WARSAW UNIVERSITY OF TECHNOLOGY

Faculty of Mathematics
and Information Science

Master's Thesis

Computer Science

Agent-based Commodity Flow Management

Author:

Tomasz Serzysko

Supervisor:

Dr Marcin Paprzycki

Warsaw, January 2007

Signatures:

Supervisor	Author
.....	.....

Agent-based Commodity Flow Management

Abstract

In today's business, information is crucial to success. Having enough information and being able to make proper use of it brings increased efficiency, which results in greater savings, and faster reaction time which in turn allows to capitalize on opportunities and thus increase income. This is why businesses invest in Enterprise Resource Planning systems.

Software Agent technology brings new approach to modelling such decision-making systems. An agent as a concept can be easily related to a human employee performing a specific task within the company's decision chain. This and other agent features should allow designers of ERP systems to implement traditional business methodologies in a computer system without much effort.

This thesis will investigate this by designing and implementing an agent system for commodity flow management. The system will manage supply levels in a warehouse, and by using demand forecasts it will communicate with wholesalers to collect their offers and pick the best one to restock ahead of time, so that the warehouse will not run out of supplies. This system will provide logistics support for the existing agent-based e-commerce system.

Inteligentni agenci programowi w zarządzaniu przepływem towarów

Streszczenie

Na dzisiejszym rynku informacja jest kluczowym czynnikiem w drodze do sukcesu. Odpowiednie jej użycie pozwala firmie zwiększyć wydajność i skrócić czas reakcji na zmiany, co pozwala podnieść oszczedności i przychody. Jest to powodem dużej popularności i dynamicznego rozwoju systemów Planowania Zasobów Przedsiębiostwa.

Technologia inteligentnych agentów jest nowym podejściem do tworzenia takich systemów wspomagających podejmowanie decyzji. Ideę agenta można łatwo odnieść do roli ludzkiego pracownika w firmowym łańcuchu decyzyjnym. Ta i inne cechy systemów agentowych powinny pozwolić twórcom systemow PZP na łatwe przeszczepienie tradycyjnych metodologii biznesowych na grunt systemów komputerowych.

Niniejsza praca bada tę hipotezę przez zaprojektowanie i wdrożenie systemu agentowego do zarządzania przepływem towarów. System zarządza stanami magazynowymi i na podstawie prognoz zapotrzebowania zamawia z wyprzedzeniem potrzebne towary u dostawców, aby utrzymać stan magazynu pozwalający na zaspokojenie oczekiwanego popytu. System ten będzie wspierał istniejący system e-commerce, również zbudowany przy użyciu technologii inteligentnych agentów programowych.

Contents

List of Figures

All figures are originals, with the exception of Figure 22 (source: [8]).

1 Introduction

1.1 Business Intelligence

The scientific approach to solving common problems in business is called business intelligence. Here “intelligence” means gathering information valued by its currency and relevance, as well as information itself. Parallels between the military and business are clear, as success in both depends on:

- available resources
- collecting data
- recognizing patterns and drawing meaning from collected data (generating information)
- making decisions based on generated information.

Just as in war, the quality of management of available resources is crucial to achieving success, especially when resources are scarce, or the resources commanded by one's competitor(s) are far greater than those of one's own. Having sufficient information one can even attempt to predict enemy troop movement (or market trend changes) so that decisions can be made before a particular situation emerges. Such feats of leadership were often regarded as divine inspiration in the past by the ignorant.

Today we recognize the value of information as the most important resource, having called our time the Information Age. However this resource was very hard to harness. In the beginning, the problem was the lack of proper means of gathering required

data. After that proper communications were missing, and so available data could not be collected before decision had to be made. When telegraph, telephone and radio became available, the data started flowing in such vast amounts, it created a problem to properly analyse, to extract useful pieces of data out of the stream of incoming information.
Therefore the value of intelligence was often overlooked, as information was unreliable, incomplete, or just plain outdated, and in critical moments decisions were made using intuition.

The explosion of Information Age was sparked by very recent (relatively speaking) development of computers, which are ideally suited to processing large amounts of data. Supplied with appropriate software, they provided the information upon which decisions could be made quickly and reliably. By now usage of specialized software packages for gathering, processing, storing, managing and presenting the information aiding in making business decisions – commonly referred to as Enterprise Resource Planning (ERP) systems – is very common in serious corporations. In 2004 the value of ERP software market was estimated at $23.6 billion and had grown steadily at a rate of around 10% each year [11].

1.2 The Next Step: Agents

While computers are not enough to make decisions themselves, as good decisions seem to require more than just cold calculation, it is possible and feasible to allow them do simple, day-to-day mundane tasks. This way people using the aid of such systems would be free to focus on more important matters. A simple example is a stock broker using searching and filtering software to find information relevant to the shares currently managed by the broker on the news feed he is receiving. This describes the idea behind a *user agent*, or more generally: *agent*. Shortly put, agents

are pieces of software that are intended act on their user's behalf to perform tasks the user would otherwise have to do themselves.

The concept of agent can be traced back to Hewitt's Actor Model first introduced in [15] published in 1973, in which agent was described as *"a self-contained, interactive and concurrently-executing object, possessing internal state and communication capability"*. This aptly describes the whole notion of agency, as though there are as many definitions as researchers in the field, there is a common agreement that an agent should have the following characteristics:

- *persistence* – code is not executed on demand but instead runs continuously and decides some activity should be performed on its own,
- *autonomy* – after being set up and launched, agents have capabilities of task selection, prioritization, goal-directed behaviour, decision-making without human intervention,
- *social ability* – agents are able to interact with other systems (possibly other agents) through some sort of communication and coordination, they may collaborate on a task with friendly agents and compete against foreign agents,
- *reactivity* – agents observe and understand the context in which they operate and react to changes appropriately.

If one was to consider *intelligent agents*, one would additionally expect them to adapt (examine the environment and change their behaviour when they decide it benefits them) and learn (analyse results of their actions and improve to achieve results they expect).

As one can see, agents as a model are designed to act in a "human" way. That is why a structure of a multi-agent system is relatively easy to understand, as responsibilities and interactions seem very natural. However it is not that easy. Although the concept of an agent is very appealing to human mind, because of the agents' purpose of imitating human behaviour, it is not the way computers work. Traditional computer

programs tend to be procedural instead of event-driven, stateless (as they do not carry over their state between executions), rather uncooperative in their autonomy and linear in behaviour.

Therefore, in order to implement a true agent system, one needs a kind of middleware, a layer that would translate the language of agents into a language of computers. While preparing such agent system, it would be also useful to allow independently developed agents to communicate effectively to enable the creation of networks of cooperating agents; and that would require agreeing on a single standard. Currently there are several interest groups working in the field, one of them being Foundation for Intelligent Physical Agents (FIPA) [4], which created a set of protocols governing agent interactions. Compliant middlewares are also in development, with the most important being Java Agent DEvelopment Framework (JADE) [5] which simplifies agent development by providing libraries for Java language and implements protocols put forth by FIPA. Both FIPA and JADE will be discussed in later parts of this paper.

1.3 Agents as ERP

Applying the concept of agent system to the purpose of ERP seems very straightforward. In essence the tasks done by such systems were originally done by people. As agents are supposed to relieve humans of doing boring, repetitive or just calculation-intensive and otherwise not too engaging work it is natural to think of using them in this role.

The benefit of utilizing agent approach for designing ERP system is that it is as powerful as a traditionally developed system and at the same time easy to understand. This feature comes from the fact that, when using agents, each of them can be treated

as if it was a company's employee. What follows from that assumption is that we can naturally assign roles and responsibilities to an agent. This greatly improves the understanding of the system, as it resembles old-fashioned “human-only” companies. So while agent systems may require some additional work for implementing agents themselves – which can be circumvented by using one of pre-made agent platforms and standard protocols – they are an advantage when designing the system, as knowledge and experience of traditional business can by still be capitalized on here.

1.4 ERP for Logistics

One of business processes that ERP systems can assist in is Supply Chain Management (SCM). It is the process of planning, implementing, and controlling the operations of the supply chain with the purpose to satisfy customer requirements as efficiently as possible. The term was first coined in 1982, but the definition was formulated by Council of Supply Chain Management Professionals [16] in 2004. It suggests that although the idea is not new, it has become a field of professional research only recently. What is interesting about SCM, is that it is part of every business, no matter its size or field of operation.

The following strategic and competitive areas can be used to their full advantage if a supply chain management system is properly implemented:

- **Fulfilment**. *“Ensuring the right quantity of parts for production or products for sale arrive at the right time.”* ([1] p.46). Efficient communication ensures that orders are placed with the appropriate amount of time available to be filled. A SCM system also allows a company to constantly see what is on stock and making sure that the right quantities are ordered to replace stock.
- **Logistics**. *“Keeping the cost of transporting materials as low as possible consistent with safe and reliable delivery.”* ([1] p.46). The SCM system enables a

company to have constant contact with its supply and distribution team. The system can allow the company to track where the required materials are at all times. Also transport cost optimizations may be made, e.g. by sharing transports with a partner company if a single shipment is not large enough to fill a whole "unit of transport". SCM can monitor and report such situations and thus give insight needed for performing such optimizations.

- **Production**. *"Ensuring production lines function smoothly because high-quality parts are available when needed."* ([1] p.46). Production and sales can run smoothly as a result of fulfilment and logistics being implemented correctly. If the correct quantity is not ordered and delivered at the requested time, production or sales will be halted, but having an effective SCM system in place will ensure that parts, raw materials or products for sale are always available in time.
- **Revenue & profit**. *"Ensuring no sales are lost because shelves are empty."* ([1] p.46). Managing the supply chain improves a company's flexibility to respond to unforeseen changes in demand and supply. Because of this, a company has the ability to produce goods at lower prices and distribute them to consumers quicker than companies without supply chain management thus increasing the overall profit.
- **Costs**. *"Keeping the cost of purchased parts and products at acceptable levels."* ([1] p.46). SCM reduces costs by "increasing inventory turnover on the shop floor and in the warehouse" (i.e. by reducing costs of storage and amount of products disposed because of exceeded expiry date), controlling the quality of goods thus reducing internal and external failure costs and working with suppliers to produce the most cost efficient means of manufacturing a product.
- **Cooperation**. *"Among supply chain partners ensures 'mutual success.'"* ([1] p.46). One of the newest trends in the field of SCM is Collaborative Planning, Forecasting and Replenishment (CPFR) which is defined as a *"longer-term commitment, joint work on quality, and support by the buyer of the supplier's managerial, technological, and capacity development."* ([2], p.293) As usual, the

definition is confusing for the layman, but simply put, through such cooperation the company has access to current, reliable information, and can therefore obtain lower inventory levels, decrease lead times, increase product quality, improve forecasting accuracy and finally improve customer service and overall profits. The competing suppliers benefit mutually from the cooperative relationship through increased buyer input from suggestions on improving the quality and costs and though shared savings. Such cooperation is also beneficial for the consumer as well through the higher quality goods provided at a lower costs.

Right now, due to globalization and prominence of multi-national corporations and also major improvements in communication technology, which led to a decrease in communication costs, market intelligence, cooperation, flexibility and quick reaction times are crucial to organizations' success. This causes companies to focus on their core competencies while outsourcing other activities (such as transport, production of parts, gathering marketing data) to other firms that specialize in those fields and should perform the task more time- or cost-effectively. Another popular way of optimizing the company's operations is using IT for gathering information and making more informed decisions.

That is where ERP for SCM steps in. As one can see from the list presented on previous pages, proper supply chain management gives many benefits to the company and coincidentally, it is a place where computer software systems can help in this respect. The dependency on good communications, fast responses, data collection, storing and processing make it ideal for improving with some kind of ERP.

2 Proposed Solution

2.1 Agent-based Commodity Flow Management

The objective of this master's thesis is to investigate the benefits that agent technology can bring into businesses by replacing or extending existing traditional systems. Having received briefing into business challenges and solutions in the shape of ERP systems in the previous chapter, in the following sections the reader will be presented an existing agent-based e-commerce system. Next, the idea of extending current functionality by adding warehouse management system will be discussed. One will examine the roles that agents can take in a system, observe the clear view of the system modules and responsibilities that agent approach brings to the table while designing the structure, and then the ease with which the model of a specific agent can be defined before proceeding to view more design details in the next chapter.

2.2 E-commerce System

The agent-based e-commerce system in question is the system that was conceptualized and is being developed by M. Paprzycki, M. Ganzha at al. Many papers describing the various aspects of the systems were published, one of the most general and comprehensive overviews of the system can be found in [3]. The description provided below is most of the time more general than those found in that and other published documents, except where it is directly connected to the subject of this thesis and therefore specificity is required for correct understanding of the proposed solution.

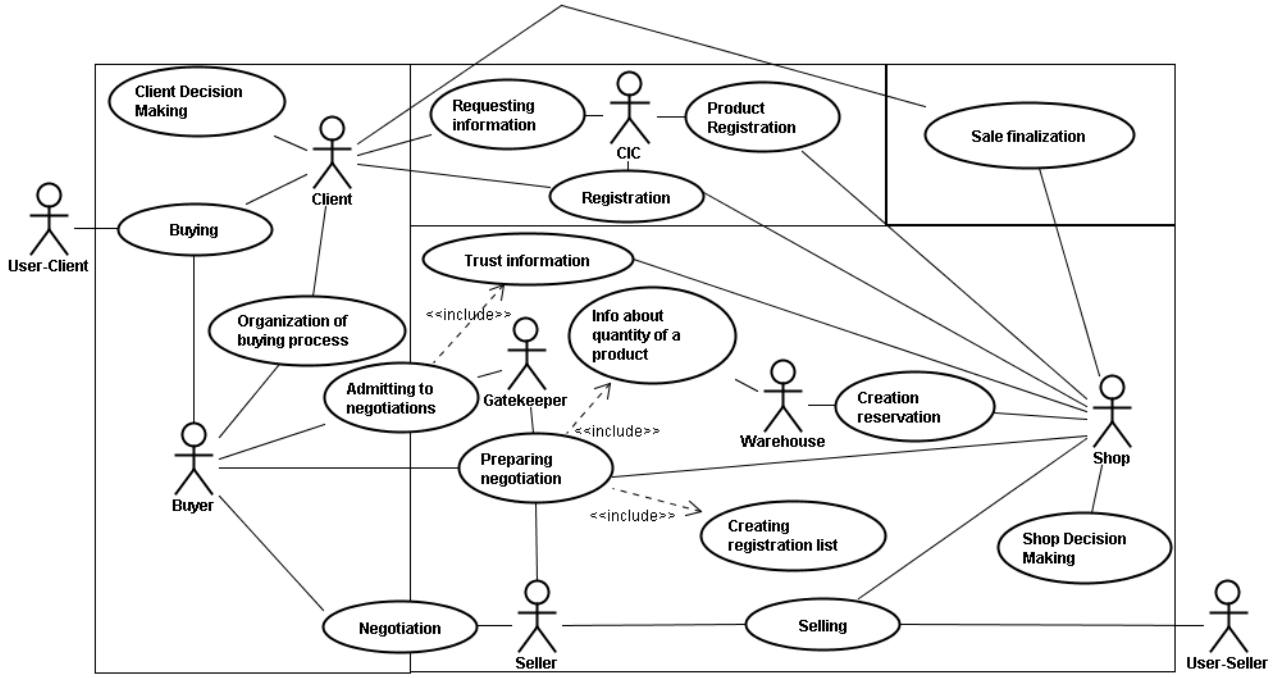

Figure 1: E-commerce system overview

The system (fig.1) is a virtual environment that acts as a distributed marketplace hosting shops (*Shop*) and allowing customers (*User-Clients* represented by *Clients*) to visit them and purchase products. Clients have the option to negotiate with the shops (*Buyer-Seller* relationship), to bid for products and to choose the shop from which to make a purchase. As another possibility, the shops may be approached by multiple clients in one time frame that wish to purchase an item which stock is limited (i.e. less items than clients) and consequently, the buyer is elected through auction-type mechanisms (coordinated by *Gatekeepers*).

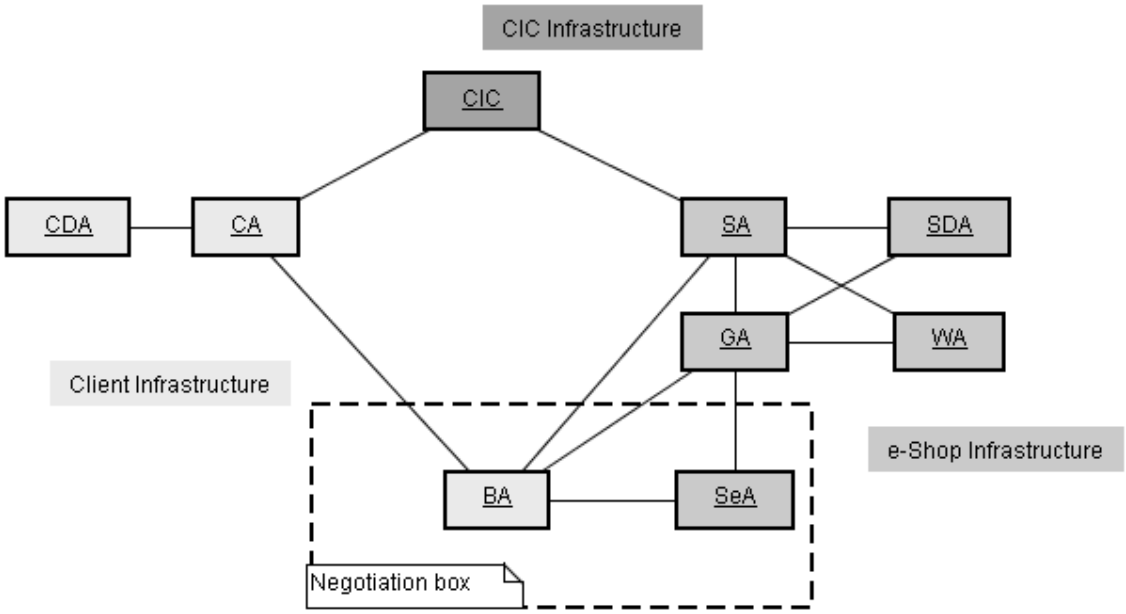

Figure 2: E-commerce - underlying agents

The trading is coordinated by Customer Information Centres (*CIC*), which serve as yellow pages where shops can advertise their services and clients may look for shops providing the services/products they need.

Each entity in the system (see fig.2) is represented by agents – in fact sometimes more then one, which reflects different roles those entities may perform. Lines connecting entities represent direct communication between agents. For instance, the shop entity consists of the following closely coupled agents:

- *Shop Agent (SA)* which represents the shop when registering to CIC and while initiating and finalizing sales with a client,
- *Seller Agent (SeA)* which represent a store in coordinated negotiations with multiple clients,
- *Gatekeeper Agent* (*GA*) which coordinates negotiations with multiple clients,
- *Warehouse Agent (WA)* which monitors stock levels and serves as database for availability queries,
- *Shop Decision Agent (SDA)* which acts as sales database for data mining and sales forecasting in the store.

An important characteristic of the system's concept: *"... we are more interested in creating an artificial agent world in which e-commerce agents perform variety of functions typically involved in e-commerce, rather than developing a particular e-commerce system targeted to solve a specific business problem that uses a limited number of application-specific agents. In other words, we are facing the multi-agent challenge from the very beginning."* [1]. As a consequence, the e-commerce system is a "pure" agent system, not hindered by the need to interface with traditional non-agent systems. This means a comprehensive homogeneous environment with all advantages of a agent approach as pointed out in previous chapter.

In defence against potential arguments of the pure agent system being an abstract concept, developed in separation from the real world, and therefore an unrealistic idea it should be noted that JADE agent platform allows for easy integration of JADE agents with traditional Java programs. This was investigated as a side project by the author of this thesis and will be described in more detail in following chapters. So what this "pure agent" approach does, it shifts the responsibility for providing a proper interface from the system to the system client, which is a usual practice when designing common frameworks.

