Georgakarakou, C. E. & Economides, A.A.: **Software Agent Technology: An Overview.** *Agent and Web Service Technologies in Virtual Enterprises,* N. Protogeros (ed.), Idea Group Publ

Software Agent Technology: an Overview Application to Virtual Enterprises

Chrysanthi E. Georgakarakou & Anastasios A. Economides

Information Systems Department
University of Macedonia
Egnatia 156, 54006 Thessaloniki, GREECE
{georgak, economid}@uom.gr

1. Introduction

The aim of this chapter is to survey some key research issues in the software agents' area. It annotates several researchers' opinions on many areas concerning software agents trying to give a more documentary point of view of each argued subject. Its main goal is to provide an overview of the rapidly evolving area of software agents serving as a reference point to a large body of literature and outlining the key aspects of software agent technology. While this chapter does not act as an introduction to all the issues in the software agents' field, it intends to point the reader at the primary areas of interest. In addition to, this chapter investigates the application of agent technology to virtual enterprises. It presents basic aspects of applying agent technology to virtual enterprises serving as an introductory step.

First of all, this overview chapter attempts to answer the question of what a software agent is. Secondly, it analyzes the three technologies that DAI (distributed artificial intelligence) has evolved: i) multi-agent system (MAS), ii) distributed problem solving (DPS), and iii) parallel AI (PAI). Thereinafter, it makes the distinction between single agent and multi-agent systems analyzing their dimensions. In addition to, it goes through the broad spectrum of agent properties. Furthermore, it discusses the most acknowledged classification schemes or taxonomies (typologies) of software agents proposed in the agent research community. Moreover, it presents the most well-known agent architecture classification schemes arguing about each distinct architecture. Besides, it explores the two most important agent communication approaches: i) communication protocols, and ii) evolving languages. It also discusses about a number of languages for coordination and communication that have been proposed. It argues about possible implementations of agent transportation mechanisms as well. Further, it annotates prominent ontology specification languages and editors for ontology creation and maintenance. Then, it lists and argues standard languages and several prototype languages for implementing agent-based systems that have been proposed for constructing agent-based systems. Afterwards, it presents a number of tools and platforms that are available and support activities or phases of the process of agent-oriented software development. Next, it examines several agent oriented software engineering (AOSE) methodologies that have been proposed to assist engineers to create agent-based systems. At the end, it investigates the application of the agent technology to virtual enterprises, answering the question of why to use agents in virtual enterprises and presenting the current research activity that focuses on the agent technology applied to virtual enterprises.

2. Background

As software agents comprise a prominent scientific area of research activity, a plethora of researchers have investigated them and stated their own point of view. Nwana & Ndumu (1996) mention that software agent technology is a rapidly developing area of research. According to Wooldridge & Jennings (1995), the concept of an agent has become important in both artificial intelligence (AI) and mainstream computer science. Oliveira *et al.* (1999) observe that for some time now agent-based and multi-agent systems (MASs) have attracted the interest of researchers far beyond traditional computer science and artificial intelligence (AI).

Although software agent technology demonstrates expeditious advancement, there is a truly heterogeneous body of work being carried out under the 'agents' banner (Nwana & Ndumu, 1996). Nwana & Ndumu (1996) introduce software agent technology by overviewing the various agent types currently under investigation by researchers. Nwana (1996) largely reviews software agents, and makes some strong statements that are not necessarily widely accepted by the agent community. Nwana (1996) presents a typology of agents, next places agents in context, defines them and overviews critically the rationales, hypotheses, goals, challenges and state-of-the-art demonstrators of the various agent types of the proposed typology. Besides, Nwana (1996) attempts to make explicit much of what is usually implicit in the agents' literature and proceeds to overview some other general issues which pertain to all the types of agents in the typology.

Agent-based and multi-agent systems (MASs) have attracted the researchers' interest to great extents. Oliveira *et al.* (1999) try to identify focal points of interest for researchers working in the area of distributed AI (DAI) and MAS as well as application oriented researchers coming from related disciplines, e.g. electrical and mechanical engineering. They do this by presenting key research topics in DAI and MAS research and by identifying application domains in which the DAI and MAS technologies are most suitable.

