality and information. Furthermore, the authorization table of the environment-model has to be filled properly.

The strength of the agent paradigm combined with the simplicity of the application design acted as an eye-opener; with two major consequences. First, the customer has given the green light to develop and implement the KIR system at a European level, as described in this paper. Second, the customer has become a strong believer in the value of solutions in the insurance domain based on intelligent agents.

The 5C model enables to pinpoint intuitively where in the agent design the desired extension has to be made, resulting in a clean and easy to understand agent implementation. Ultimately, we expect the 5C model to become the basis for an agent architecture and framework. Through the different application prototypes that we designed we are stepwise generalizing the behavior of our agents into different agent prototypes, which can be simply instantiated for novel purposes. We expect to be able to define agent prototypes that represent different roles that an agent can play in an organizational (or social) model.

ACKNOWLEDGEMENTS

This work has been partially funded by the IST project IBROW, nr.-1995-19005. We would like to thank Peter van Hapert and Joost van Dijk (foreign claims department, Interpolis), Taco Leemans and Will Swinkels (IT department, Interpolis), and Patrick Storms, Kees Zorge and Leo Blommers (Acklin).

References

[1] A. Bond and L. Gasser, *Readings in Distributed Artificial Intelligence*, Morgan Kaufmann publishers Inc.: San Mateo, CA, USA, 1988.
[2] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns*, Addison Wesley Reading, MA, 1995.
[3] M. Genesereth and S. Ketchpel, 'Software agents', *Communications of the ACM*, **37**(7), 48–53, (1994).
[4] G. Morgan, *Images of organization*, Sage Publications, 1996.
[5] J. Odell, H. Van Dyke, and B. Bauer, 'Extending UML for agents', in *Proc. Of the Agent-Oriented Information Systems Workshop at the 17th National Conference on Artificial Intelligence*, eds., G. Wagner, Lesperance Y., and E. Yu, (2000).
[6] C.J. van Aart, K. Van Marcke, and L.N. Eriksen, 'Agentbased Logistic Sevice Provision', in *Proc. the Fifth International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM 2000), London*, (2000).
[7] K. Van Marcke and C.J. van Aart, 'The Five Capabalities Architecture For Intelligent Agents', *in submission*, (2002).