The system in its current state focuses on trading between buyer agents and shop agents, the issues of presenting offers to the clients, finding shops selling products a client needs, and most of all engaging in, conducting and overseeing negotiations. However, the issue of how shops acquire the goods they sell, how they make sure that adequate supply exists in their warehouse is not addressed. Investigating possible approaches to solving those logistics problems – which are inherent in any shop – is the subject of this thesis.

2.3 Logistics in E-commerce

To provide logistics support to the e-commerce system, one first needs to analyse the needs of such a system, while also keeping in mind its purpose. It is clear the designer would want to provide a mechanism supplying the goods for the store to sell to clients, but one must consider the degree of "technicality" – or in other words – the level of abstraction the system will operate on.

As the stores are competing in the e-commerce market, which is characterized by lightweight, highly mobile, innovative, objective-focused and efficient businesses, it is typical for them not to have their own transportations department. What is meant by that is they do not have their own means of transporting goods (i.e. trucks, vans, planes, etc.) – they outsource the task of moving goods to specialized delivery companies, such as UPS, FedEx, DHL, to name only a few. Therefore the proposed logistics system will not engage in counting trucks, developing efficient parcel stacking algorithms or solving network optimization problems. It remains a responsibility of the contracting delivery firm to optimize its operations. What interests "our" system is the quote given by the carrier – how much, and how fast.

Another typical feature of e-commerce is that it is customary for the supplier to have a deal with one of the transports companies. What it means to "us" – from now on let us refer to the owner/designer of the system as "we" and to the system as being "ours", which will clearly set "our" system apart from "other" systems, to avoid confusion. Again, what this means to us, is that we have basically no control on the choice of the deliverer. This greatly reduces the number of decisions that must be made while placing an order to the supplier and receiving the goods afterwards.

Another consequence of this is that there is usually no separate cost of the handling

and delivery of ordered product. This cost is either included in the offered price, or is paid by the supplier (which in reality resolves to the receiver paying the cost – the supplier may be only more or less obvious about it). This fact frees our system from making additional calculations. Even if the supplier offers discounts on delivery costs – or to that end, possibly also on unit price – based on amount ordered, it is assumed that they will incorporate such discounts in their offers, so that our system will receive only a single figure.

The final limitation, comes not from e-commerce in general, but rather from the concept and design of our e-commerce system, in which advanced buying techniques, such as bidding and negotiating are implemented on the side of store-client interaction. For this reason, for the separation into areas of focus, and finally for the sake of clarity, we simplify the buying process by discarding the ability to bid and negotiate on the store-supplier side. The suppliers present their offers, and the buyer selects the one they think is the best. What is done here is actually still a kind of an auction – called a closed auction (where bids are not publicly announced).

The above assumptions allow the logistics system implemented as part of this thesis to concentrate solely on mechanisms for predicting commodity flow, setting up safeguards against supplies running out, decreasing the probability that shop profits are lost because of empty shelves, as well as reducing the amount of goods lying needlessly around. All this can be done without obfuscating the picture by details and activities which are not important from the point of view of e-store's “supply officer”, as they are but technicalities connected with the manner goods are transported.

2.4 Agents in Logistics

In order for our e-commerce system to perform aforementioned logistics tasks, several new roles have to be introduced into the system. Some of those roles can be given to existing agents, others are discrete enough to warrant adding new agents which will perform them.

The system needs agents to perform the following functions:

- *demand estimation* – the agent will be responsible for drawing information from sales data and/or external premises to somehow predict future sales of various goods.
- *warehouse monitoring* – this agent's responsibility will be to observe the shop's supply levels and react in case there is a risk of those levels dropping below quantities sufficient to satisfy estimated demand.
- *order management* – this role will require the agent to coordinate the issuing of orders for goods and also assisting in evaluating offers received by the agent doing the actual ordering, by utilizing the knowledge gathered while coordinating the orders to provide information on supplier reliability.
- *ordering goods* – the agent will contact the suppliers for their offers and will select the best offer, based on proposed price and opinion on the supplier provided by ordering manager.
- *selling goods* – this will be the representative of the supplier. As mentioned before, the goods bought by logistics system will be acquired in a simplified way, without negotiations, which requires the use of a new simple agent, similarly to the use of the agent ordering goods, instead of a full-blown existing customer/buyer agent (however in case of ordering agent the separation is also enforced by its additional functionality, not only simplification of trading process).

- *yellow pages* – this is another CIC-type role, which is however separated from the role of connecting fellow negotiators. It may be considered another simplification, but at the same time it is decision based on real life example. In reality, not all traders are willing – and what is more common – prepared to handle large scale transactions. The reasons may for example be lack of large storage areas, or poor logistics support. We also expect high reliability. From this comes the distinction of wholesalers and retailers. While we do not assume that the existing e-commerce system deals only with retail – although selling items by auctioning usually implies scarcity – the logistics system is most definitely wholesale. Therefore a clear separation would have to be placed in existing CIC between sellers capable and incapable of wholesale. And since the logistics system (shop-supplier) is otherwise quite independent of the e-commerce side (shop-client) it makes sense to implement another, similar in design, but different in function, type of CIC.

The *demand estimation* role will be attributed to the existing SDA. This is a natural choice, as the agent already collects sales data and deals with data mining (for example when generating sales reports, if not for any other purpose), so adding additional functionality is not a problem. On the other hand, however, a separate agent for this task would use the same data and procedures as SDA, either increasing the strain on the sales data warehouse or forcing the system to replicate the data for sole purpose of demand estimation. For the purposes of the logistics system, SDA would additionally generate advice for warehouse monitoring role. The nature of how that advice would be produced to be most effective is the topic of one of later sections of this document.

The *warehouse monitoring* role is already part of the existing WA. Until now, however, the role of this agent was quite passive, as its task was merely responding to stock level queries and watching the supplies decrease. Unless they were “magically”

restored – which is quite a fitting description, as magic is something that does not exist – just like a supplier in the system until now. This time WA will be given the ability to act instead of just watching. Upon receiving the prediction from SDA, the agent will examine if there is enough in stock to satisfy projected demand – if not, WA will issue order request (paradoxically sounding at first) to the agent filling the order management role. WA will also notify the order manager when a delivery arrives – this will help the agent to gauge the reliability of suppliers.

The *order management* role is the first role so far to introduce a new agent. This agent will be subsequently called the Logistics Agent (LA). Its responsibilities were outlined before, and currently there is no agent that could perform them – as they are strictly connected to the logistics system. This agent will be the intermediary between the WA and the agents buying goods from the wholesalers and it will use that position to collect data on supplier reliability, which will be a major factor – the other one being, as usual, the price – in deciding on our supplier. LA will be described in more detail in following chapters.

The agent *ordering goods* (OA) will be also a new one. Its task will on one hand be quite simple, without the requirement of participating in negotiations, on the other – crucial for the performance of the system, as selecting the right supplier by proper balancing of honest price, adequate delivery time and acceptable level of reliability will influence how visible the benefits of employing an intelligent agent SCM system over simple mechanisms such as “buy when there is room on the shelves” or “buy when shelves are empty” will be.

The agent *selling goods* will be most likely the simplest agent to be added, as it is something of a stripped down version of the smart negotiating agent. Its codename will be WhA for wholesale Agent.

Finally, the *yellow pages* logistics CIC agent, which will be the same in design as the CIC in the e-commerce part of the system. Its role is simple but effective, and also quite important in bringing the OA (through LA) and WhA together to potentially close the deal.

UML use case diagram (fig.3) summarizes the roles of the new and the extended agents.

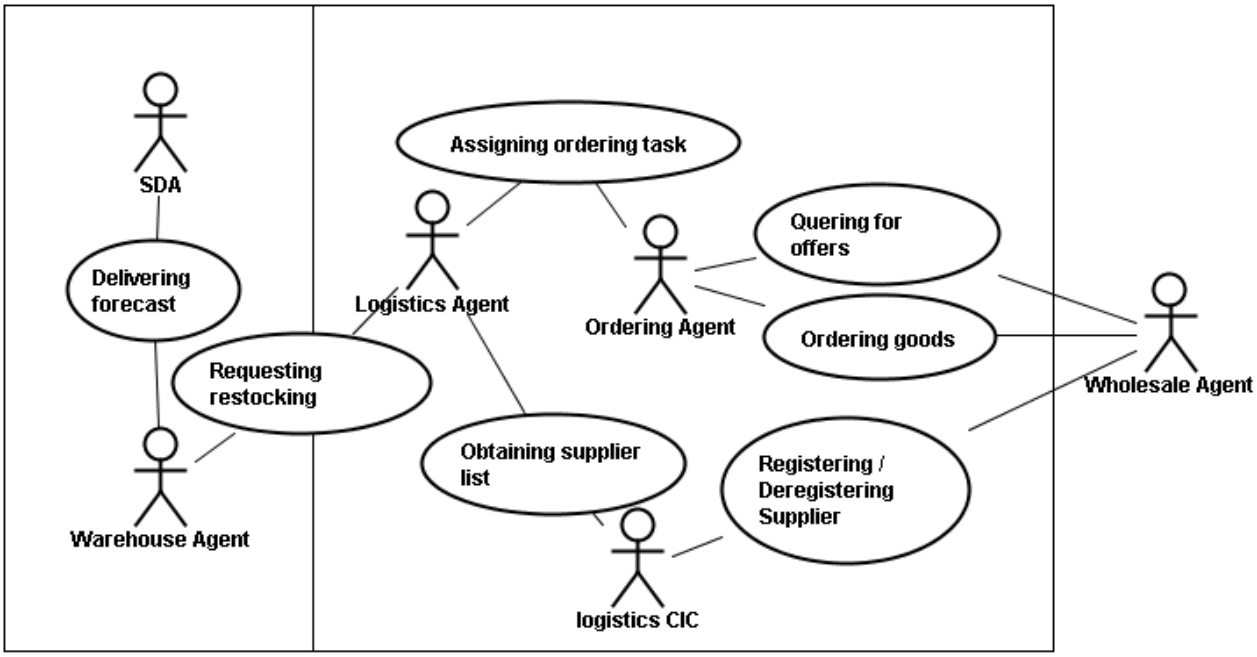

Figure 3: Logistics agents - use cases

3 Design

3.1 FIPA Agents

The idea of an agent was a revolutionary concept. However for any technology to become more than a researchers' toy and enter the business market, ruled by risk and profit, good support from wealthy and established players is required. Other than that, nobody invests their money in projects without a good business plan. For a technology, the formal specification plays that role, which serves as an outline of what the technology can be used for and how.

Foundation for Intelligent Physical Agents [4] was established in 1996 in Switzerland as an independent body to create software standards specifications for heterogeneous and interacting agents and agent based systems. Since that time it played a crucial role in the development of agents standards and has promoted a number of initiatives and events that contributed to the development and uptake of agent technology. Many of the ideas originated and developed in FIPA are now coming into sharp focus in new generations of Web/Internet technology and related specifications.

Currently FIPA has 32 members, most of which are commercial organizations and universities. Of those, names like Boeing Corporation, Nippon Telegraph & Telephone, Rockwell, Siemens and Toshiba should be familiar even to people not tied with IT. On June 8th 2005, FIPA was officially accepted by IEEE as its eleventh standards committee. This signalled the move of standards for agents and agent-based systems into the wider context of software development. In short, it was officially

recognized that agent technology needs to work and integrate with non-agent technologies.

In 2002 FIPA completed a process of standardising a subset of all its specifications, which now contains 25 specifications that made it to standardisation stage. The specifications describe various categories of agent technology: agent communication, agent transport, agent management, abstract architecture and applications. The core category of all these is agent communication as communication lies at the heart of the FIPA multi-agent system model, and the concept of agents in general.

Usage of the specifications while designing an agent system is crucial, as it allows for robust agents whose behaviour in terms of external communication is predictable and thus allows them to communicate with other, heterogeneous agents. As heterogeneous we understand agents that: are performing different roles, work within a separate environment, on other platforms, written in other programming languages. Several of the conditions may hold true for any pair of agents. However, for a system to be effective, communication must be possible, while internal workings of the agents may differ.

The proposed solution adheres to FIPA rules to ensure smooth communication even among “homogeneous” agents internal to the system. Some specifications will be mentioned in following sections while describing specific agents and their interactions, governed by documents such as *FIPA Contract Net Interaction Protocol Specification* ([12] SC00029). In case of such specifications as *FIPA ACL Message Structure Specification* ([12] SC00061) and *FIPA Communicative Act Library Specification* ([12] SC00037), their knowledge is imperative at every step of development of any agent. Fortunately, compatibility with most of the other standards is handled “under the table” by the chosen agent platform, library and middleware, JADE. Still, being aware of them is helpful in proper understanding of agents.

3.2 JADE Framework

Java Agent DEvelopment Framework [5] was first released in 2000 by Telecom Italia Lab, which is a R&D department of Telecom Italia Group, focused on *"promoting technological innovation by scouting new technologies, carrying out and assessing feasibility studies, and developing prototypes and emulators of new services and products"*. JADE was made available under Lesser GNU Public License (LGPL) open source license in February 2000 (version 1.3), and soon thereafter Tilab formed JADE Governing Board with Motorola to promote the evolution and the adoption of JADE by the mobile telecommunications industry as a Java-based de-facto standard middleware for agent-based applications in the mobile personal communication sector. Since then JADE has been actively developed by the community under the auspices of the Board and its current version at the time of this writing (23-11-2006) is 3.4.1 released on November 17th.

Let us review the features of JADE that benefit a potential agent programmer. First of all, it is a Java library consisting of classes which are representing FIPA prototypes, e.g. ACLMessage, modelled in accordance to *FIPA ACL Message Structure Specification* ([12] SC00061). This gives a solid base when implementing agent system, as most of the tasks such as implementing an agent communication model over the network are performed by the JADE building blocks – the developer needs only to put them together and then provide intelligence to the agent by programming the internals of its Behaviours (which are classes that represent agent "actions").

To illustrate this mechanism, suppose that we want to create two agents, personal Alarm and public MessageBoard. MessageBoard is used to post messages from users. We want Alarm to notify us when a message that might interest us arrives (for

instance with our name in it). For this to work, Alarm should periodically ask MessageBoard to show it new messages, and look for our name in the text.
While simple in concept, if not for JADE, we would have to write the agents from scratch. Run them as daemons listening on TCP/IP sockets for any messages, agree on details such as contents of protocol header which would distinguish our messages from others, specify one encoding for message contents and then the format of the message itself. Imagine the difficulty of trying to extend our Alarm to work with other MessageBoards written by others. And this of course does not even touch the issues of communication reliability and of course we assume that exact contact details of agents we are trying to connect is known from the start.

JADE simplifies all of this by providing for the Alarm the *Agent* class which can have associated *CyclicBehaviour* that communicates by sending *ACLMessages* to the MessageBoard which in turn runs more advanced *REResponder* behaviour which responds to *ACLMessages* of type *FIPA_REQUEST* (refer to *FIPA Communicative Act Library Specification* – [12] SC00037). We only customize the behaviours by telling Alarm what to ask in the message about, and MessageBoard how to answer to such message. This sounds simple, but one may wonder how exactly are the messages transmitted. And what is more interesting – how agents find each other.

This is solved by the other thing JADE provides to the user: the JADE Agent Platform. The platform is a complete environment for agents to live in, including a message transport system, which agents tap into when they want to send a message to a local or remote (i.e. sitting in another agent platform, different from our platform) agent and yellow pages service for local agents. The platform also allows the user to monitor the state of agents and to add and remove agents.

By providing those two components required for implementing agents, JADE frees the programmer from dabbling in technicalities and allows him to focus on the

intelligence in Intelligent Agents. This means setting up proper Behaviours and programming their contents, and giving meaning to messages sent between agents by describing *ontologies*, which are again supported by JADE's *jade.onto* package.

The concept of ontology may be unfamiliar to the reader. For now it is enough to say that ontology (from Greek *ontologia* – "the study of being") is a data model that represents a domain and is used to reason about the objects in that domain and the relations between them. In other words, it is used inside the agent and in communication between agents as a form of representing some knowledge about the world. For more information on ontologies, along with a description of ontology used in the proposed solution, see section on ontology in the Appendix.

After this introduction to tools used in constructing agent systems, I will now proceed to discuss the implementation details in scope of the logistics agents.

3.3 New System Structure

In comparison to the existing structure, 4 new agent entities were added, and functionality of 2 other agents were extended. The following diagram (fig.4) shows where those new agents fit in the picture (existing unmodified agents are darker, new agents are shown in pale, existing agents with new functionality are shown in half-tone; note the new connection between SDA and WA).

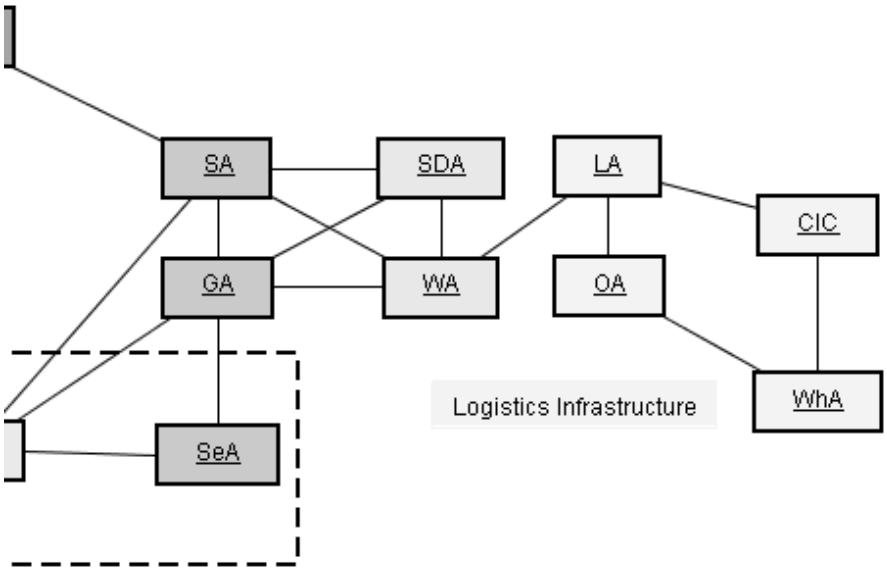

Figure 4: Logistics agents within existing structure

As seen on the diagram, the part of the system dealing with already existing functionality is unchanged, which was one of intended goals of the proposed solution. New agents and functionality will be described in detail one by one in following sections. Right now let us review features universal to all parts of the new logistics system.

The ontology used by new agents is *LogisticsOntology* which describes concepts used by agents while coordinating their tasks. As an example of concepts described by the ontology, let us review the action of registering a WhA in the CIC. For this, *CICRegister* action is used, which contains registration data in the form of *CICAgentDescription*. This piece of data is required to have *agentAID* property, containing a single AID (globally unique Agent Identifier) of the agent willing to register and *offeredProducts* property describing the products the agent is offering for sale, containing at least one *Product*.

Product is only superficially defined in *LogisticsOntology*, as from the logistics point of view, the system needs only to identify required products (i.e. be able to decide

whether the product needed in the warehouse is in fact the same as the product offered by a given supplier). That is why the only property of *Product* in *LogisticsOntology* is *productName*. The incorporation of a name is more of a gesture towards humans that might work with the system. For the agent, any obscure number, e.g. hash value of the *Product* instance is enough. However, in production environment (a real agent system working "in the field" not being built for research) *Product* would be typically expanded by importing into *LogisticsOntology* a domain ontology suitable for the product sold. Such ontology would describe the products from the domain more specifically, for example: cars ontology (engine, number of doors, fuel consumption, etc.), book ontology (author, title, genre, publisher, etc.), grid resource ontology (architecture, RAM size, processor, OS, etc.), hotel ontology (star rating, room types, leisure facilities, etc.) and anything else one can imagine that needs describing.

More technical details on *LogisticsOntology* is available in the Appendix. Now let us proceed to the descriptions of particular agents, that are part of the logistics solution. Please note that full logistics system use case diagram was shown in fig.3. Following use case diagrams show only use cases related directly to the discussed agent.