Sycara (1998) presents some of the critical notions in MASs, the research work that has addressed them and organizes these notions around the concept of problem-solving coherence. Sycara (1998) believes that problem-solving coherence is one of the most critical overall characteristics that a MAS should exhibit.

Jennings *et al.* (1998) provide an overview of research and development activities in the field of autonomous agents and multi-agent systems. They aim to identify key concepts and applications, and to indicate how they relate to one-another. Some historical context to the field of agent-based computing is given, and contemporary research directions are presented (Jennings *et al.*, 1998). Finally, a range of open issues and future challenges are highlighted (Jennings *et al.*, 1998).

Wooldridge & Jennings (1995) aim to point the reader at what they perceive to be the most important theoretical and practical issues associated with the design and construction of intelligent agents. For convenience, they divide these issues into three areas (agent theory, agent architectures and agent languages). Their paper is not intended to serve as a tutorial introduction to all the issues mentioned and includes a short review of current and potential applications of agent technology.

Wooldridge (1998) provides an introductory survey of agent-based computing. The article begins with an overview of micro-level issues in agent-based systems: issues related to the design and construction of individual intelligent agents. The article then goes on to discuss some macro-level issues: issues related to the design and construction of agent societies. Finally, the key application areas for agent technology are surveyed (Wooldridge, 1998).

An article that should not be omitted at this point is Weiß's (2002) paper. Weiß (2002) offers a guide to the broad body of literature of agent-oriented software engineering (AOSE). The guide, which is intended to be of value to both researchers and practitioners, is structured according to key issues and key topics that arise when dealing with AOSE: methods and frameworks for requirements engineering, analysis, design, and implementation; languages for programming, communication and coordination, and ontology specification; and development tools and platforms.

On the other hand, considering the agent technology application to virtual enterprises, Jennings, Norman *et al.* (1998) exhibit considerable concepts. They argue the case of the agent-based approach showing how agent technology can improve efficiency by ensuring that business activities are better scheduled, executed, monitored, and coordinated.

According to Camarinha-Matos (2002), multi-agent systems represent a promising approach to both model and implement the complex supporting infrastructures required for virtual enterprises and related emerging organizations. The current status of application of this approach to industrial virtual enterprises, virtual communities, and remote supervision in the context of networked collaborative organizations is presented (Camarinha-Matos, 2002). Examples of relevant projects are provided and major challenges and open issues identified as well (Camarinha-Matos, 2002).

Petersen *et al.* (2001) describe how virtual enterprises can be modelled using the AGORA multi-agent architecture, designed for modelling and supporting cooperative work among distributed entities. They underline that the distributed and goal-oriented nature of the virtual enterprise provides a strong motivation for the use of agents to model virtual enterprises. They also mention the main advantages of their approach.

This chapter provides an overview of research activity regarding the scientific domain of software agents. As the field of software agents can appear chaotic, this chapter briefly introduces the key issues rather than present an in-depth analysis and critique of the field. References to more detailed treatments are provided. The purpose of this chapter is to make a list of the most important themes concerning software agents, apposing some order and consistency and serve as a reference point to a large body of literature. In addition to, this chapter makes an introduction of applying agent technology to virtual enterprises and describes current research activity that addresses the above mentioned issue.

3. A Brief Overview of Software Agent Technology

3.1. What is a Software Agent?

Software agent technology is a rapidly developing area of research and probably the fastest growing area of information technology (IT) (Nwana & Ndumu, 1996; Jennings & Wooldridge, 1996). Application domains in which agent solutions are being applied or researched into include workflow management, telecommunications network management, air-traffic control, business process reengineering, data mining, information retrieval/management, electronic commerce, education, personal digital assistants (PDAs), e-mail filtering, digital libraries, command and control, smart databases, and scheduling/diary management (Nwana & Ndumu, 1996).

Over the last years, many researchers in the area of agents have proposed a large variety of definitions for the agent term. It is stated that it is difficult to give a full definition for the note of agency. Nwana (1996) predicates there are at least two reasons why it is so difficult to define precisely what agents are. Firstly, agent researchers do not 'own' this term in the same way as fuzzy logicians/AI researchers, for example, own the term 'fuzzy logic' - it is one that is used widely in everyday parlance as in travel agents, estate agents, etc. Secondly, even within the software fraternity, the word 'agent' is really an umbrella term for a heterogeneous body of research and development (Nwana, 1996). Concerning the agent definition, Nwana (1996) states that 'W*hen we really have to, we define an agent as referring to a component of software and/or hardware which is capable of acting exactingly in order to accomplish tasks on behalf of its user. Given a choice, we would rather say it is an umbrella term, meta-term or class, which covers a range of other more specific agent types, and then go on to list and define what these other agent types are. This way, we reduce the chances of getting into the usual prolonged philosophical and sterile arguments which usually proceed the former definition, when any old software is conceivably recastable as agent-based software*' (p. 6).