3.4 Consumer Information Centre

As mentioned earlier, CIC concept is basically the same as in the existing system. The only difference are agents using it and *LogisticsOntology* employed by the CIC. CIC serves as yellow pages knowledge base and advertisement board for wholesalers and buyers looking for a specific item. The diagram (fig.5) shows the interactions between CIC and its environment.

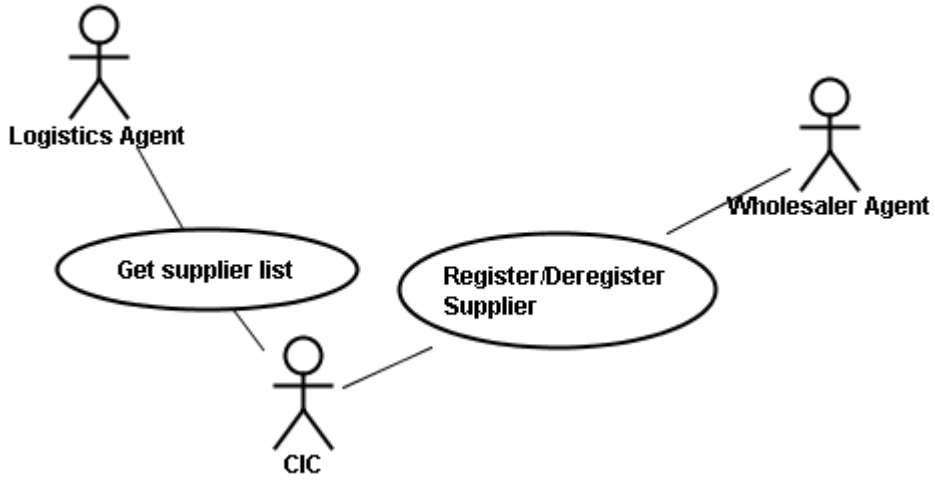

Figure 5: CIC use cases

As we can see, agents interacting with CIC are WhA and LA. WhAs register to offer their products, and LAs later query CIC database for that information. More details on those interactions in sections on respective client agents.

CIC of the logistics system stores the data as *CICAgentDescription*. While optimizing data storage and retrieval is outside the scope of this thesis, for the sake of completeness, let us consider how such data can be stored. This depends on the purpose and size of the system:

- *memory* – for very small system where persistence of the data is not required, such as concept testing and ontology development. This simple solution (no need to manage external data sources) is currently employed in this system.
- *XML* – small / research systems, where expected number of clients is small, and / or peak update and query performance is not required.
- *Database* – large systems, where there are many clients and a lot of data needs to be stored and accessed quickly.
- *Advanced Database Solutions* – for high number of clients and peak performance while serving multiple queries per second. For more details on this solution, see [6].

Following diagram (fig.6) describes CIC internal state changes.

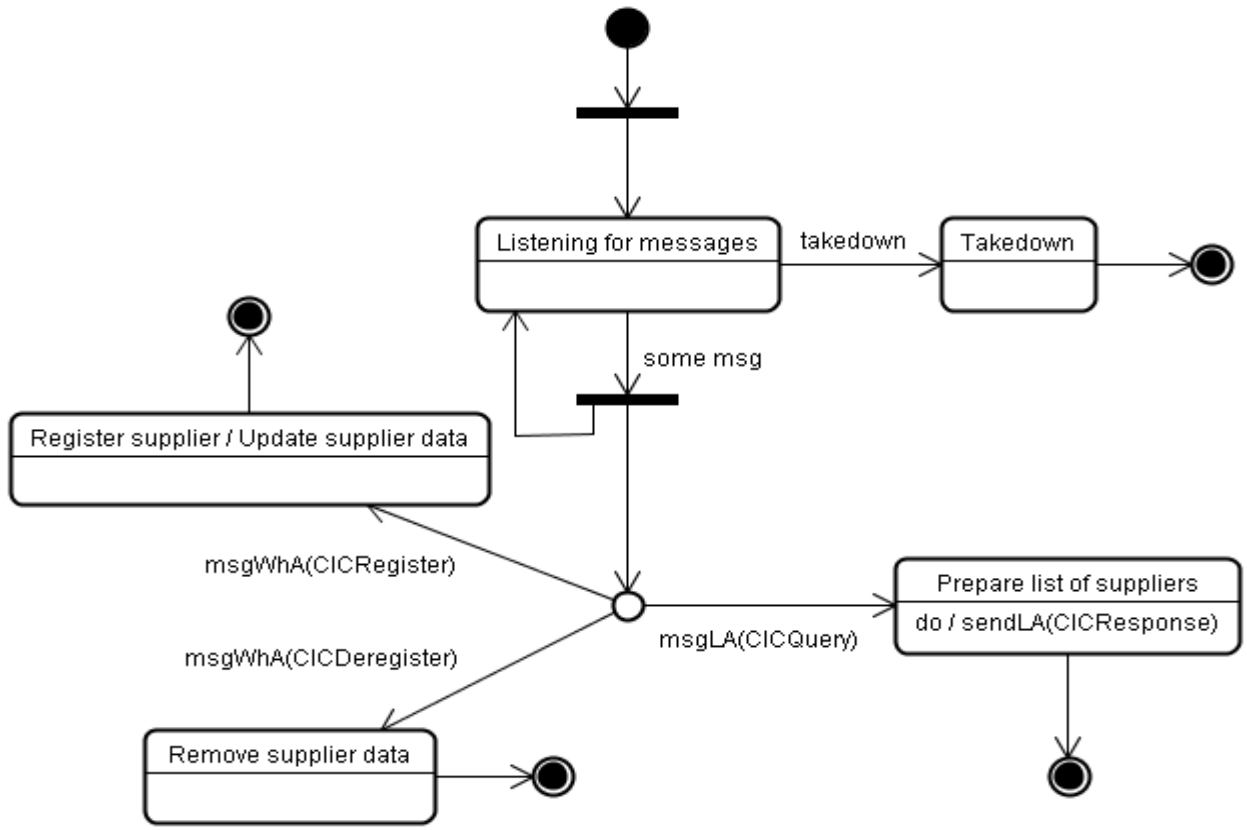

Figure 6: CIC state transitions

3.5 Wholesale Agent

The diagram (fig.7) shows the interactions between WhA and other agents in the system.

WhA communicates with CIC to register its services and OA to sell goods.

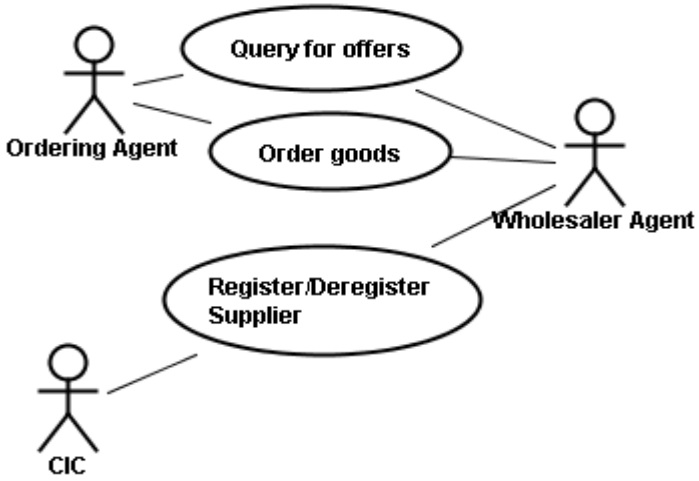

Figure 7: WhA use cases

When WhA enters the logistics system with the intent of selling goods, it contacts the CIC agent to register itself with its range of offered products. When CIC is queried about a product claimed to be sold by some WhA, it returns that (and possibly other) WhA's *CICAgentDescription* in answer to the query. As a consequence, WhA is contacted by OA to present the details of its offer and possibly to do business. When WhA shuts down, it is expected that it will contact CIC again, to request removal of its data, so that CIC doesn't send outdated information. The sequence diagram for WhA – CIC interactions is shown in (fig.8).

Communication between OA and WhA closely follows the defined *FIPA Contract Net Interaction Protocol Specification* ([12] SC00029). For more information on this interaction see OA description below.

The diagram (fig.9) describes WhA internal state changes.

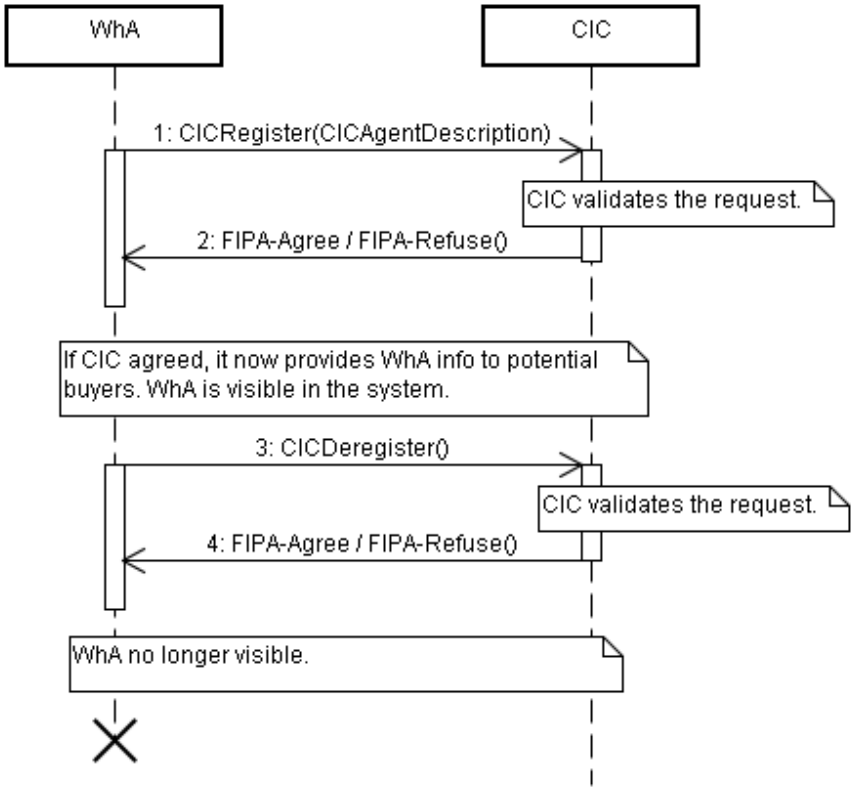

Figure 8: WhA-CIC interaction sequence

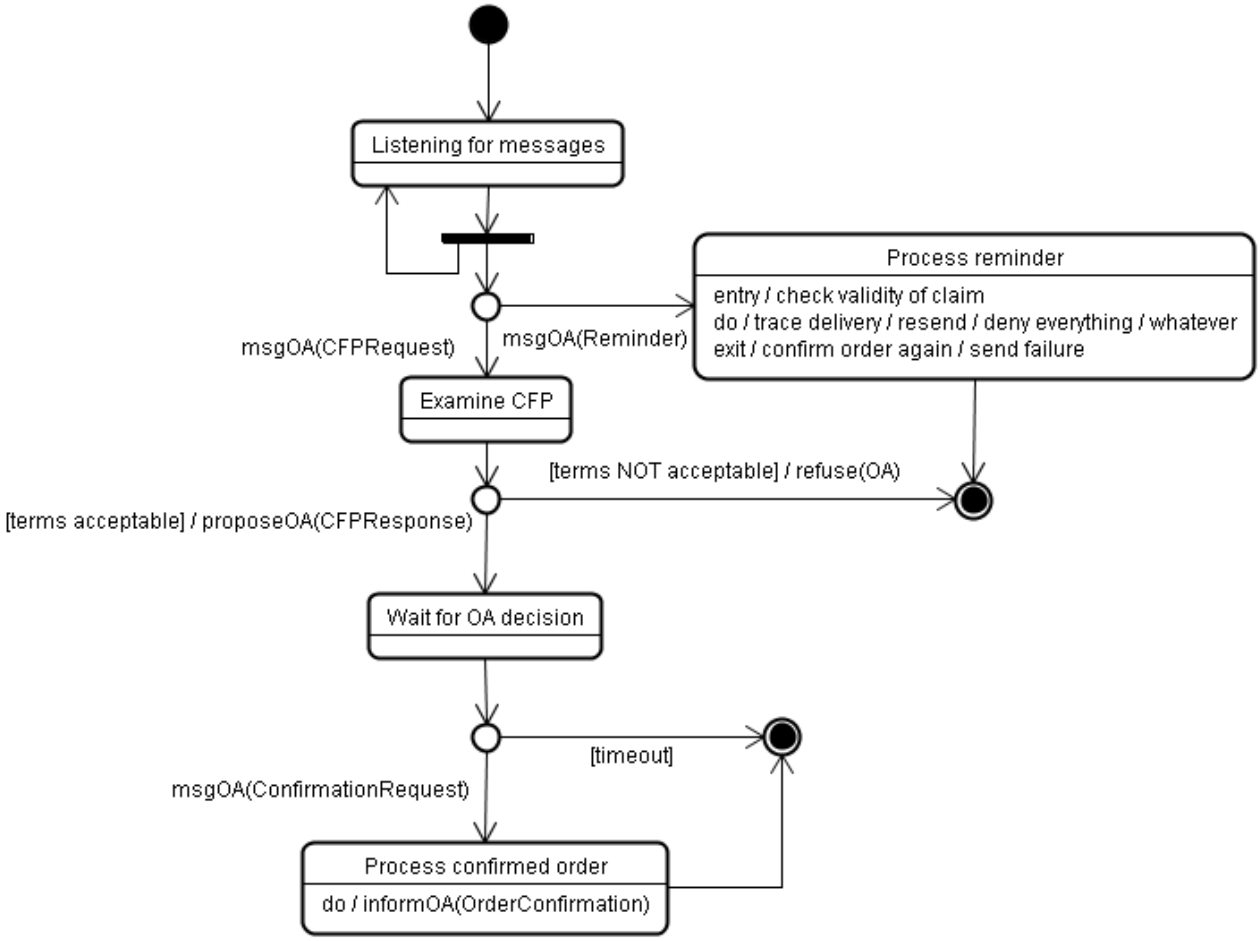

Figure 9: WhA state transitions

3.6 Ordering Agent

The following diagram (fig.10) shows the interactions between OA and other agents in the system.

OA communicates with WhA to buy appointed goods, and LA to update it on the status of the order. It also receives instructions and updates from LA.

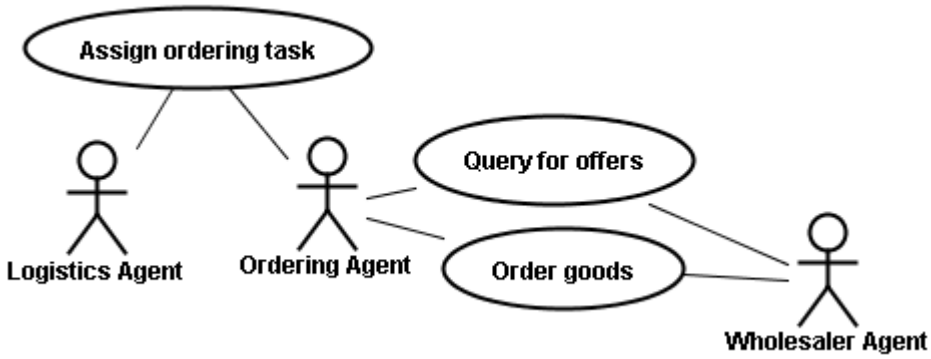

Figure 10: OA use cases

The process of purchasing goods from WhA is performed according to *FIPA Contract Net Interaction Protocol.* The diagram (fig.11) describes the sequence of actions in that interaction.

As we can see, the protocol is initiated by CFP (call for proposal) from OA, which WhA responds to by stating its terms (amount, price and delivery time). Offers from all WhAs that OA has contacted are processed internally and after ranking the offers (the factors are price, delivery time and WhA reliability) winner is contacted to confirm the order. In case of no response or refusal from the winner, runners-up are contacted in an iterative process.

Cooperation between OA and LA is a close one. LA has a pool of OA at its disposal. At any time LA requires an OA to perform some action, it contacts the first OA that is not busy. The diagram (fig.12) is a sequence of their interactions.

LA "wakes up" the OA by providing it with order details and the list of WhAs received from CIC together with their reliability rating that LA computes. This information is used to send CFPs and then properly filter and rank the offers received. When an offer is confirmed, LA is notified of this and the "monitoring phase" of the ordering begins. At this time OA is free to return to the pool. If delivery associated

with the order is not received in due time, LA “wakes up” any free OA, and OA issues a reminder to the winner.

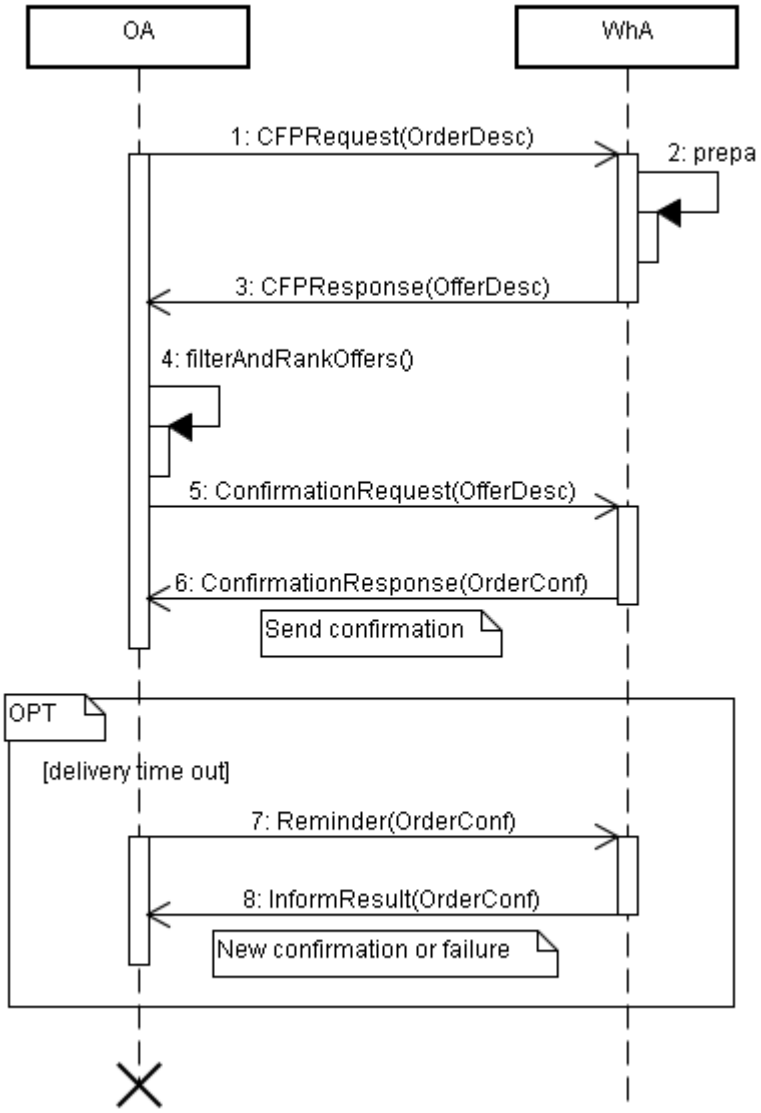

Figure 11: OA-WhA interaction sequence

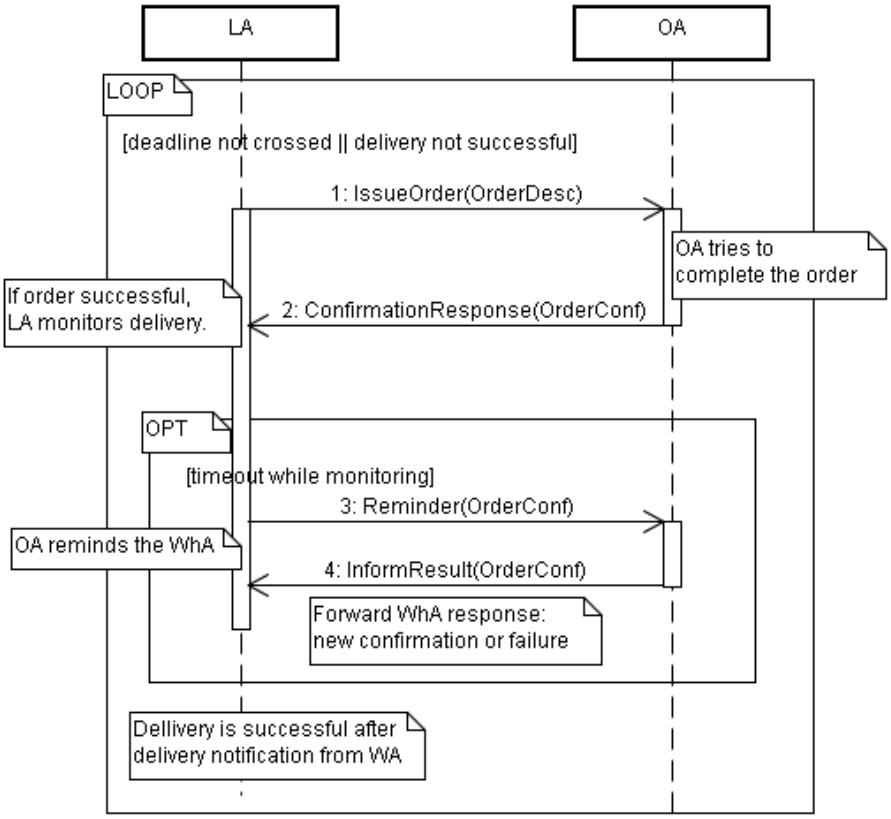

Figure 12: LA-OA interaction sequence

More details on LA role can be found in the next section. Diagram (fig.13) describes state transitions in OA.

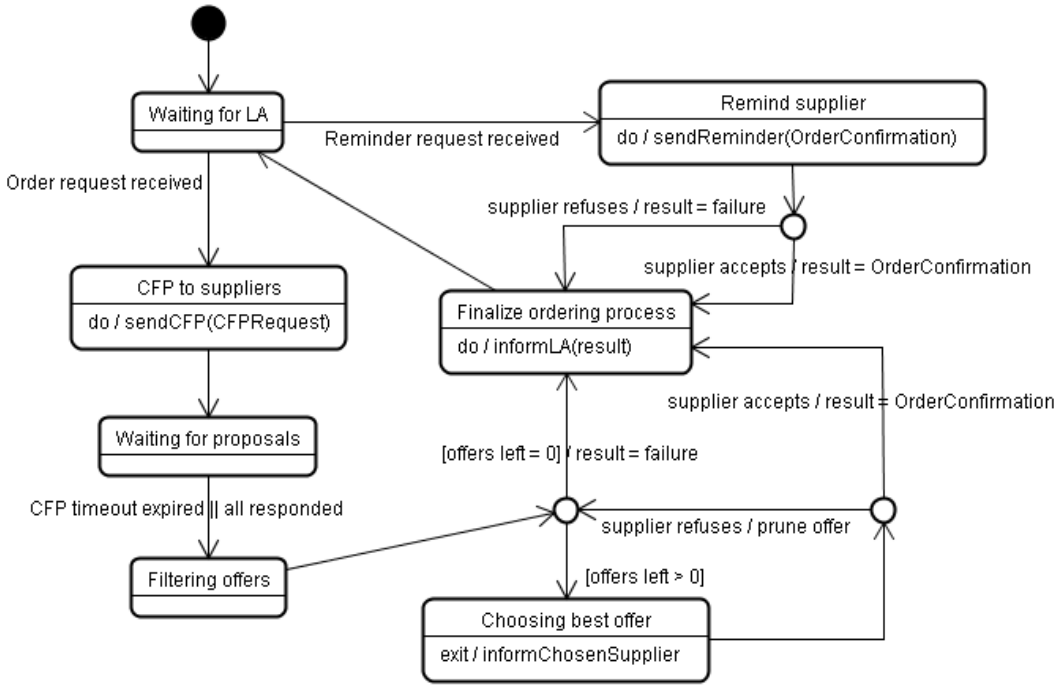

Figure 13: OA state transitions

3.7 Logistics Agent

The diagram (fig.14) shows the interactions between LA and its neighbouring agents (i.e. those that it has direct contact with).

LA communicates with WA while receiving order requests and delivery confirmations and also reports failed deliveries it discovers, as well as it designates and sends instructions to OA and closely monitors order progress. LA also contacts CIC to retrieve the list of WhAs that can deliver required goods.

Interaction between WA and LA is carried out using simple *FIPA-Request* and *FIPA-Inform* protocols and will be more closely described in the section devoted to WA.

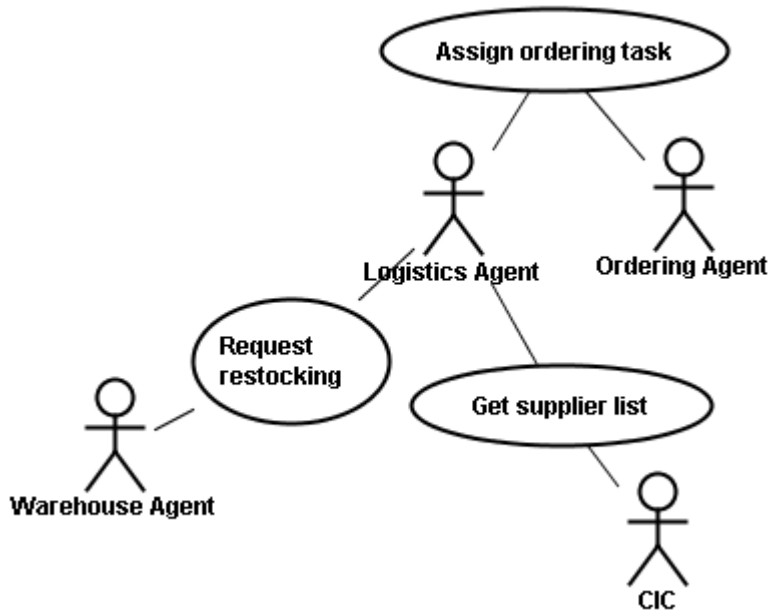

Figure 14: LA use cases

LA serves as ordering coordinator. Having multiple OAs at its disposal, it delegates order requests received from WA to them, while providing them with data required to accomplish the task. One piece of such data is the list of required product providers. LA contacts the CIC using *FIPA-Query* protocol to obtain such list and then retrieves information for each supplier from its reliability database (used database option is in-memory for simplicity; reliability score reflects our previous dealings with a given supplier – for more information, refer to the section 5.5). In case such information is not present (unknown supplier) LA associates with it a default reliability value (see section 5.5 for a discussion on reliability management strategies). Such aggregate data is provided to the OA.

LA monitors orders by receiving *orderId* from the OA. Information on arriving deliveries received from WA contains this information and this allows LA to identify whether the order was fulfilled or perhaps the delivery was somehow lost. Depending on this, LA decides if delivery was successful, or might call OA again to issue reminder to the supplier or to try to find another supplier. Finally it may declare

delivery failure.

The complete LA state transition diagram below (fig.15).

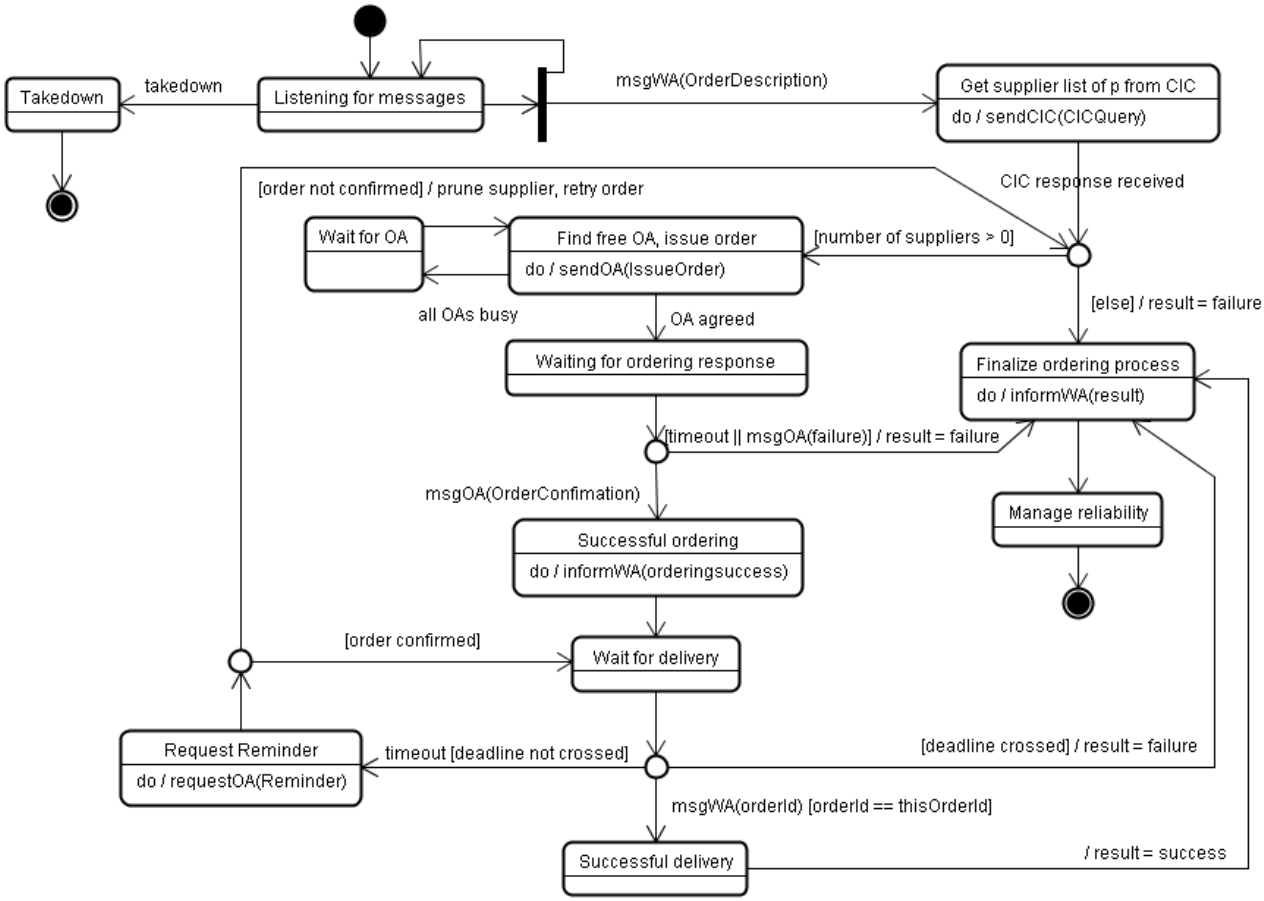

Figure 15: LA state transitions

3.8 Warehouse Agent

WA is one of the already existing agents in the system. Currently its role is maintain the warehouse manifest and to answer queries regarding the availability of products. This role was completely passive, since even when supplies were getting low, no actions were taken. The logistics system extends the role of WA by giving it the possibility to act. Based on information received from SDA, WA can decide whether existing supplies need to be replenished in order to maintain product availability in the future.

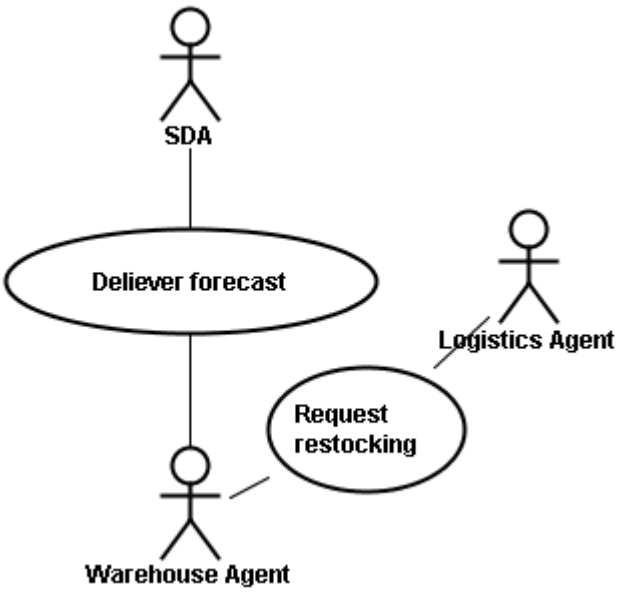

Figure 16: WA use cases

As we can see from the above diagram (fig.16), WA communicates with SDA and with LA. On the diagram below (fig.17) we can observe the process of SDA interacting with WA.

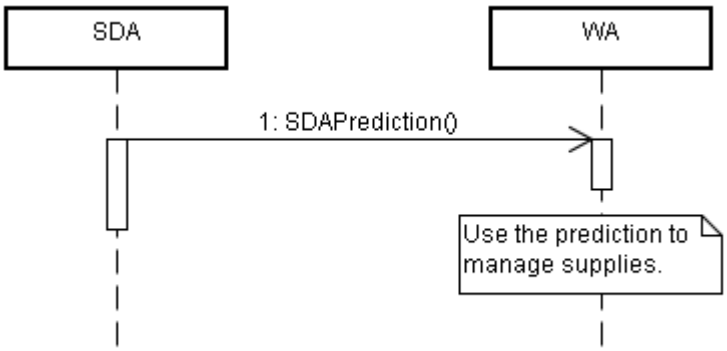

Figure 17: SDA-WA interaction sequence

The communications with SDA are the forecasts prepared by SDA regarding the sales of some product. If forecasts for a given product in the warehouse are available, the WA will monitor its levels periodically (depending on the forecast period) and estimate whether supplies are sufficient. The forecasts are renewed automatically, i.e. if SDA doesn't send updated forecast information, WA assumes that future sales will remain at the levels of the previous forecasts. That means that in order to cancel automatic product restocking, SDA must send an explicit order. In order to update the forecast, SDA needs only to send new information as if the forecast for the product was sent the first time – WA will perform an initial check of the supplies level (perhaps the product needs to be ordered immediately) and then resume normal periodic checks.

This diagram (fig.18) illustrates WA-LA interactions.

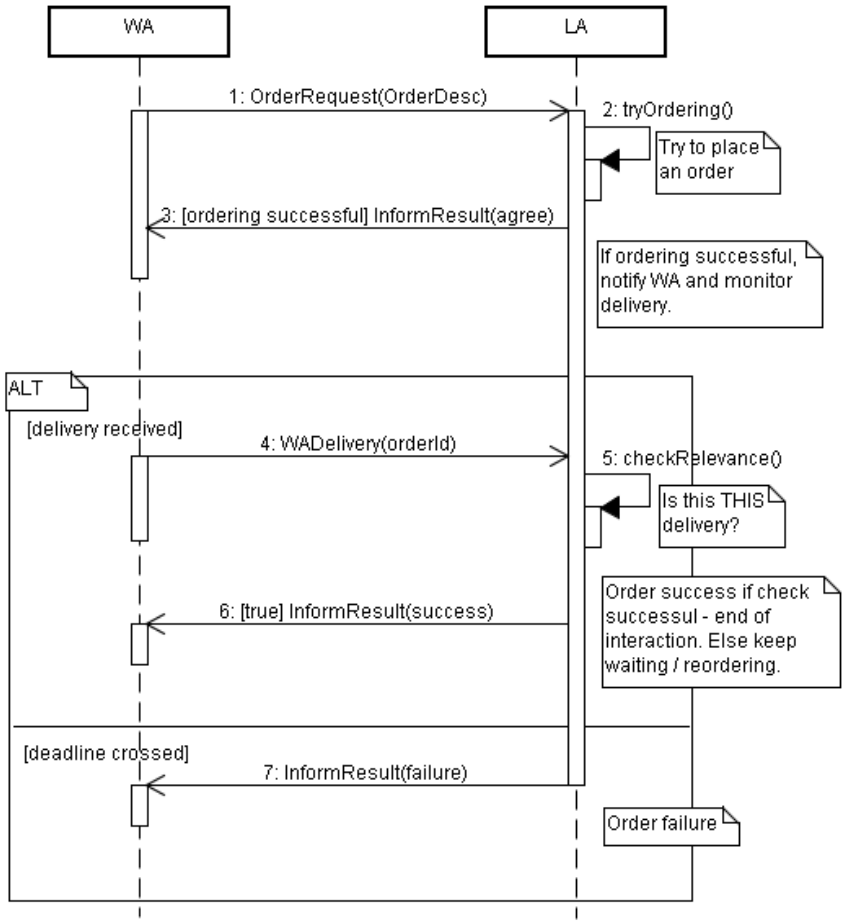

Figure 18: WA-LA interactions sequence

As we can see, there are two types of WA-initiated interactions.

First type is the WA issuing "order request" to LA. WA sends information to LA on the product that needs to be replenished along with order constraints, such as required amount, available time for delivery and price limitations. If LA manages to find a source for the product, it notifies that ordering has succeeded, otherwise that it has failed, and perhaps WA needs to relax the constraints (for example broaden the price range). However, ordering success does not equal a successful order – i.e. we still wait for the delivery. Only when the delivery succeeds, the order is fulfilled.

The other part of WA-LA interactions are delivery notifications. When some delivery

enters the warehouse, WA receives an *orderId* connected with that delivery. This information is passed on to LA, which uses it to conclude the order of the given id, at which time reliability bonuses and penalties are awarded to all participating suppliers. In case such information is not sent to LA in due time, and the order is not yet closed, OA will be used to send reminders to the supplier, and LA will make note of the failed delivery.

To view a complete ordering process, showing all actions of involved agents, refer to appropriate section in Chapter 4.

Here is the complete WA state transitions diagram (fig.19).

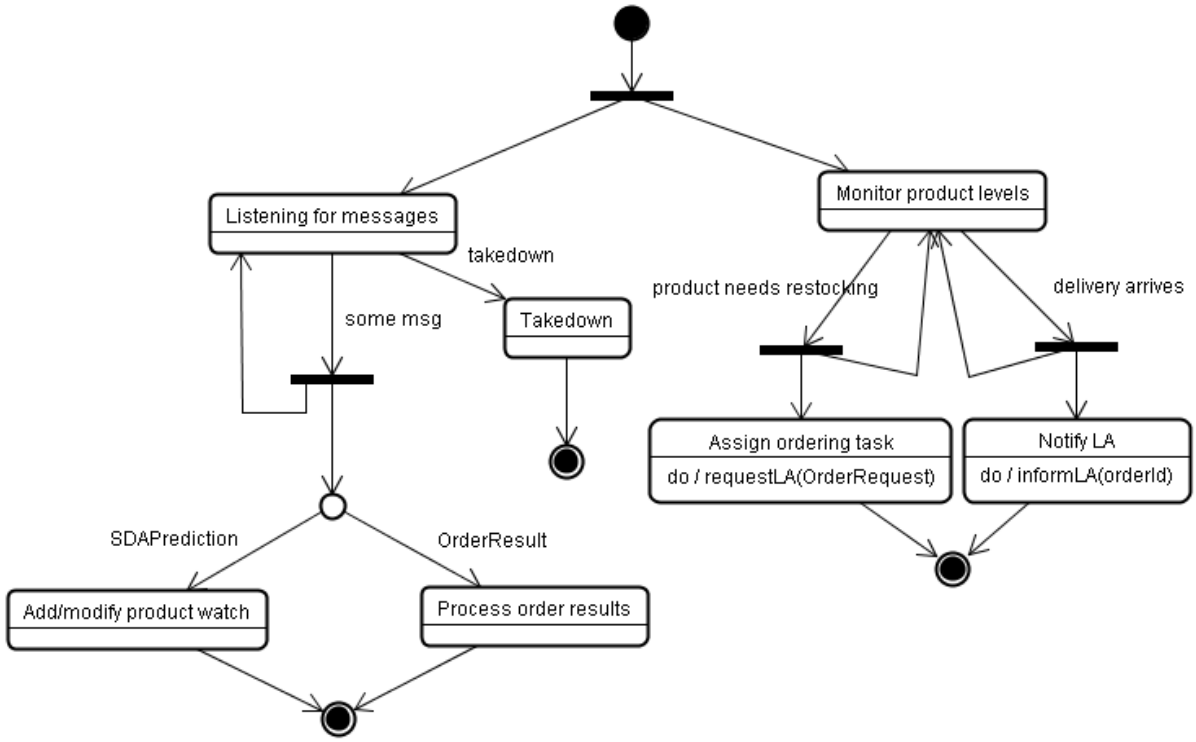

Figure 19: WA state transitions

3.9 Shop Database Agent

SDA is another agent that already exists in the system. Its current responsibilities are maintaining the sales database for statistical purposes. The logistics system will expand on those abilities and require SDA to use the available historical information to produce sales forecasts, that can be used by WA to maintain product availability and reach SCM goals.