Bradshow (1997) identifies two approaches to the definition of an agent as follows: i) agent as an ascription: this approach is based on the concept that "agency cannot ultimately be characterized by listing a collection of attributes but rather consists fundamentally as an attribution on the part of some person", and ii) agent as a description: agents are defined by describing the attributes they should exhibit.

Jennings & Wooldridge (1996) offer a relatively loose notion of an agent as a self-contained program capable of controlling its own decision making and acting, based on its perception of its environment, in pursuit of one or more objectives will be used here.

Wooldridge (1998) defines an intelligent agent as a system that enjoys the following four properties: autonomy (agents operate without the direct intervention of humans or others, and have control over their actions and internal state), social ability (agents are able to cooperate with humans or other agents in order to achieve their tasks), reactivity (agents perceive their environment, and respond in a timely fashion to changes that occur in it), and pro-activeness (agents do not simply act in response to their environment, they are able to exhibit goal-directed behaviour by taking the initiative).

According to Hayes (1999), an agent is an entity (either computer, or human) that is capable of carrying out goals, and is part of a larger community of agents that have mutual influence on each other. Agents may co-exist on a single processor, or they may be constructed from physically, but intercommunicating processors (such as a community of robots) (Hayes, 1999). The key concepts in this definition are that agents can act autonomously to some degree, and they are part of a community in which mutual influence occurs (Hayes, 1999).

3.2. Distributed Artificial Intelligence (DAI) Technologies

Distributed artificial intelligence (DAI) is a sub-field of artificial intelligence (AI) which is concerned with a society of problem solvers or agents interacting in order to solve a common problem: computers and persons, sensors, aircraft, robots, etc (Green *et al.*, 1997). Such a society is termed a multi-agent system, namely, a network of problem solvers that work together to solve problems that are beyond their individual capabilities (Green *et al.*, 1997). Software agents have evolved from multi-agent systems (MAS), which in turn form one of three broad areas which fall under DAI, the other two being distributed problem solving (DPS) and parallel AI (PAI) (Nwana, 1996). Therefore, agents inherit potential benefits from both DAI e.g. modularity, speed, reliability and AI e.g. operation at knowledge level, easier maintenance, reusability, platform independence (Nwana, 1996).

3.3. Agent Systems

Jennings *et al.* (1998) state that an agent-based system is a system in which the key abstraction used is that of an agent. In principle, an agent-based system might be conceptualised in terms of agents, but implemented without any software structures corresponding to agents at all (Jennings *et al.,* 1998). A parallel with object-oriented software can be drawn, where it is entirely possible to design a system in terms of objects, but to implement it without the use of an object-oriented software environment (Jennings *et al.,* 1998). But this would at best be unusual, and at worst, counter-productive (Jennings *et al.,* 1998). According to Jennings *et al.* (1998), a similar situation exists with agent technology and they therefore expect an agent-based system to be both designed and implemented in terms of agents.

An agent-based system may contain one or more agents (Jennings *et al.,* 1998). There are cases in which a single agent solution is appropriate (Jennings *et al.,* 1998). However, the multi-agent case — where the system is designed and implemented as several interacting agents — is arguably more general and more interesting from a software engineering standpoint (Jennings *et al.,* 1998). Multi-agent systems are ideally suited to representing problems that have multiple problem solving methods, multiple perspectives and/or multiple problem solving entities (Jennings *et al.,* 1998). Such systems have the traditional advantages of distributed and concurrent problem solving, but have the additional advantage of sophisticated patterns of interactions (Jennings *et al.,* 1998). Examples of common types of interactions include: cooperation (working together towards a common aim), coordination (organising problem solving activity so that harmful interactions are avoided or beneficial interactions are exploited), and negotiation (coming to an agreement which is acceptable to all the parties involved) (Jennings *et al.,* 1998).

As the technology matures and endeavors to attack more complex, realistic, and large-scale problems, the need for systems that consist of multiple agents that communicate in a peer-to-peer fashion is becoming apparent (Sycara, 1998). The most powerful tools for handling complexity are modularity and abstraction (Sycara, 1998). Multi-agent systems (MASs) offer modularity (Sycara, 1998). If a problem domain is particularly a complex, large, or unpredictable, then the only way it can reasonably be addressed is to develop a number of functionally specific and (nearly) modular components (agents) that are specialized at solving a particular problem aspect (Sycara, 1998). In MASs, applications are designed and developed in terms of autonomous software entities (agents) that can flexibly achieve their objectives by interacting with one another in terms of high-level protocols and languages (Zambonelli *et al.,* 2003). A MAS can be defined as a collection of, possibly heterogeneous, computational entities, having their own problem solving capabilities and which are able to interact among them in order to reach an overall goal (Oliveira *et al.*, 1999).

3.4. Agent Properties

A software agent is a computer system situated in an environment that acts on behalf of its user and is characterised by a number of properties (Chira, 2003). Most researchers agree that autonomy is a crucial property of an agent. Alonso (2002) states about agents that it is precisely their autonomy that defines them. Furthermore, cooperation among different software agents may be very useful in achieving the objectives an agent has (Chira, 2003). According to the weak notion of agency given by Wooldridge & Jennings (1995) the most general way in which the term agent is used is to denote hardware or (more usually) software-based computer system that enjoys the following properties: autonomy, social ability, reactivity and pro-activeness (Wooldridge & Jennings, 1995). Jennings *et al.* (1998) identify three key concepts in their definition that they adapt from (Wooldridge & Jennings, 1995): situatedness, autonomy, and flexibility (by the term flexible they mean that the system is responsive, pro-active and social). For Wooldridge (1998), an intelligent agent is a system that enjoys autonomy, social ability, reactivity and pro-activeness. He also refers to the fact that other researchers argue that different properties, such as mobility, veracity, benevolence, rationality and learning, should receive greater emphasis.

An agent may possess many properties in various combinations. Table 1 enumerates and defines all the properties that we adopt for the purposes of this research.

Property	Definition
1. Autonomy	It means that the agent can act without direct intervention by humans or other agents and that it has control over its own actions and internal state (Sycara, 1998).
2. Reactivity or situatedness or sensing and acting	It means that the agent receives some form of sensory input from its environment, and it performs some action that changes its environment in some way (Chira, 2003; Sycara, 1998).
3. Proactiveness or goal directed behaviour	It means that the agent does not simply act in response to its environment; it is able to exhibit goal-directed behaviour by taking the initiative (Chira, 2003; Wooldridge & Jennings, 1995; Odell, 2000).
4. Social ability	It means that the agent interacts and this interaction is marked by friendliness or pleasant social relations; that is, the agent is affable, companionable or friendly (Odell, 2000).
5. Coordination	It means that the agent is able to perform some activity in a shared environment with other agents (Odell, 2000). Activities are often coordinated via plans, workflows, or some other process management mechanism (Odell, 2000).
6. Cooperation or collaboration	It means that the agent is able to coordinate with other agents to achieve a common purpose; non-antagonistic agents that succeed or fail together (Odell, 2000).
7. Flexibility	It means that the system is responsive (the agents should perceive their environment and respond in a timely fashion to changes that occur in it), pro-active and social (Jennings *et al.,* 1998).
8. Learning or adaptivity	It means that an agent is capable of i) reacting flexibly to changes in its environment; ii) taking goal-directed initiative, when appropriate; and iii) learning from its own experience, its environment, and interactions with others (Chira, 2003; Sycara, 1998).
9. Mobility	It means that the agent is able to transport itself from one machine to another and across different system architectures and platforms (Etzioni & Weld, 1995).
10. Temporal continuity	It means that the agent is a continuously running process, not a "one-shot" computation that maps a single input to a single output, then terminates (Etzioni & Weld, 1995).
11. Personality or character	An agent has a well-defined, believable "personality" and emotional state (Etzioni & Weld, 1995).
12. Reusability	Processes or subsequent instances can require keeping instances of the class 'agent' for an information handover or to check and to analyze them according to their results (Horn *et al.*, 1999).
13. Resource limitation	An agent can only act as long as it has resources at its disposal (Horn *et al.*, 1999). These resources are changed by its acting and possibly also by delegating (Horn *et al.*, 1999).
14. Veracity	It is the assumption that an agent will not knowingly communicate false information (Wooldridge & Jennings, 1995; Wooldridge 1998).
15. Benevolence	It is the assumption that agents do not have conflicting goals and that every agent will therefore always try to do what is asked of it (Wooldridge & Jennings, 1995; Wooldridge 1998).
16. Rationality	It is the assumption that an agent will act in order to achieve its goals, and will not act in such a way as to prevent its goals being achieved — at least insofar as its beliefs permit (Wooldridge & Jennings, 1995; Wooldridge 1998).
17. Inferential capability	An agent can act on abstract task specification using prior knowledge of general goals and preferred methods to achieve flexibility; goes beyond the information given, and may have explicit models of self, user, situation, and/or other agents (Bradshow, 1997).
18. "Knowledge-level" communication ability	The ability to communicate with persons and other agents with language more resembling humanlike "speech acts" than typical symbol-level program-to-program protocols (Bradshow, 1997).
19. Prediction ability	An agent is predictive if its model of how the world works is sufficiently accurate to allow it to correctly predict how it can achieve the task (Goodwin,