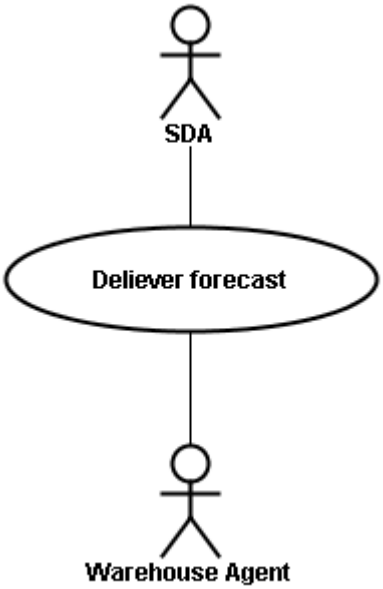

Figure 20: SDA use cases

The above diagram shows interactions of SDA within the system. Only logistics system-relevant interactions are shown, as potential other interactions are irrelevant to the system. In the scope of this thesis, SDA communicates only with WA.

SDA-WA sequence diagram was already presented in the WA description. What is worth noting on the SDA side is that SDA receives no direct feedback from WA, apart of WA acknowledging SDA information. That is because any information that could be sent from WA is either not useful to SDA or is already available to the agent, as the performance of logistics system cannot be gauged directly, and rather is

reflected in the general performance of the system, as seen in the sales database, which SDA already owns.

4 System Scenarios

4.1 Supplier Registration Process

Let us start the review of system scenarios with WhA registering with CIC. Each supplier must register after entering the system, in order to be visible to potential customers – i.e. LA and its OAs. For the best possible system performance it is also useful for WhA to deregister itself when it goes offline or for some reason decides to withdraw from participation in the system.

The messages exchanged between WhA and CIC during WhA's lifetime in the system were presented in Figure 8. The process is initiated by WhA entering the system. Upon entering, the agent contacts platform's singleton CIC agent (here by CIC we mean the logistics CIC, not the old CIC). Following the rules of *FIPA-Request Protocol*, WhA initiates the conversation by sending a message containing *CICRegister* action with *CICAgentDescription* as action contents. *CICAgentDescription* is composed of WhA's own globally unique AID (see [7] p.19 for details on a JADE's typical way of generating such identifiers) and a list of *Products* it offers for sale.

CIC validates such request by comparing the *AID* provided as part of the *CICAgentDescription* message with *AID* of the sender of the message (as defined by FIPA specification SC00061). That means that an agent can only register itself by

sending the message personally, which ensures that the future “supplier” is aware of the registration, and can be held responsible for any deals it may eventually enter. If validation is successful, CIC attempts to enter the provided information into its database. If an entry for a given *AID* is found, CIC updates supplier information, otherwise new entry is created. Successful database insert means successful CIC registration, and WhA is sent confirmation message (*FIPA-Agree* performative) containing *InformResult* action with *CICResult* set for success.

If at any point registration produces an error (validation failure, database connection failure, database operation failure, communication problem), error messages are sent to WhA, containing *InformResult* with *CICResult* set for failure. Descriptive *ResultReasons* are also set. Performatives for the message are set as follows:

- validation failure – *FIPA-Refuse*.
- database connection failure, database operation failure – *FIPA-Failure*.
- communication problem – *FIPA-NotUnderstood*.

After registration at CIC, WhA is reachable by the logistics system. If queried by LA for suppliers of the product WhA declared, the CIC will add that WhA's *CICAgentDescription* to the list of other product suppliers returned to LA. This process happens without notifying WhA. If respective LA forwards WhA's data to OA, WhA will be contacted by OA.

When WhA wishes to leave the system, it is required for it to deregister from CIC. If this obligation is not fulfilled, CIC will continue to provide WhA's data to LAs which might cause OAs to attempt futile contact (if WhA is offline) or unwanted contact (if WhA only decided to end its connection to the system). Deregistration is performed similarly to registration. WhA sends *CICDeregister* action with its *AID*. CIC performs validation as when registering – WhA can deregister only itself. Performatives are set analogously. One expected error is trying to deregister a WhA

that is not present in CIC database – *FIPA-Refuse* is provided as answer, to differ it from CIC internal errors (this error, as well as validation error is a mistake on the WhA side). After removal from CIC, WhA details will no longer be provided to interested agents (since no reliability data is stored in CIC, the details may be safely deleted).

Please note that identical error messages connected to internal agent database problems may arise also while communicating with other agents, and they should be treated similarly. Such errors will not be described in detail in such cases, where it is understandable that agent may fail while saving persistent data, such as product monitoring requests in WA.

4.2 Typical Product Restocking Process

After explaining the mechanics behind CIC registration, we can proceed to describe product restocking process. This will involve maximum number of agents. Previous registration of WhA in CIC is required but not mentioned in the diagram and the description that follows, as it is a completely separate process. Initiation of restocking process by SDA providing forecast data to WA is included. The sequence of actions initiated by WA can be considered as an iterative process (fig.21).

The process starts when SDA produces a sales forecast for a given product. SDA communicates with WA using *FIPA-Request Protocol* and therefore initiates the conversation by sending a *FIPA-Request* message containing *SDAPrediction* action containing *PredictionDescription* (which in turn contains data such as product in question, sales amount, standard deviation of sales, expected buying price and a period for which this forecast is made). This request can be ignored by WA – we can imagine there are warehouses that do not support automation, or perhaps are being

closed for inventory check, or maintenance, and do not need to restock at this time. In such case it is polite to reply with *FIPA-Refuse* message (*InformResult* with *ResultReason* as appropriate). Otherwise WA will respond with *FIPA-Agree* (*InformResult* with *ResultReason* as appropriate, again – this is the default behaviour unless noted otherwise) and get down to business.

WA picks up by examining current stock. If current supplies at the time of receiving the forecast are enough, WA sets up to check the product levels at the beginning of every following forecast period. Forecast periods are set to be units of time, such as *hour, day, week, month, year*. Stocks are checked by WA at the end of each time unit for products which have forecasts for this particular time unit (i.e. products forecast on weekly basis are checked once a week).

If stocks are decided to be insufficient, WA initiates a conversation with LA, as described by *FIPA-Request Protocol* again. This time, however, the system uses extended version of the protocol (refer to FIPA specification SC00026) which requires the receiver to send two confirmation messages. This is done to address the already mentioned problem that successful ordering does not mean a successful order. Let us proceed while keeping this in mind. Initial WA *FIPA-Request* message to LA contains *OrderRequest* action with *OrderRequestDescription*. *OrderRequestDescription* contains information on the product as well as preferred delivery times, amounts and prices. Delivery times and amounts are computed by WA based on current product levels and allowed delivery time and inventory strategy.

Upon receiving the request message, LA dispatches a query to CIC for suppliers of given product. Ensuing conversation conforms to *FIPA-Query Protocol*, starting with *FIPA-Query* message containing *CICQuery* action with *Product*. CIC responds with *FIPA-Inform* containing *CICResponse* with a list of suppliers of the *Product*. It is

possible that the returned list is empty – in such case LA reports order error to WA by immediately sending *FIPA-Failure* with *OrderRequestResult* set to failure. Similar response is sent when CIC cannot be contacted. In case the list was received, and is non-empty, LA prepares *LAAgentDescriptions* to be sent to OA. This is done by supplementing each *CICAgentDescription* received from CIC with reliability information stored in LA database. Agents are identified by their *AIDs*. If given WhA's *AID* is not found, default value is used. This additional information will be used to rank offers in OA.

After preparing the list, LA initiates a conversation with agents in OA pool according to *FIPA-Request Protocol*. Busy agents will respond with *FIPA-Refuse*. One agent must respond with *FIPA-Agree*, otherwise order error will be reported to WA. A free OA will send a *FIPA-Agree* message to LA (no special content is needed, only the performative is required for proper understanding of the message), to notify that it is on the job. LA then stops searching the pool and awaits for the result of the ordering. This initiates the actual purchasing process, as the Information sent by LA in the initial FIPA-Request message – *IssueOrder* action with *OrderDescription* and *LAAgentDescriptions* – is enough to execute the order.

After sending agreement notification OA engages in *FIPA-ContractNet Protocol* with WhAs from the received list by sending *FIPA-CallForProposal* containing *CFPRequest* with *OrderDescription* to all of them. WhAs evaluate the CFP and propose their offers satisfying conditions given by OA by sending *FIPA-Propose* containing *CFPResponse* action with *OfferDescription* or, if terms are unacceptable, respond using *FIPA-Refuse*.

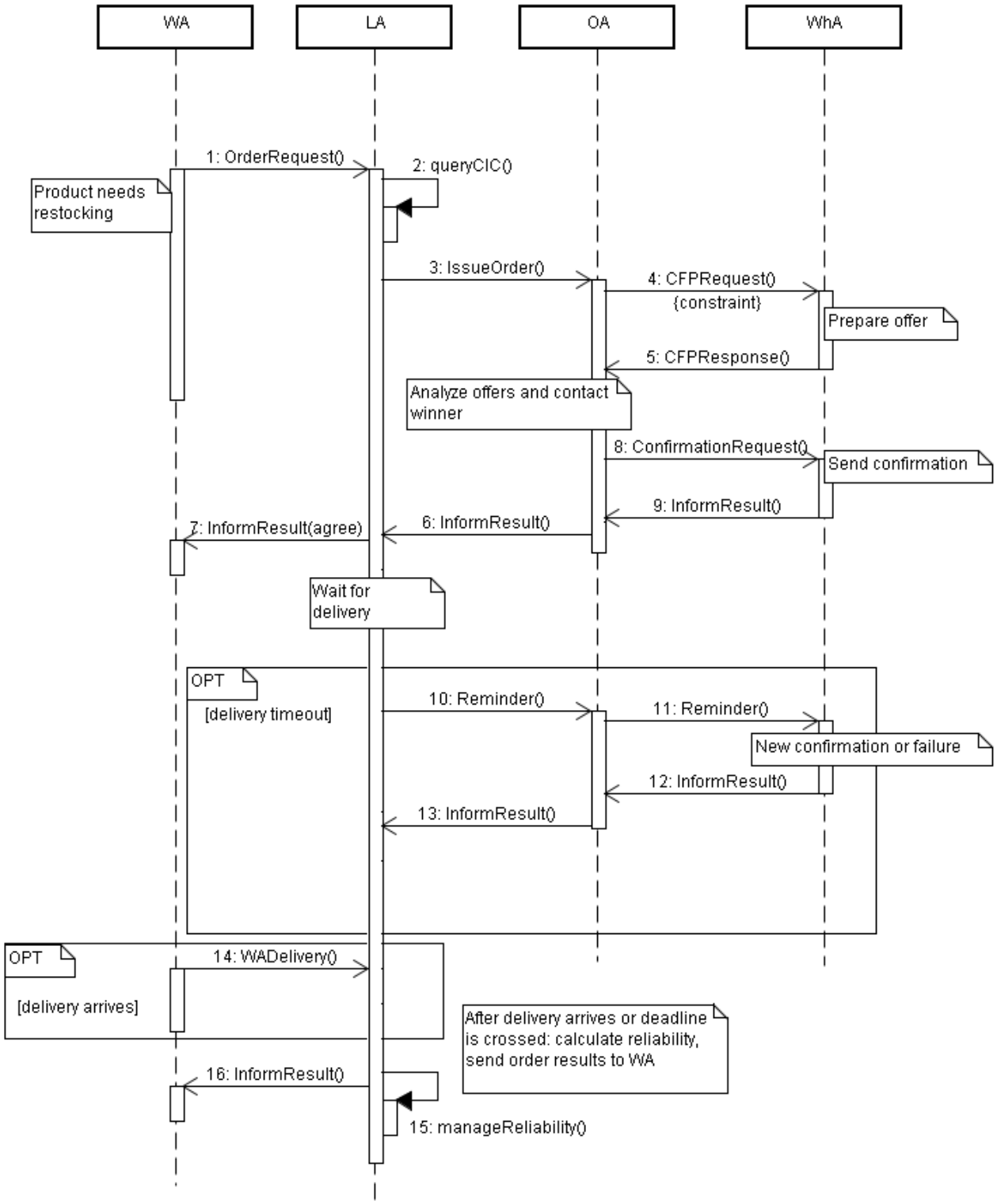

Figure 21: Restocking process

Responses must arrive within a timeframe specified by OA, after which OA filters the offers, since they might not satisfy the conditions, as OA has no control on how loosely WhAs interpret them. OA may optionally prune offers from suppliers whose reliability is sufficiently low, however such offers usually have low rating in the next step anyway, and suppliers we were previously unhappy with can be given the chance to compete with attractive prices. Eventually offers are ranked inside OA (more on this in section 5.6) and the winner is determined. Winner is sent *FIPA-AcceptProposal* containing *ConfirmationRequest* action with its offer quoted, to which it must reply with *FIPA-Inform* containing *ConfirmationResponse* action with *OrderConfirmation* which has supplier generated unique *orderId* attached (or withdraw the offer by sending *FIPA-Failure*). In case the winner withdraws the offer, runners-up are contacted in an iterative manner. In case there is no more offers left or there were no offers to begin with, LA is sent *FIPA-Failure* and it forwards the error to WA in the same manner as in LA when there is no suppliers received from CIC or no available OAs. In the one lucky case when the winner confirms the order, OA sends LA a *FIPA-Inform* message containing *InformResult* action with WhA-received *OrderConfirmation*, thus completing the protocol. At this time LA sends good news to WA inside a message of type *FIPA-Agree*. This performative is used in compliance with the protocol to indicate that LA is performing the desired task (ordering), but its efforts do not guarantee success (ordering success != order success), and so sending final response (*FIPA-Inform*) is inappropriate. Meanwhile OA has completed its part of the task, and so returns to the pool.

This is not the end of the process however. Now the purchase enters delivery monitoring stage. In this stage LA waits for the delivery from WhA to be registered in WA. When any delivery arrives to the warehouse WA knows about it, and sends the information to its LA, which might be waiting to complete its ordering process. Such message is a plain FIPA-Inform message containing *WADelivery* action with *DeliveryDescription*, which has supplier's *AID* and already mentioned *orderId* inside.

LA does not need to respond to the message.

LA checks such messages to see if it isn't currently waiting for a delivery with this *orderId* coming from a supplier with this *AID*. If it finds a match, that means that the ordering process is completed, since the supplier kept his end of the deal. As a reward, the supplier receives reliability bonus, which will be taken into consideration when another opportunity to do business arises, and OA chooses which supplier to trust. If a delivery notification does not come within the time promised by the supplier in *OrderConfirm*, actions must be taken. Those actions, which are described in more detail below, are:

- retry the ordering (sending reminder to WhA / choosing new WhA), if there is still time,
- noting that a retry has been made.

If there is still time before the deadline (see section on setting deadlines for more details), all is not yet lost and the order can by retried. If it is the first time we attempt to retry the order, we can send a reminder to the same WhA. To this end, LA contacts a free OA with *FIPA-Request Protocol* starting with *Reminder* action containing *AID* of the WhA and orderId. OA initially accepts (*FIPA-Agree*) and contacts the WhA (also using *FIPA-Request Protocol*), sending the exact same action to WhA. WhA is expected to reply within a timeframe using either *FIPA-Failure* / *FIPA-Refuse* (withdraws its offer) or *FIPA-Inform* providing new *OrderConfirmation* with new delivery time (all other information contained within is irrelevant). This information is forwarded to LA unchanged (aside from changing WhA *FIPA-Refuse* performative to *FIPA-Failure* for clarity in protocol). In case of agreement, LA returns to awaiting delivery, in case of failure, LA removes this WhA from *LAAgentDescription* list and calls OA to perform entirely new search for a supplier. New search is also ordered if we already sent a reminder to the supplier whose delivery we were waiting on.

The monitoring stage (and order process) ends when:

1. delivery is received (order request result – success)
2. reminder to the same supplier was made, but it was refused, while deadline is already crossed (order request result – failure).

First case is obvious. The second case might need to be explained. While the deadline is not crossed, we may retry orders freely. However, if deadline is crossed, we cannot automatically assume that the delivery will not come and end the process – the delivery may be delayed and come slightly after the deadline. We should wait until delivery time expires. At this time we should send reminder to WhA – perhaps the delivery is on the way. If it refuses or we already reminded that WhA, there is no point in searching for new supplier – the deadline is already crossed, no delivery is on the way, so order request fails.

When monitoring stage ends, WA should be notified of the result and reliability bonuses and penalties should be calculated and applied. More details on reliability issues reward mechanics in the next chapter.

LA notifies WA of the order result by sending *FIPA-Inform* or *FIPA-Failure* to complete *FIPA-Request protocol*. The message will contain *OrderRequest* action with *OrderRequestResult* set appropriately.

5 Challenges and Solutions

5.1 Meeting Demand

The only thing certain about a forecast is that it is always wrong. This is a very popular saying, yet many companies invest a lot of time and resources to predict future demand as accurately possible. The reason behind it is that is beneficial to know the future. When sales for next week, next month, and next year are known to supply chain participants, they only invest in the facilities, equipment, materials, and staffing that they need. There are huge opportunities to minimize costs and maximize profits if we have such knowledge. Unfortunately it is not available, therefore approximate methods are used. In our system we also try to forecast – it is one of required elements of a logistics support system that tries to be of any good.

Pull strategy

If no forecasts are made, orders for supplies cannot be placed ahead of time. Required goods are ordered when the customer places an order. This approach is called *pull strategy* in supply chain management. In industry, another name for this is *Build to Order* (BTO). This is the oldest style of order fulfilment and the simplest, as it requires no additional resources nor effort to perform as expected. Nonetheless, it is used even today, as it is still the most appropriate approach for highly customized or low-volume products, such as engineering designs (e.g. bridges, buildings), specialized computer systems, luxury goods (e.g. cars, yachts), or aircraft. In addition to the level of customization required, additional causes may be high production and storage costs, which make production without a guaranteed sale and instant transfer

of product to the customer infeasible. In such markets it is normal to wait for order fulfilment.

Push strategy

However in case of mass-produced goods *pull* approach is inadequate. High responsiveness to changing customer demand in a highly competitive market is crucial. In such cases *push strategy* – also called *Build to Stock* (BTS) satisfies customer requirements better. There are also some markets (e.g. perfume, designer clothing, gaming consoles) in which assuming this strategy (together with advertising) is required to sell anything – the customer must first be "convinced" by the supplier that they need the product, and when the demand is sparked, the product must be instantly available in large quantities. In this strategy, the supplier accumulates large amounts of product before receiving orders, and effectively when they come, the product is already waiting for the customer.

This approach, although very convenient for the consumer, is dangerous or at the very least inefficient for the supplier. Stocking up on the product means investing resources in something which might not sell as well as expected (or not at all). Also storage costs must be taken into consideration. Additionally, certain market analysis or historical research must be made to produce a sales forecast. Overall, this approach is well suited for products for which the demand is constant or changes in demand are easily predicted (e.g. seasonal goods).