	1993).
20. Interpretation ability	An agent is interpretive if can correctly interpret its sensor readings (Goodwin, 1993).
21. Sound	An agent is sound if it is predictive, interpretive and rational (Goodwin, 1993).
22. Proxy ability	An agent can act on behalf of someone or something that is, acting in the interest of, as a representative of, or for the benefit of, some entity (Odell, 2000).
23. Intelligence	The agent's state is formalized by knowledge and interacts with other agents using symbolic language (Odell, 2000).
24. Unpredictability	An agent is able to act in ways that are not fully predictable, even if all the initial conditions are known (Odell, 2000). It is capable of nondeterministic behaviour (Odell, 2000).
25. Credibility	An agent has a believable personality and emotional state (Odell, 2000).
26. Transparency and accountability	An agent must be transparent when required, but must provide a log of its activities upon demand (Odell, 2000).
27. Competitiveness	An agent is able to coordinate with other agents except that the success of one agent implies the failure of others (Odell, 2000).
28. Ruggedization	An agent is able to deal with errors and incomplete data robustly (Odell, 2000).
29. Trustworthiness	An agent adheres to laws of robotics and is truthful (Odell, 2000).

Table 1: Agent properties

3.5. Agent Typology

Agents may be usefully classified according to the subset of these properties that they enjoy (Franklin & Graesser, 1996). There are, of course, other possible classifying schemes (Franklin & Graesser, 1996). For example, software agents might be classified according to the tasks they perform, for example, information gathering agents or email filtering agents (Franklin & Graesser, 1996). Or, they might be classified according to their control architecture (Franklin & Graesser, 1996). Agents may also be classified by the range and sensitivity of their senses, or by the range and effectiveness of their actions, or by how much internal state they possess (Franklin & Graesser, 1996).

There are several classification schemes or taxonomies proposed in the agent research community from which the following three are well acknowledged (Chira, 2003):
i) Gilbert's scope of intelligent agents (Bradshow, 1997).
ii) Nwana's primary attributes dimension typology (Nwana, 1996).
iii) Franklin and Graesser's agent taxonomy (Franklin & Graesser, 1996).

A typology refers to the study of types of entities and there are several dimensions to classify existing software agents (Nwana, 1996). Agents may be classified according to (Bradshow, 1997):
i) Mobility, as static or mobile,
ii) Presence of a symbolic reasoning model, as deliberative or reactive,
iii) Exhibition of ideal and primary attributes, such as autonomy, cooperation, and learning,
iv) Roles, as information or Internet,
v) Hybrid philosophies, which combine two or more approaches in a single agent, and
vi) Secondary attributes, such as versatility, benevolence, veracity, trustworthiness, temporal continuity, ability to fail gracefully, and mentalistic and emotional qualities (Nwana, 1996).

Nwana (1996) identifies seven types of agents (Chira, 2003). Table 2 enumerates and describes each agent type.

Agent type	Description
1. Collaborative agents	They are "able to act rationally and autonomously in open and time-constrained multi-agent environments" (Chira, 2003; Nwana, 1996). Key characteristics: autonomy, social ability, responsiveness, and pro-activeness (Chira, 2003; Nwana, 1996).
2. Interface agents	They support and assist the user when interacting with one or more computer

	applications by learning during the collaboration process with the user and with other software agents (Chira, 2003; Nwana, 1996). Key characteristics: autonomy, learning (mainly from the user but also from other agents), and cooperation with the user and/or other agents (Chira, 2003; Nwana, 1996).
3. Mobile agents	They are autonomous software programs capable of roaming wide area networks (such as WWW) and cooperation while performing duties (e.g. flight reservation, managing a telecommunications' network) on behalf of its user (Chira, 2003; Nwana, 1996). Key characteristics: mobility, autonomy, and cooperation (with other agents – for example, to exchange data or information) (Chira, 2003; Nwana, 1996).
4. Information/internet agents	They are designed to manage, manipulate or collate the vast amount of information available from many distributed sources (information explosion) (Chira, 2003; Nwana, 1996). These agents "have varying characteristics: they may be static or mobile; they may be non-cooperative or social; and they may or may not learn" (Chira, 2003; Nwana, 1996).
5. Reactive agents	They act/respond to the current state of their environment based on a stimulus-response scheme (Chira, 2003; Nwana, 1996). These agents are relatively simple and interact with other agents in basic ways but they have the potential to form more robust and fault tolerant agent-based systems (Chira, 2003; Nwana, 1996). Key characteristics: autonomy and reactivity (Chira, 2003; Nwana, 1996).
6. Hybrid agents	They combine two or more agent philosophies into a single agent in order to maximise the strengths and minimise the deficiencies of the most relevant techniques (for a particular purpose) (Chira, 2003; Nwana, 1996).
7. Smart Agents	They are equally characterised by autonomy, cooperation, and learning (Chira, 2003; Nwana, 1996).

Table 2: Agent types

According to Nwana (1996), there are some applications which combine agents from two or more of the above types. Nwana (1996) refers to these as heterogeneous agent systems. This category of agent systems is generally referred to (by most researchers) as multi-agent systems (Chira, 2003).

3.6. Agent Architectures

Researchers working in the area of agents' architectures are concerned with the design and construction of agents that enjoy the properties of autonomy, reactivity, pro-activeness, and social ability (Wooldridge, 1998). Wooldridge (1999) states that agent architecture is essentially a map of the internals of an agent — its data structures, the operations that may be performed on these data structures, and the control flow between these data structures**.** Three classes of agent architectures can be identified (Wooldridge & Jennings, 1995):
i) Deliberative or symbolic architectures are those designed along the lines proposed by traditional, symbolic AI,
ii) Reactive architectures are those that eschew central symbolic representations of the agent's environment, and do not rely on symbolic reasoning, and
iii) Hybrid architectures are those that try to marry the deliberative and reactive approaches (Wooldridge, 1998).
Wooldridge & Jennings (1995) indicate that agent architectures can be viewed as software engineering models of agents and identify the above mentioned classes of agent architectures.

Wooldridge (1999) considers four classes of agents. Table 3 enumerates and gives a short description of each class. In our opinion most agents follow one of the above four architectural classes.

Agent class	Description
1. Logic based agents	In which decision making is realised through logical deduction (Wooldridge, 1999).
2. Reactive agents	In which decision making is implemented in some form of direct mapping from situation to action (Wooldridge, 1999).
3. Belief-desire-intention (BDI) agents	In which decision making depends upon the manipulation of data structures representing the beliefs, desires, and intentions of the agent (Wooldridge, 1999).
4. Layered architectures	In which decision making is realised via various software layers, each of

	which is more-or-less explicitly reasoning about the environment at different levels of abstraction (Wooldridge, 1999).

Table 3: Agent classes

3.7. Agent Communication Approaches

One of the most important features of an agent is interaction. In other words, agents recurrently interact to share information and to perform tasks to achieve their goals (Kostakos & Taraschi, 2001). Without communication, different agents cannot know from each other who is doing what and how they can cooperate (Bussink, 2004). Therefore communication is a must if we want to set up a useful multi-agent system (Bussink, 2004).