Hybrid strategy

Usually the supplier tries to achieve some kind of balance between the cost efficiency of *pulling* and the product availability provided by *pushing*. That is – to minimize storage while still having enough product to satisfy customer demand in the shortest time. This can be achieved by ordering only as much product as there is demand for and doing this in such a moment that it arrives in the warehouse just in time for the client to pick it up. Of course for such ideal scenario a perfect knowledge of the

future is required. However, using a reasonably accurate forecasting mechanism and a dynamic ordering scheme the system can come close.

5.2 Forecasting Mechanisms

This section is for informational purposes to provide more complete picture of the logistics system. Development of a forecasting module for the logistics system is a separate research project and is beyond the scope of this thesis.

There are numerous approaches that can be taken to forecasting. Some are more suited to our problem than others. The graph shown on next page (fig.22), courtesy of [8], is very helpful in determining the most appropriate methods. By examining our problem and answering provided questions, we can find a methodology that should be most useful to us.

We progress as follows:

1. *Do we have data that can be analysed mathematically?* – **Yes**. That data is our sales database tended to by SDA. Therefore we consider quantitative methods.
2. *Is useful knowledge about causal relationships and other associations available*? – **Yes**, we have sufficient knowledge (e.g. we know that bathing suits sell better in warm seasons, and gore-tex jackets in cold months).
3. *Is the relevant future likely to be substantially different from the past?* – **Depends** on product. Due to the diversity of the products, that our e-commerce system may offer, we cannot give a single answer. For bread, for instance, change in demand is not likely. For high-tech gadgets, products enter and leave the market constantly. Therefore it is likely that the system would need to use several methods interchangeably.

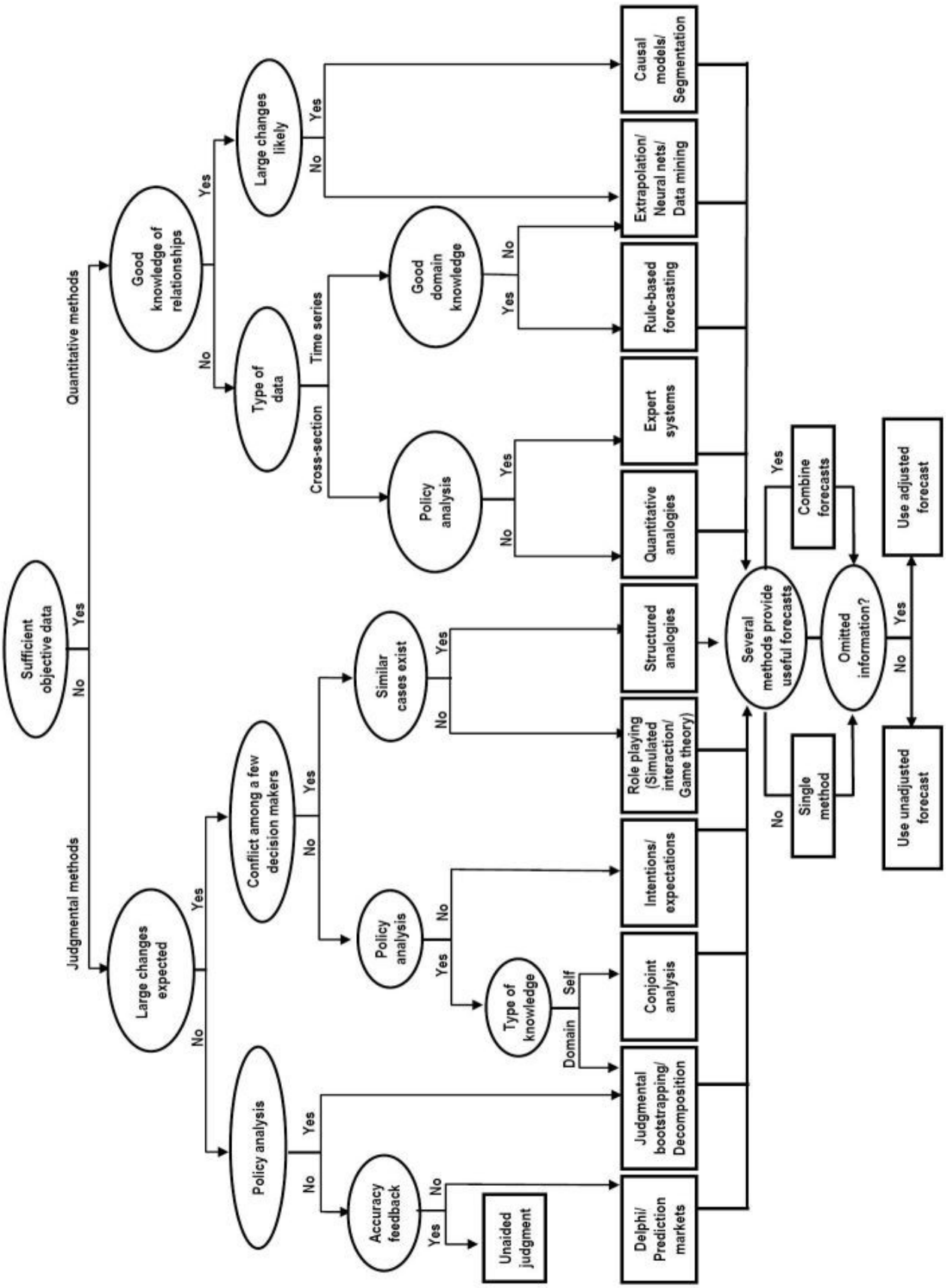

Figure 22: Forecasting methodology selection graph

The process leaves us with several options.

For more or less stable markets we have:

1. *Extrapolation* – this approach uses time-series data, or similar cross-sectional data, to make predictions. For example, *exponential smoothing* is used to extrapolate over time and *diffusion models* are used for innovations.
2. *Neural networks* – the utility of artificial neural network models lies in the fact that they can be used to infer a function from observations. They particularly excel in applications where the complexity of the data or task makes the design of such a function by hand impractical, such as large data sets we might deal with while forecasting.
3. *Data mining* – involves letting the data speak for itself. In general, theory is not considered. Although the method is widely used and the forecasts it produces are claimed to be accurate, there is not much evidence that data mining provides forecasts that are more accurate than those from alternative methods [9].

For dynamic markets we can use:

4. *Causal models* – here theory, prior research and expert domain knowledge are used to specify relationships between a variable to be forecast and explanatory variables. In the case of *econometric methods*, regression analysis is commonly used to estimate model coefficients such that they are consistent with prior knowledge. *System dynamics* models relationships using stocks and flows, often with an emphasis on feedback loops. *Causal models* aided by the use of *econometrics* has in fact been found to improve accuracy. The use of *system dynamics* has not.
5. *Segmentation* – when segments are independent, a *tree structure* is appropriate. When information is available on relationships between segments, *input-output analysis*, *system dynamics*, and *cluster analysis* can be used. Of the dependent segmentation techniques, only *input-output* analysis has been found to improve accuracy.

For in-depth information on proposed methods, refer to [10].

5.3 Ordering Scheme

Of more interest to this thesis is the selection of a appropriate ordering scheme. The proposed strategy is the Just In Time (JIT), which is an inventory strategy designed designed to improve the return on investment of a business – the ratio of income to the expenses connected with producing that income. The means of this improvement in JIT is the reduction of in-process inventory and its associated costs.

The technique was first used by the Ford Motor Company as described explicitly by Henry Ford's "*My Life and Work*" (1922): "*We have found in buying materials that it is not worthwhile to buy for other than immediate needs. We buy only enough to fit into the plan of production, taking into consideration the state of transportation at the time. If transportation were perfect and an even flow of materials could be assured, it would not be necessary to carry any stock whatsoever. The carloads of raw materials would arrive on schedule and in the planned order and amounts, and go from the railway cars into production. That would save a great deal of money, for it would give a very rapid turnover and thus decrease the amount of money tied up in materials. With bad transportation one has to carry larger stocks*." That system was quickly adopted by Japanese companies, because they could not afford large amounts of land to warehouse finished products and parts. Ford's concept was expanded on and philosophies like the Toyota Production System were developed.

It is a mistake to think of JIT as a system for manufacturers, as it originally has nothing to do with production itself, although it was first used in manufacturing and several of its features translate directly to manufacturing optimization. In fact, the

strategy focuses on logistics support for production. We can simply omit the production process and assume that we sell what we buy (what is exactly what the e-commerce system does) and the strategy remains applicable.

The main innovation of JIT is that inventory is seen as something that incurs costs, in contrast to the traditional way of thinking. What is more, full inventory is considered to be a result of problems with management, production or sales. In short, the just-in-time inventory system is all about having "the right material, at the right time, at the right place, and in the exact amount". In our case, the supplies should be ordered almost exactly in amounts required by the clients. This of course must be weighted against the costs of ordering the supplies, which forces the system to use the warehouse as a buffer between supplier deliveries and customer pick-ups. Additionally the size of this buffer should account for potential supply problems.

Here is a highly simplified mathematical model of the ordering process, which we will use to try to find the optimal size for the buffer.

Let:

K = the incremental cost of placing an order

kc = the cost of carrying one unit of inventory for some period T

D = demand in units over a period T

Q = optimal order size in units

TC = total cost over a period T

We want to know Q.

We assume that demand is constant and that the store runs down the stock to zero and then places an order, which arrives instantly. Hence the average stock held (the average of zero and Q, assuming constant usage) is $Q / 2$. Also, the number of orders placed in the period T is D / Q.

TC consists of two components. The first is the cost of carrying inventory, which is given by *Q* * *kc* / 2, i.e. the average inventory times the carrying cost per unit. The second cost is the cost of placing orders, given by *D* * *K* / *Q*, the number of orders (in period T), *D* / *Q,* times the cost per order, *K*.

Thus total cost for T is:

$$TC = \frac{Q \times kc}{2} + \frac{D \times K}{Q}$$

We differentiate TC with respect to Q and set it equal to 0 to find the value of Q for the minimum of total cost, giving:

$$\frac{d(TC)}{dQ} = \frac{kc}{2} - \frac{K \times D}{Q^2} = 0$$

$$Q^2 = \frac{2 \times K \times D}{kc}$$

$$Q = \sqrt{\frac{2 \times K \times D}{kc}}$$

The above formula is known as Economic Order Quantity or EOQ formula (a model that defines the optimal quantity to order that minimizes total variable costs required to order and hold inventory). This simplified form can be fairly easily adapted to take into account additional realistic features such as delays in delivery times and fluctuations in demand. Both of these are usually modelled by normal distributions.

The delay in delivery, in particular, means that additional 'safety stocks' need to be held if a stockout is to be rendered very unlikely. Experience shows that on average the warehouse should carry about 2 standard deviations of demand as emergency stock to meet 95% service rate (i.e. to be able to satisfy all orders 95% of the time). This stock would be added to our buffer.

In terms of implementation, the value of Q is calculated by WA and is provided in *OrderRequestDescription* as *preferredAmount*. Minimal amount that is required to fill the stock is passed as *requiredAmount*. Preferred amount can only be higher than required amount. It can be lower, however in such case the preference will be ignored, as the priority for the system is preventing a stockout, delivery optimization comes later. Since WA will try to keep the emergency stock filled, its size will be taken into account when determining the amounts that must be ordered.

However, some features of our solution hinder the implementation of complete EOQ:

1. The system interacts with multiple suppliers, while traditional businesses usually tend to pick one supply partner. During its operation, the system will also gradually limit the supplier list to reliable partners, but leaving only one is an isolated case, which creates a problem. The suppliers may have different costs associated with delivering their goods to our warehouse, which implies K having a multitude of values instead of a single one.
2. A choice has been made in the beginning of the development that the suppliers do not quote their delivery costs as a separate value, they rather include it in the total price of the order. Therefore K is not known explicitly, its value can only be inferred, which makes dependence on exact values of this variable unreliable.

Keeping the above in mind, the system still allows the operator to set the value of K manually. kc is a parameter that can be set on per-product basis in WA database as well.

5.4 Deadlines

Another pitfall is determining a proper deadline to use while ordering a product. Here

the solution largely depends on how WA's strategy of setting delivery times it provides to LA and ultimately to OA. We define *maximalTime* as the last moment the supplies must arrive to be available to customers. *allowedTime* can be treated as a measure of flexibility allowed to the ordering system. Therefore the strategy may be described as strict (*allowed* and *maximal* delivery times are very close) or flexible (more time between *allowed* and *maximal* time).

Bearing in mind the discussion on JIT model, it is obvious that using strict strategy is optimal for the model. The goal is to decrease the time the goods are stored in the warehouse, so allowing them to arrive at the latest possible time is preferred. However this only works when the suppliers are reliable. In this approach a supplier failure leaves no time to reorder. Relaxing the time restrictions gives the ordering system to recover from supplier failure by sending reminders and possibly choosing another supplier. This however makes it more likely that goods may arrive early and have to wait for customers.

WA strategy cannot be fixed a priori, as this would restrict the flexibility on WA's part. For instance, normally we may allow a lot of flexibility. However in case of a critical shortage of product (sudden sales spike) the use of a strict strategy is desired. Therefore it must be determined while ordering. The solution to handle this on the LA/OA side is a smart ranking formula used to select suppliers.

5.5 Reliability Management

In logistics system we define reliability as a measure of quality of service received during previous dealings with suppliers. High reliability score means that the supplier was dependable in the past and may be expected to deliver in the future. Low reliability means late or missing deliveries. Having that in mind, it is obvious that

suppliers with higher numbers are more desirable, especially in case of critical orders (e.g. very little time available for delivery).

Reliability management involves assigning proper rewards/penalties to suppliers that confirmed the order. Confirming the orders is treated as entering a binding contract. Fulfilling the contract in agreed timeframe is rewarded, delays or withdrawals are penalized.

LA is the logistics system's reliability manager. It manages a persistent reliability database which contains entries for all known WhAs (those that entered a delivery contract at least once) with their current reliability value. This information is added to *IssueOrderDescription* sent to OA and used to determine the best offer.

Three values must be considered for the reliability system to work:

1. *SuccessBonus*,
2. *FailurePenalty*,
3. *InitialReliability*.

SuccessBonus (further called B) is the award for successful completion of order for the agent which delivered the supplies. *FailurePenalty* (P) is the penalty for breaking the contract (not delivering as promised – delivering late, withdrawing, or not delivering at all). *InitialReliability* (I) is the reliability value that new suppliers (i.e. those the system had not dealt with before) are assigned.

Specific values of those constants are not important, especially since the values are normalized before ranking, rather their relative values. Let us consider several possible relations and discuss their consequences.

B >> I ~ P (large bonus compared to initial value and penalty) – the system favours reliable suppliers. After several suppliers prove their worth to the system, new suppliers' reliability may be too low to compete with the already established.

P >> I ~ B (large penalty compared to initial value and bonus) – the system severely punishes unreliable suppliers. Even after one failure, the supplier will lose with new suppliers.
P ~ B ~ I (comparable values) – the balanced scenario, system does not favour anyone, new suppliers have chances of getting orders, the unreliable still have chance of getting orders and improving their rank. This approach is currently used.

During order monitoring phase, LA stores information on all suppliers (WhAs) involved in the order, i.e. *AIDs* of all agents which confirmed the order (sent *OrderConfirm* to OA). The information is stored as a simple list. At any time OA forwards to LA information that a WhA confirmed the order, that agent's *AID* is added to the list (when a WhA responds to a reminder, it is added to the list again). That list is used to assign reliability points to all involved WhAs when the order is completed – refer to section 4.2 for details on ordering process details and process completion.

When an order is completed (successfully or not) points are awarded, according to the following rules. Rules' explanations are provided below.

1. If order is successful, last agent on the list is awarded *SuccessBonus*.
2. All agents on the list except the last receive *FailurePenalty*.
3. If order fails after deadline, the last agent receives *FailurePenalty* as well.

Rule 1: If order is successful, the last agent is the one that delivered the order. If delivery is after deadline, then a) the agent was sent a reminder earlier, in which case it appears on the list twice, and will be also penalized under rule 2; b) the agent did not have sufficient time before the deadline (e.g. it took over another agent's order), so it shouldn't be penalized.
Rule 2: All agents except the last failed to deliver the supplies. Agents that were sent a reminder, confirmed the order and failed to deliver again, will be penalized twice.

Rule 3: When order fails, it means that an agent which promised to deliver was contacted after the deadline, and then sent refuse message. That agent should be penalized, but it is not subject to rule 2 and needs to be penalized separately.

5.6 Supplier Ranking Formula

The Supplier Ranking Formula (SRF) is a computer representation of the human reasoning performed while determining the 'best' offer. It is used by OA to assign to each offer from received from agents that responded the CFP a rank value. The offers are sorted by this rank value. The agent whose offer got the highest value is chosen to deliver the supplies, and is contacted to confirm the terms. In case the order cannot be confirmed for some reason (network failure, price change, sudden supply shortage in WhA, agent leaving the system, etc.), the second agent in line is contacted, and so on.

The following factors must be taken into consideration by the formula:

1. *order strategy,*
2. *offered price*,
3. *offered amount*,
4. *offered delivery time*,
5. *supplier reliability*.

Order strategy

The main indicators to the chosen ordering strategy (strict or flexible), as mentioned before, are *maximalTime* and *allowedTime* values (or more specifically: their relation to each other). Their ratio (*allowedTime* / *maximalTime*) forms the *Strategy Factor* (*SF*) in the formula. This factor is intended to change the importance (weight) of supplier *reliability*, and in this way cause the formula to give more points to offers that should be more interesting in cases the particular strategy is involved:

1. *Strict strategy* – is useful in cases when stock levels are low, or time is short. In such cases we prefer more reliable suppliers, that are more likely to deliver the goods quickly, to reduce the risk of the need for reminders and reorders.
2. *Flexible strategy* – is useful when we have more time available. In this case we are more wiling to find bargains with new suppliers, as reorders are not that big of a risk.

Offered Price

As usual, lower prices are preferred.

Offered Amount

Issues described in section 5.3 must be considered. Since the logistics system tries to optimize order sizes, the offers with amounts closer to the preferred amount are ranked better. Unless the preferred amount was set to be less than required amount by WA – in such case the preference is ignored.

Offered delivery time

Typically shorter delivery times are preferred.

Supplier reliability

Of course, reliable suppliers are more desirable. This becomes even more important when using strict strategy (see above).

Each of the mentioned factors in the formula has its initial weight assigned. The weights are set in the logistics system scope and reflect the general strategy of the store the logistics system is part of. Greedy stores may place more importance on getting goods cheap, quality stores may stress supplier reliability. Stores more willing to optimize the logistics process may shift weights towards preferred delivery times

and preferred delivery sizes.

All weights are from interval [-1, 1], with 0 causing the SRF to disregard that parameter.

All offer values are normalized according to the average of that value over all offers received by OA using the formula:

$$newV = \frac{oldV - avg(allV)}{avg(allV)}$$

where *avg(allV)* is the average of that value in all offers.

Considering the following variables:

wSF = Strategy Factor initial weight

wP = Price initial weight

wA = preferred Amount initial weight

wT = preferred Time initial weight

wR = Reliability initial weight

SF = Strategy Factor

noP = normalized offered Price

noA = normalized offered Amount

noT = normalized offered delivery Time

noR = normalized offer Reliability (reliability of offerer)

the Supplier Ranking Formula has the following form:

$$SRF = \frac{wP \cdot noP + wA \cdot noA + wT \cdot noT + wR \cdot noR}{wP + wA + wT + wR} + 2 \cdot wSF \cdot SF \cdot noR$$

The SRF takes into consideration all weights and all offer parameters equally and then applies the Strategy Factor that enhances the influence of reliability on final rank

of the offer. Strategy Factor influence can be adjusted by setting its weight as well. Value “2” in the last component was found experimentally as a good adjustment to the “power” of SF. This adjustment was made for all weights equal to 0.2 (all weights summing to 1).

SF controls how willing the OA is to choose other factors, for example price, over reliability. For SF values close to 1, offers with lower reliability need to be very strong in other areas to beat an offer from a reliable supplier.

6 Implementation

6.1 Technology

As the core platform used in this thesis was JADE, the system was implemented using Java 1.4 Runtime Library and JADE version 3.4.1. Additional tools used were Protege versions 3.1 and 3.2-beta with BeanGenerator for Protege 3.1.
UML diagrams were created using JUDE Community 3.0.

6.2 Restocking Scenario in Practice

To illustrate the system in action, the following scenario is run and analyzed:

1. The system consists of the following agents (names of agents given in parentheses): single SDA (sda), WA (wa) and LA (la) and CIC (cic) agents, a pool of 2 OAs (oa1, oa2) and 2 WhAs (wha1, wha2) selling the same product.
2. Both WhAs have already registered with CIC.
3. SDA produces a prediction for a product sold by both WhAs.

4. WA stock is empty.
5. Delivery times and deadlines are adjusted and WhA agents fail to deliver, so there is need and there is enough time to show reminder mechanism.
6. JADE platform Sniffer Agent is attached to all system agents.
7. JADE platform Introspector Agent is attached to LA agent.

The following screenshot (fig.23) shows Sniffer output after WA delivery order failed.

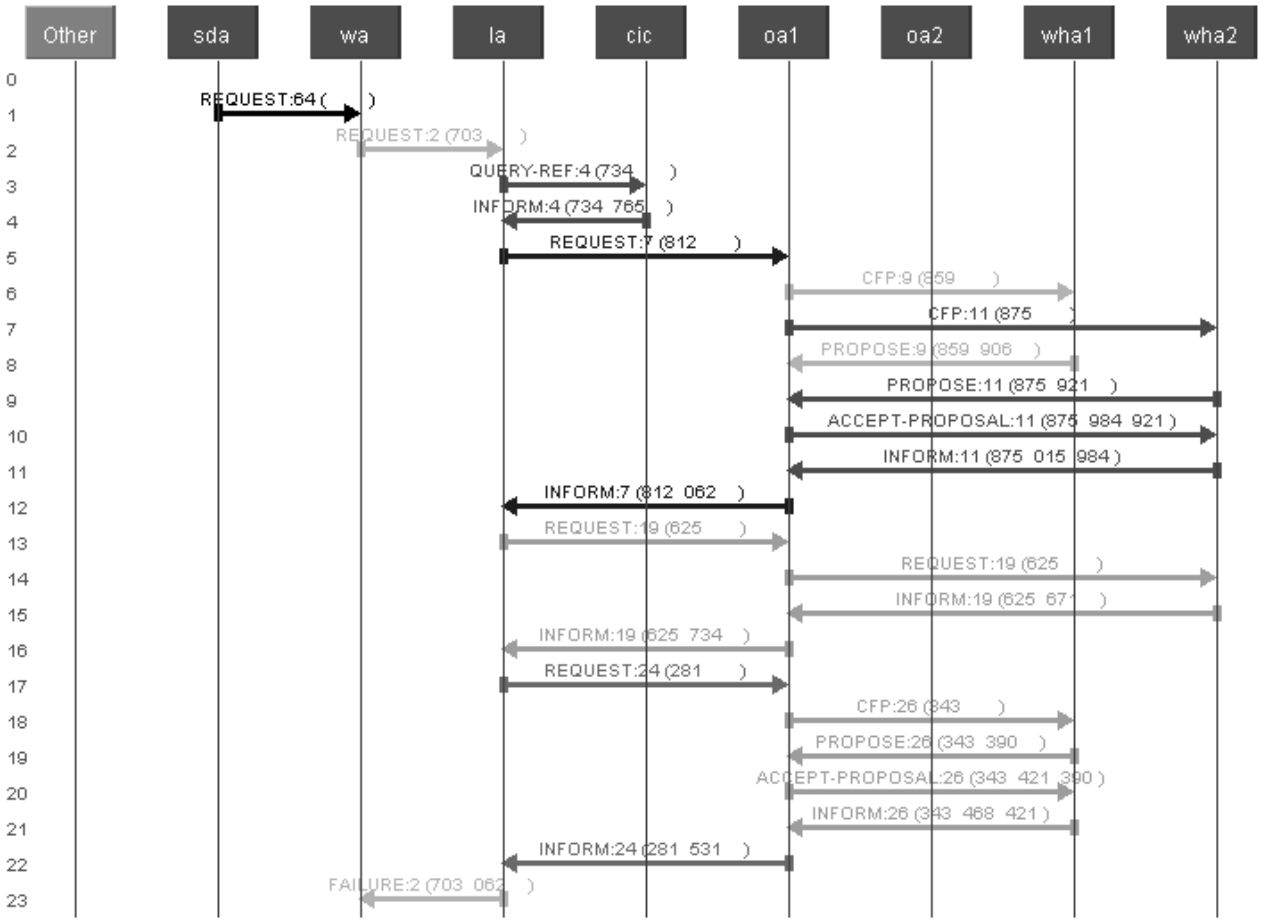

Figure 23: Messages in the system as observed using Sniffer Agent

Let us analyze what happens in the scenario. Numbers of relevant messages are enclosed in curly brackets.

- *sda* sends its prediction to *wa* {1}
- WA checks stock levels for the product – since stocks are empty, it immediately requests ordering from *la* {2}

- upon receiving the request, *la* queries *cic* for suppliers of required product {3} and receives the list {4} containing addresses of *wha1* and *wha2*
- since some suppliers of product are available, *la* requests first available OA – in this case *oa1* is contacted first and is free – to place the order {5}
- *oa1* sends CFP to both WhAs {6, 7} and they both respond {8, 9}
- after proposals are received, *oa1* filters them (none is rejected) and ranks them. *wha2* wins and its proposal is accepted {10}
- *wha2* responds confirming the order {11}
- *oa1* forwards the confirmation from *wha2* to la as success message {12}
- *la* checks the delivery time provided in confirmation and waits. After the time promised passes, and no delivery notification is received from *wa*, and there is still time to deadline, *la* contacts the first available OA (*oa1* again). Since it is the first delivery timeout for this supplier in this order, reminder request is sent to *oa1* {13}
- *oa1* contacts specified supplier by sending him the reminder {14}, and it confirms with new expected time for delivery {15}. This information is again forwarded to *la* {16}
- *la* checks the delivery time provided in confirmation and waits again. After the time promised passes, and no delivery notification is received from *wa* (while still time to deadline), *la* contacts the first available OA (*oa1* again). This time however, we already sent reminder to this supplier. So wha2 is removed from potential supplier list, and new order request is sent to oa1 {17}
- *oa1* initiates FIPA-ContractNet again, this time however with last remaining supplier, *wha1*, which "automatically" wins and its offer is accepted and confirmed {18 – 21}. Confirmation is forwarded to *la* {22}
- *la* checks the delivery time provided in confirmation and waits yet again. After the time promised passes, and no delivery notification is received from *wa*, this time deadline is crossed, so order fails, and proper notification is sent to *wa* {23}

The following screenshot (fig. 24) presents Introspector display for *la* after ordering is finished. We can notice all messages received and sent by *la* and view internal behaviours of the agent.

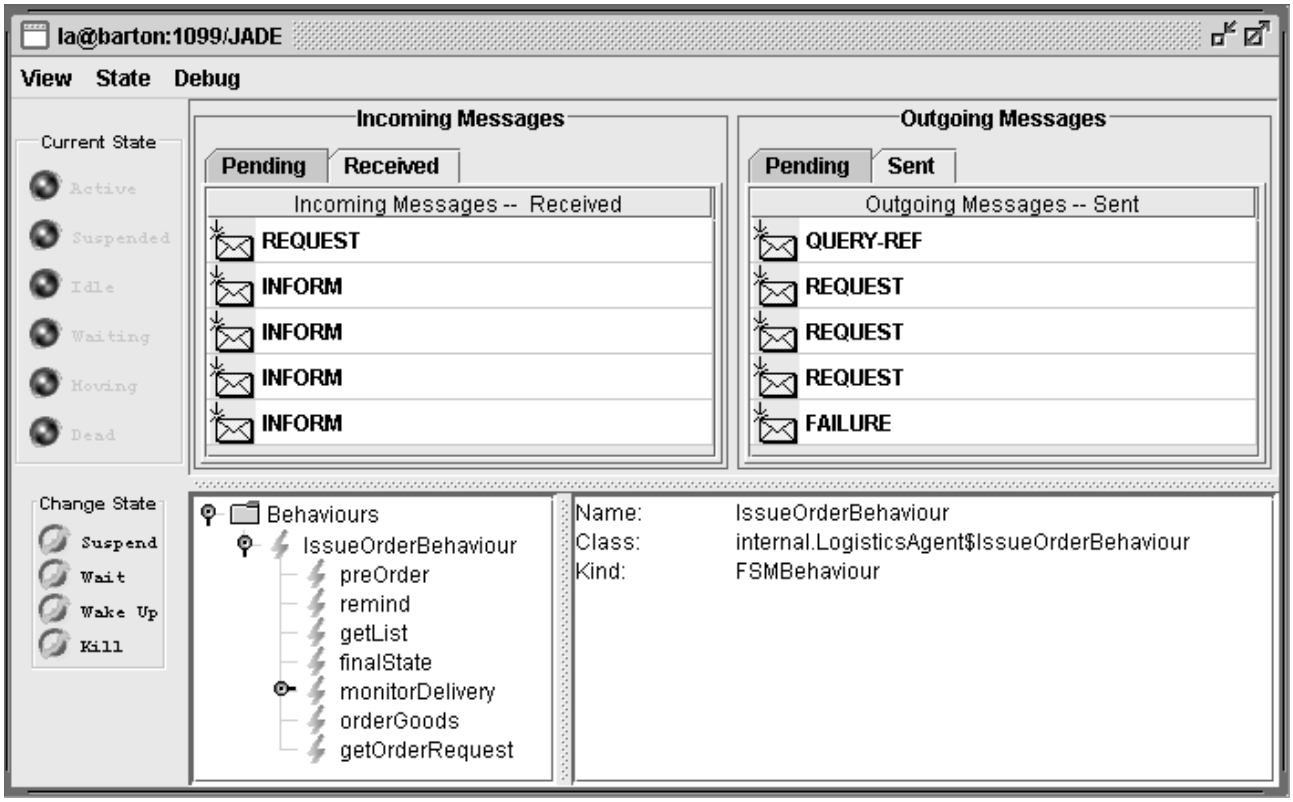

Figure 24: Introspector display for LA after ordering is finished

Following sections will discuss various issues encountered during the implementation phase of the system.

6.3 Performance and Used Technical Solutions

Due to the design assumptions of this system, as well as purpose of this thesis, performance issues were not of key importance in implementation of the system. The focus here was on attempting to apply traditional business methodologies to assigning and managing the tasks performed by agents.

Therefore technical solutions used in the thesis project are the simplest possible, with several obvious optimizations in several places, however this choice was made with the same purpose in mind as when choosing JADE as ready-to-use agent platform – to shift developer efforts from technical issues to providing functional value to the application. There are many other papers dealing with optimizing the performance of agent systems, as well as of computer systems in general.

For functional testing, the approach taken was discovered to be more than enough.

6.4 JADE Ontology Support

Agents use ontologies to represent concepts and actions. The operational ontology developed for the use by the thesis project is called *Logistics Ontology* and its RDF/XML (Resource Definition Framework [13]) representation is provided in the Appendix 9.1. JADE simplifies the use of ontologies in the code of agents by providing support for Java classes implementing the ontologies. This allows ontologies to be designed in visual editors (Protege [14] was used) and generate the required classes automatically using plugins (such as BeanGenerator [14]).

One encounters several problems, however. The tools (as JADE itself) are under constant development and some are no longer supported (for example BeanGenerator), which leaves bugs in their functionality, so more often than not supervision and manual corrections are required. The author of the thesis had to recreate several ontology classes by hand when faced with one of BeanGenerator glitches. All in all, the tools provide valuable support and save a good amount of time that would otherwise be spent on work that can be automated.

Likewise, although JADE provides API for translating ontology objects represented

by Java classes to and from *FIPA*-compliant messages, the API itself is often counter-intuitive, cumbersome and overall hard to use, forcing the programmer to rewrite lengthy code to facilitate simple actions. Code snippet of a function extracting an ontology object from ACL message is provided in Appendix 9.2. It would be useful to further the abstraction level when working with ontology objects to streamline the serialization and deserialization of those objects into/from *ACLMessages*.

6.5 Behaviours

Tasks performed inside agents are represented in JADE as Behaviours. Many types of generic behaviours are provided, such as *OneShotBehaviour* (one time execution) *CyclicBehaviour* (continuous execution), which can be subclassed by the developer to provide required functionality. There are also more advanced behaviours that are aimed to relieve the programmer of coding support for FIPA protocols, such as *AchieveREInitiator*/*Responder* that help with *FIPA-Request*-type protocols or even more advanced *ContractNetInitiator* /*Responder* intended to handle the mechanics of *FIPA-ContractNet*.

While this solution is usually sufficient for supporting direct conversations (an agent with another agent), problems arise when conversations require the agent to coordinate with another agent before (or while) deciding what to do next in the conversation. An example may be the LA, which needs to inform WA of the progress of its talks with OA. We face another problem, when we require some additional functionality in the agent while he is performing the protocol. For example in the OA, which could use *ContractNetInitiator* when talking using *FIPA-ContractNet* with WA (which is using *ContractNetResponder*). However we cannot, as in standard protocol, pick the winning offer, send *Accept-Proposal*, wait for answer (which can be negative) and complete the protocol. Negative answer in our case means the need

to contact the WhA that gave the second best offer. It is not possible to simply inject that functionality into provided initiator behaviour.

Those problems force the programmer to implement the required functionality using the most advanced *FSMBehaviour*, which, as the name implies, itself implements a finite state machine. This requires setting up individual simple behaviours as states and organizing the transitions between them. The developer must take great care to save exchanged messages, conversation identifiers and other information, that might be useful to create replies adhering to protocol rules later on. This requirement denies the convenience of using predefined Behaviours.

7 Conclusion

In this thesis I have presented the application of software agents to solving a common problem in today's business – logistics management. The main benefit of using agents over more traditional software solutions is that due to their design to imitate human behaviour, they are easy to assign roles and responsibilities. Typically, one can prototype an agent for each task performed by a human within the business' logistics department and model its behaviours to mimic the decisions made by that person.

This leads to a clear view of the system from the business point of view. The past problems of implementing the agents themselves have been lessened by development of common standards for communication and agent definitions and middlewares such as JADE, that take care of most of technical issues. The developer may focus on implementing human ways of thinking in the agents. To this end, I have shown that traditional methodologies and strategies for business process management are

applicable for agents, as well as for people.

Possible future extension to the proposed agent-based logistics system is to increase the system's awareness of technical issues connected with logistics that can be used to further system's real-life usefulness and efficiency, and which were omitted while preparing the initial concept. Knowledge of issues like warehouse and supplier locations, distances, modes of transport, freight sizes and product expiration, as well as collecting data on system performance, would provide more flexibility in decision-making in addition to bringing the system closer to real world.

The next important goal in the future is to completely implement the business methodologies and strategies, without the limitations that were imposed by lack of some of the mentioned information in the system (such as making predictions based on previous supplier performance).

However, when choosing agents as the technology for their system, one should be wary of the difficulties connected with developing a system using this relatively new approach, which were described in Chapter 6. One must realize that the pioneer's path is never easy, and when making the choice whether to follow it or not one must consider if the promised benefits are worth the effort.

8 References

1. Haag, S., Cummings, M., McCubbrey, D., Pinsonneault, A., & Donovan, R., "Management Information Systems For the Information Age" (3rd Canadian Ed.), McGraw Hill Ryerson 2006.

2. Ritzman, Larry P., Lee J. Krajewski and Robert D. Klassen, "Foundations of Operations Management" Canadian Edition, Pearson Education Canada 2004.

3. Ganzha, M., Paprzycki, M., Pirvanescu A., Badica, C., Abraham, A., "JADE-based Multi-agent E-commerce Environment: Initial Implementation" 2004.

4. Foundation for Intelligent Physical Agents, http://www.fipa.org

5. Java Agent Development Framework, http://jade.tilab.com

6. Dominiak, M. et al., "Efficient Matchmaking in an Agent-based Grid Resource Brokering System", 2006.

7. Fabio Bellifemine et al., "JADE Administrator's Guide", Version 3.4, 28 February 2006.

8. International Institute of Forecasters, http://www.forecasters.org/

9. J. Scott Armstrong, "Findings from Evidence-based Forecasting: Methods for Reducing Forecast Error", Wharton School, University of Pennsylvania 2006.

10. J. Scott Armstrong, "Long-range Forecasting" (2nd Edition), John Wiley & Sons Inc, 1985.

11. ERP Market Survey 2004, http://www.itjungle.com/tfh/tfh062005-story03.html

12. FIPA Specifications, http://www.fipa.org/repository/standardspecs.html

13. W3C RDF/XML Specification, http://www.w3.org/RDF/

14. Protege & BeanGenerator plugin, http://protege.stanford.edu/

15. Carl Hewitt, Peter Bishop, Richard Steiger, "A Universal Modular ACTOR

Formalism for Artificial Intelligence", 1973.

16. Council of Supply Chain Management Professionals, http://www.cscmp.org/

9 Appendices

9.1 Logistics Ontology in RDF/XML

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:p1="http://jade.cselt.it/beangenerator#"
    xmlns="http://LogisticsOntology.owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
  xml:base="http://LogisticsOntology.owl" >
  <rdf:Description rdf:about="#CFPResponse">
    <rdfs:subClassOf rdf:resource="http://jade.cselt.it/beangenerator#AgentAction"/>
    <rdf:type rdf:resource="http://jade.cselt.it/beangenerator#JADE-CLASS"/>
  </rdf:Description>
  <rdf:Description rdf:about="#DeliveryDescription">
    <rdfs:subClassOf rdf:resource="#Description"/>
    <rdf:type rdf:resource="http://jade.cselt.it/beangenerator#JADE-CLASS"/>
  </rdf:Description>
  <rdf:Description rdf:about="#OfferDescription">
    <rdfs:subClassOf rdf:resource="#Description"/>
    <rdf:type rdf:resource="http://jade.cselt.it/beangenerator#JADE-CLASS"/>
  </rdf:Description>
  <rdf:Description rdf:nodeID="A0">
    <rdf:rest rdf:nodeID="A1"/>
    <rdf:first rdf:resource="#Reminder"/>
  </rdf:Description>
  <rdf:Description rdf:about="#result">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
    <rdfs:range rdf:resource="#Result"/>
    <rdfs:domain rdf:resource="#InformResult"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  </rdf:Description>
  <rdf:Description rdf:nodeID="A2">
    <rdf:first rdf:resource="#CICAgentDescription"/>
    <rdf:rest rdf:nodeID="A3"/>
  </rdf:Description>
  <rdf:Description rdf:about="#predictionPeriod">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
    <rdfs:domain rdf:resource="#PredictionDescription"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  </rdf:Description>
  <rdf:Description rdf:nodeID="A4">
    <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
    <rdf:first rdf:resource="#OrderConfirmation"/>
  </rdf:Description>
  <rdf:Description rdf:nodeID="A5">
    <rdf:rest rdf:nodeID="A6"/>
    <rdf:first rdf:resource="#OrderConfirmation"/>
  </rdf:Description>
  <rdf:Description rdf:about="#Product">
    <rdfs:subClassOf rdf:resource="http://jade.cselt.it/beangenerator#Concept"/>
    <rdf:type rdf:resource="http://jade.cselt.it/beangenerator#JADE-CLASS"/>
  </rdf:Description>
  <rdf:Description rdf:about="#OrderResult">
    <rdf:type rdf:resource="http://jade.cselt.it/beangenerator#JADE-CLASS"/>
```

```
    <rdfs:subClassOf rdf:resource="#Result"/>
  </rdf:Description>
  <rdf:Description rdf:about="#Request">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">used for both
WA->LA logistics request and OA CFP</rdfs:comment>
    <rdfs:subClassOf rdf:resource="http://jade.cselt.it/beangenerator#AgentAction"/>
    <rdf:type rdf:resource="http://jade.cselt.it/beangenerator#JADE-CLASS"/>
  </rdf:Description>
  <rdf:Description rdf:about="#CICQuery">
    <rdfs:subClassOf rdf:resource="http://jade.cselt.it/beangenerator#AgentAction"/>
    <rdf:type rdf:resource="http://jade.cselt.it/beangenerator#JADE-CLASS"/>
  </rdf:Description>
  <rdf:Description rdf:nodeID="A7">
    <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
    <rdf:first rdf:resource="#ConfirmationResponse"/>
  </rdf:Description>
  <rdf:Description rdf:nodeID="A8">
    <rdf:first rdf:resource="#OfferDescription"/>
    <rdf:rest rdf:nodeID="A9"/>
  </rdf:Description>
  <rdf:Description rdf:about="#Reminder">
    <rdfs:subClassOf rdf:resource="http://jade.cselt.it/beangenerator#AgentAction"/>
    <rdf:type rdf:resource="http://jade.cselt.it/beangenerator#JADE-CLASS"/>
  </rdf:Description>
  <rdf:Description rdf:nodeID="A10">
    <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
    <rdf:first rdf:resource="#CICDeregister"/>
  </rdf:Description>
  <rdf:Description rdf:about="#Result">
    <rdf:type rdf:resource="http://jade.cselt.it/beangenerator#JADE-CLASS"/>
    <rdfs:subClassOf rdf:resource="http://jade.cselt.it/beangenerator#Concept"/>
  </rdf:Description>
  <rdf:Description rdf:nodeID="A11">
    <owl:unionOf rdf:nodeID="A12"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
  </rdf:Description>
  <rdf:Description rdf:about="#OrderRequestResult">
    <rdfs:subClassOf rdf:resource="#Result"/>
    <rdf:type rdf:resource="http://jade.cselt.it/beangenerator#JADE-CLASS"/>
  </rdf:Description>
  <rdf:Description rdf:nodeID="A13">
    <rdf:first rdf:resource="#OrderRequestResult"/>
    <rdf:rest rdf:nodeID="A0"/>
  </rdf:Description>
  <rdf:Description rdf:nodeID="A14">
    <rdf:first rdf:resource="#ConfirmationRequest"/>
    <rdf:rest rdf:nodeID="A4"/>
  </rdf:Description>
  <rdf:Description rdf:nodeID="A15">
    <rdf:rest rdf:nodeID="A7"/>
    <rdf:first rdf:resource="#IssueOrderResult"/>
  </rdf:Description>
  <rdf:Description rdf:nodeID="A16">
    <rdf:first rdf:resource="#CFPRequest"/>
    <rdf:rest rdf:nodeID="A17"/>
  </rdf:Description>
  <rdf:Description rdf:about="#predictionAmount">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
    <rdfs:domain rdf:resource="#PredictionDescription"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  </rdf:Description>
  <rdf:Description rdf:about="#CICDeregister">
    <rdf:type rdf:resource="http://jade.cselt.it/beangenerator#JADE-CLASS"/>
    <rdfs:subClassOf rdf:resource="http://jade.cselt.it/beangenerator#AgentAction"/>
  </rdf:Description>
  <rdf:Description rdf:about="#orderId">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
```

```
    <rdfs:domain rdf:nodeID="A18"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </rdf:Description>
  <rdf:Description rdf:about="#predictionDeviation">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
    <rdfs:domain rdf:resource="#PredictionDescription"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  </rdf:Description>
  <rdf:Description rdf:about="#ConfirmationRequest">
    <rdf:type rdf:resource="http://jade.cselt.it/beangenerator#JADE-CLASS"/>
    <rdfs:subClassOf rdf:resource="http://jade.cselt.it/beangenerator#AgentAction"/>
  </rdf:Description>
  <rdf:Description rdf:about="#ResultFailure">
    <rdfs:subClassOf rdf:resource="#ResultType"/>
    <rdf:type rdf:resource="http://jade.cselt.it/beangenerator#JADE-CLASS"/>
  </rdf:Description>
  <rdf:Description rdf:nodeID="A19">
    <rdf:rest rdf:nodeID="A14"/>
    <rdf:first rdf:resource="#CFPResponse"/>
  </rdf:Description>
  <rdf:Description rdf:nodeID="A20">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Restriction"/>
    <owl:onProperty rdf:resource="#reasonText"/>
    <owl:cardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:cardinality>
  </rdf:Description>
  <rdf:Description rdf:nodeID="A17">
    <rdf:first rdf:resource="#OrderRequest"/>
    <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
  </rdf:Description>
  <rdf:Description rdf:about="#orderDescription">
    <rdfs:range rdf:resource="#OrderDescription"/>
    <rdfs:domain rdf:nodeID="A11"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  </rdf:Description>
  <rdf:Description rdf:about="#orderConfirmation">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
    <rdfs:range rdf:resource="#OrderConfirmation"/>
    <rdfs:domain rdf:nodeID="A21"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  </rdf:Description>
  <rdf:Description rdf:about="#ResultType">
    <rdfs:subClassOf rdf:resource="#ResultInfo"/>
    <rdf:type rdf:resource="http://jade.cselt.it/beangenerator#JADE-CLASS"/>
    <owl:disjointWith rdf:resource="#ResultReason"/>
  </rdf:Description>
  <rdf:Description rdf:about="#CICResponse">
    <rdf:type rdf:resource="http://jade.cselt.it/beangenerator#JADE-CLASS"/>
    <rdfs:subClassOf rdf:resource="http://jade.cselt.it/beangenerator#AgentAction"/>
  </rdf:Description>
  <rdf:Description rdf:nodeID="A22">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
    <owl:unionOf rdf:nodeID="A23"/>
  </rdf:Description>
  <rdf:Description rdf:about="#OrderConfirmation">
    <rdfs:subClassOf rdf:resource="#Description"/>
    <rdf:type rdf:resource="http://jade.cselt.it/beangenerator#JADE-CLASS"/>
  </rdf:Description>
  <rdf:Description rdf:about="#deliveryTimeSpecific">
    <rdfs:domain rdf:resource="#OfferDescription"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
  </rdf:Description>
  <rdf:Description rdf:about="#amountRequired">
```

```
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
  <rdfs:domain rdf:resource="#OrderDescription"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
</rdf:Description>
<rdf:Description rdf:about="#offeredProducts">
  <rdfs:domain rdf:resource="#CICAgentDescription"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <rdfs:range rdf:resource="#Product"/>
</rdf:Description>
<rdf:Description rdf:about="#amountSpecific">
  <rdfs:domain rdf:resource="#OfferDescription"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
</rdf:Description>
<rdf:Description rdf:nodeID="A24">
  <rdf:first rdf:resource="#Request"/>
  <rdf:rest rdf:nodeID="A16"/>
</rdf:Description>
<rdf:Description rdf:about="#WADelivery">
  <rdfs:subClassOf rdf:resource="http://jade.cselt.it/beangenerator#AgentAction"/>
  <rdf:type rdf:resource="http://jade.cselt.it/beangenerator#JADE-CLASS"/>
</rdf:Description>
<rdf:Description rdf:about="#amountPreferred">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  <rdfs:domain rdf:resource="#OrderDescription"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</rdf:Description>
<rdf:Description rdf:nodeID="A21">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
  <owl:unionOf rdf:nodeID="A13"/>
</rdf:Description>
<rdf:Description rdf:about="#offerDescription">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdfs:domain rdf:nodeID="A25"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <rdfs:range rdf:resource="#OfferDescription"/>
</rdf:Description>
<rdf:Description rdf:nodeID="A23">
  <rdf:first rdf:resource="#OrderDescription"/>
  <rdf:rest rdf:nodeID="A8"/>
</rdf:Description>
<rdf:Description rdf:about="#Description">
  <rdfs:subClassOf rdf:resource="http://jade.cselt.it/beangenerator#Concept"/>
  <rdf:type rdf:resource="http://jade.cselt.it/beangenerator#JADE-CLASS"/>
</rdf:Description>
<rdf:Description rdf:about="#agentList">
  <rdfs:range rdf:resource="http://jade.cselt.it/beangenerator#AID"/>
  <rdfs:domain rdf:resource="#CICResponse"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</rdf:Description>
<rdf:Description rdf:about="#ResultReason">
  <rdf:type rdf:resource="http://jade.cselt.it/beangenerator#JADE-CLASS"/>
  <rdfs:subClassOf rdf:nodeID="A20"/>
  <rdfs:subClassOf rdf:resource="#ResultInfo"/>
</rdf:Description>
<rdf:Description rdf:about="#SDAPrediction">
  <rdfs:subClassOf rdf:resource="http://jade.cselt.it/beangenerator#AgentAction"/>
  <rdf:type rdf:resource="http://jade.cselt.it/beangenerator#JADE-CLASS"/>
</rdf:Description>
<rdf:Description rdf:about="#agentRank">
  <rdfs:domain rdf:resource="#LAAgentDescription"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
</rdf:Description>
```

```
  <rdf:Description rdf:nodeID="A6">
    <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
    <rdf:first rdf:resource="#DeliveryDescription"/>
  </rdf:Description>
  <rdf:Description rdf:nodeID="A12">
    <rdf:first rdf:resource="#IssueOrder"/>
    <rdf:rest rdf:nodeID="A24"/>
  </rdf:Description>
  <rdf:Description rdf:about="">
    <owl:imports rdf:resource="http://jade.cselt.it/beangenerator"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Ontology"/>
  </rdf:Description>
  <rdf:Description rdf:nodeID="A26">
    <owl:unionOf rdf:nodeID="A2"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
  </rdf:Description>
  <rdf:Description rdf:nodeID="A18">
    <owl:unionOf rdf:nodeID="A5"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
  </rdf:Description>
  <rdf:Description rdf:nodeID="A27">
    <rdf:rest rdf:nodeID="A10"/>
    <rdf:first rdf:resource="#OfferDescription"/>
  </rdf:Description>
  <rdf:Description rdf:about="#LAAgentDescription">
    <rdfs:subClassOf rdf:resource="#Description"/>
    <rdf:type rdf:resource="http://jade.cselt.it/beangenerator#JADE-CLASS"/>
  </rdf:Description>
  <rdf:Description rdf:about="#resultType">
    <rdfs:domain rdf:resource="#Result"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
    <rdfs:range rdf:resource="#ResultType"/>
  </rdf:Description>
  <rdf:Description rdf:about="#deliveryTimeAllowed">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
    <rdfs:domain rdf:resource="#OrderDescription"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  </rdf:Description>
  <rdf:Description rdf:about="#OrderDescription">
    <rdf:type rdf:resource="http://jade.cselt.it/beangenerator#JADE-CLASS"/>
    <rdfs:subClassOf rdf:resource="#Description"/>
  </rdf:Description>
  <rdf:Description rdf:nodeID="A9">
    <rdf:first rdf:resource="#CICQuery"/>
    <rdf:rest rdf:nodeID="A28"/>
  </rdf:Description>
  <rdf:Description rdf:about="#ConfirmationResponse">
    <rdf:type rdf:resource="http://jade.cselt.it/beangenerator#JADE-CLASS"/>
    <rdfs:subClassOf rdf:resource="http://jade.cselt.it/beangenerator#AgentAction"/>
  </rdf:Description>
  <rdf:Description rdf:nodeID="A3">
    <rdf:rest rdf:nodeID="A27"/>
    <rdf:first rdf:resource="#LAAgentDescription"/>
  </rdf:Description>
  <rdf:Description rdf:about="#theProduct">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
    <rdfs:domain rdf:nodeID="A22"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
    <rdfs:range rdf:resource="#Product"/>
  </rdf:Description>
  <rdf:Description rdf:nodeID="A29">
    <owl:onProperty rdf:resource="#offeredProducts"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Restriction"/>
    <owl:minCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:minCardinality>
  </rdf:Description>
```

```
<rdf:Description rdf:about="#productName">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain rdf:resource="#Product"/>
</rdf:Description>
<rdf:Description rdf:about="#CICRegister">
  <rdfs:subClassOf rdf:resource="http://jade.cselt.it/beangenerator#AgentAction"/>
  <rdf:type rdf:resource="http://jade.cselt.it/beangenerator#JADE-CLASS"/>
</rdf:Description>
<rdf:Description rdf:about="http://jade.cselt.it/beangenerator#AID">
</rdf:Description>
<rdf:Description rdf:nodeID="A28">
  <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
  <rdf:first rdf:resource="#PredictionDescription"/>
</rdf:Description>
<rdf:Description rdf:about="#agentCICDescription">
  <rdfs:domain rdf:resource="#CICRegister"/>
  <rdfs:range rdf:resource="#CICAgentDescription"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</rdf:Description>
<rdf:Description rdf:about="#CFPRequest">
  <rdfs:subClassOf rdf:resource="http://jade.cselt.it/beangenerator#AgentAction"/>
  <rdf:type rdf:resource="http://jade.cselt.it/beangenerator#JADE-CLASS"/>
</rdf:Description>
<rdf:Description rdf:about="#agentAID">
  <rdfs:range rdf:resource="http://jade.cselt.it/beangenerator#AID"/>
  <rdfs:domain rdf:nodeID="A26"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</rdf:Description>
<rdf:Description rdf:about="#InformResult">
  <rdfs:subClassOf rdf:resource="http://jade.cselt.it/beangenerator#AgentAction"/>
  <rdf:type rdf:resource="http://jade.cselt.it/beangenerator#JADE-CLASS"/>
</rdf:Description>
<rdf:Description rdf:about="#predictionDescription">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <rdfs:range rdf:resource="#PredictionDescription"/>
  <rdfs:domain rdf:resource="#SDAPrediction"/>
</rdf:Description>
<rdf:Description rdf:about="#ResultInfo">
  <rdf:type rdf:resource="http://jade.cselt.it/beangenerator#JADE-CLASS"/>
  <rdfs:subClassOf rdf:resource="http://jade.cselt.it/beangenerator#Concept"/>
</rdf:Description>
<rdf:Description rdf:about="#IssueOrder">
  <rdf:type rdf:resource="http://jade.cselt.it/beangenerator#JADE-CLASS"/>
  <rdfs:subClassOf rdf:resource="http://jade.cselt.it/beangenerator#AgentAction"/>
</rdf:Description>
<rdf:Description rdf:about="#ResultSuccess">
  <rdfs:subClassOf rdf:resource="#ResultType"/>
  <rdf:type rdf:resource="http://jade.cselt.it/beangenerator#JADE-CLASS"/>
  <owl:disjointWith rdf:resource="#ResultFailure"/>
</rdf:Description>
<rdf:Description rdf:about="#priceSpecific">
  <rdfs:domain rdf:resource="#OfferDescription"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
</rdf:Description>
<rdf:Description rdf:about="#deliveryTimeRequired">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdfs:domain rdf:resource="#OrderDescription"/>
</rdf:Description>
<rdf:Description rdf:about="#reasonText">
```

```
    <rdfs:domain rdf:resource="#ResultReason"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </rdf:Description>
  <rdf:Description rdf:nodeID="A25">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
    <owl:unionOf rdf:nodeID="A19"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://jade.cselt.it/beangenerator#AgentAction">
  </rdf:Description>
  <rdf:Description rdf:about="#PredictionDescription">
    <rdf:type rdf:resource="http://jade.cselt.it/beangenerator#JADE-CLASS"/>
    <rdfs:subClassOf rdf:resource="#Description"/>
  </rdf:Description>
  <rdf:Description rdf:about="#IssueOrderResult">
    <rdfs:subClassOf rdf:resource="#Result"/>
    <rdf:type rdf:resource="http://jade.cselt.it/beangenerator#JADE-CLASS"/>
  </rdf:Description>
  <rdf:Description rdf:about="#agentLADescriptions">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
    <rdfs:range rdf:resource="#LAAgentDescription"/>
    <rdfs:domain rdf:resource="#IssueOrder"/>
  </rdf:Description>
  <rdf:Description rdf:about="#OrderRequest">
    <rdf:type rdf:resource="http://jade.cselt.it/beangenerator#JADE-CLASS"/>
    <rdfs:subClassOf rdf:resource="http://jade.cselt.it/beangenerator#AgentAction"/>
  </rdf:Description>
  <rdf:Description rdf:about="#CICAgentDescription">
    <rdf:type rdf:resource="http://jade.cselt.it/beangenerator#JADE-CLASS"/>
    <rdfs:subClassOf rdf:resource="#Description"/>
    <rdfs:subClassOf rdf:nodeID="A29"/>
  </rdf:Description>
  <rdf:Description rdf:about="#resultReason">
    <rdfs:domain rdf:resource="#Result"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
    <rdfs:range rdf:resource="#ResultReason"/>
  </rdf:Description>
  <rdf:Description rdf:about="#priceMax">
    <rdfs:domain rdf:resource="#OrderDescription"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  </rdf:Description>
  <rdf:Description rdf:nodeID="A1">
    <rdf:rest rdf:nodeID="A15"/>
    <rdf:first rdf:resource="#OrderResult"/>
  </rdf:Description>
</rdf:RDF>
```

9.2 Example function extracting ontology object from ACLMessage.

```
private OrderDescription extractDescription(ACLMessage msg) {
   ContentElement content = null;

   try {
      try {
         content = getContentManager().extractContent(msg);
      } catch (OntologyException e) {
         AgentHelper.errorMessage(this, "Error relating to message and "
               + "ontology");
         throw new IDontUnderstand("(" + msg.toString() + ")");
      } catch (Codec.CodecException e) {
         AgentHelper.errorMessage(this,
               "Error Parsing the message format");
         throw new IDontUnderstand("(" + msg.toString() + ")");
      }

      Concept action_content;

      if (content == null || !(content instanceof Action)) {
         AgentHelper.errorMessage(this, "Message body was not an action");
         throw new IDontUnderstand("(" + msg.toString() + ")");
      }

      action_content = ((Action) content).getAction();

      if (action_content == null) {
         AgentHelper.errorMessage(this, "Message body was not present");
         throw new IDontUnderstand("(" + msg.toString() + ")");
      } else {
         if (action_content instanceof OrderRequest) {

            OrderRequest information_request = (OrderRequest) action_content;

            // get order details
            OrderDescription theDescription = information_request.getOrderDescription();
            return theDescription;
         } else {
            AgentHelper.errorMessage(this,
                  "Message body was not a correct action");
            throw new IDontUnderstand("(" + msg.toString() + ")");
         }
      }
   } catch (IDontUnderstand e) {
      ;
   }
   return null;
}
```

9.2 Sample content slot of ACLMessage exchanged between agents.

Message encoded in FIPA-SL, no headers included. The message is IssueOrder request from LA to OA.

```
((action (agent-identifier
      :name "")
    (IssueOrder
      :orderDescription
        (OrderDescription
          :deliveryTimeRequired 20
          :priceMax 10.0
          :amountRequired 0.21198610961437225
          :deliveryTimeAllowed 60
          :amountPreferred 1.1833372116088867
```

```
    :theProduct
      (Product
        :productName someproduct))
:agentLADescriptions
  (sequence
    (LAAgentDescription
      :agentRank 100
      :agentAID
        (agent-identifier
          :name wha1@computer:1099/JADE
          :addresses (sequence http://computer:7778/acc)))
    (LAAgentDescription
      :agentRank 100
      :agentAID
        (agent-identifier
          :name wha2@computer:1099/JADE
          :addresses (sequence http://computer:7778/acc)))))))
```

End of document

Warszawa, dnia 28.01.2007.

Oświadczenie

Oświadczam, że pracę magisterską pod tytułem; „ Agent-based Commodity Flow Management” , której promotorem jest dr Marcin Paprzycki wykonałem samodzielnie, co poświadczam własnoręcznym podpisem.

Tomasz Serzysko