There are several approaches to how this communication can take shape (Bussink, 2004). The two most important approaches are communication using communication protocols and communication using an evolving language (Bussink, 2004). Both techniques have their advantages and disadvantages (Bussink, 2004). In industrial applications communication protocols will be the best practice, but in systems where homogeneous agents can work together language evolution is a good option (Bussink, 2004). The basis for language evolution is in human communication (Bussink, 2004). The agent languages consist of grammars and vocabularies, just like any human language (Bussink, 2004). Some researchers even do research in the area of language evolution using agents in order to get more understanding of how human communication has evolved (Bussink, 2004). For a long time, the only way agents communicated was using communication protocols (Bussink, 2004). Therefore research often focussed on this area and a lot of specifications have been written (Bussink, 2004). Because of the formal nature of protocols, there are a quite a few widely known and used standards (Bussink, 2004).

3.8. Agent Communication Languages (ACLs)

The difficulty to precisely handle coordination and communication increases with the size of the agent-based software to be developed. A number of languages for coordination and communication have been proposed. Weíβ (2002) enumerates a list of such languages. Table 4 enumerates and describes the most prominent examples of agent communication languages (ACLs) according to Weíβ (2002).

Agent communication language	Description
1. KQML ("Knowledge Query and Manipulation Language")	It is perhaps the most widely used agent communication language (Weíβ, 2002).
2. ARCOL ("ARTIMIS COmmunication Language")	It is the communication language used in the ARTIMIS system (Weíβ, 2002). ARCOL has a smaller set of communication primitives than KQML, but these can be composed (Weíβ, 2002).
3. FIPA-ACL ("FIPA Agent Communication Language")	It is an agent communication language that is largely inuenced by ARCOL (Weíβ, 2002). Together FIPA-ACL, ARCOL, and KQML establish a quasi standard for agent communication languages (Weíβ, 2002).
4. KIF ("Knowledge Interchange Format")	It is a logic-based language that has been designed to express any kind of knowledge and meta-knowledge (Weíβ, 2002). KIF is a language for content communication, whereas languages like KQML, ARCOL, and FIPA-ACL are for intention communication (Weíβ, 2002).
5. COOL ("Domain independent COOrdination Language")	It aims at explicitly representing and applying coordination knowledge for multi-agent systems and focuses on rule-based conversation management (Weíβ, 2002). Languages like COOL can be thought of as supporting a coordination/communication (or "protocol-sensitive") layer above intention communication (Weíβ, 2002).

Table 4: Most prominent agent communication languages

Apart from these most prominent languages, several others showing unique properties have been proposed (Weíβ, 2002). Table 5 enumerates some of the above mentioned languages.

Agent communication language

1. ICL (“Interagent Communication Language”) (Weíβ, 2002)
2. AgentTalk (Weíβ, 2002)
3. CoLa (“Communication and coordination Language”) (Weíβ, 2002)
4. TuCSoN (“Tuple Centres Spread over Networks”) (Weíβ, 2002)
5. LuCe (Weíβ, 2002)
6. STL++ (“Simple Thread Language ++”) (Weíβ, 2002)
7. SDML (“Strictly Declarative Modelling Language”) (Weíβ, 2002)

Table 5: Agent communication languages showing unique properties

3.9. Agent Transportation Mechanisms

In agent environments, messages should be schedulable, as well as event driven (OMG Agent Working Group, 2000). They can be sent in synchronous or asynchronous modes (OMG Agent Working Group, 2000). The transportation mechanism should support unique addressing as well as role-based addresses (OMG Agent Working Group, 2000). Lastly, the transportation mechanism must support unicast, multicast, and broadcast modes and such services as broadcast behaviour, non-repudiation of messages, and logging (OMG Agent Working Group, 2000). Table 6 enumerates and describes possible implementations of the agent transportation mechanism.

Implementation of agent transportation mechanism	Description
1. CORBA (“Common Object Request Broker Architecture”)	It is the acronym for Common Object Request Broker Architecture, OMG's open, vendor-independent architecture and infrastructure that computer applications use to work together over networks (URL1). Using the standard protocol IIOP, a CORBA-based program from any vendor, on almost any computer, operating system, programming language, and network, can interoperate with a CORBA-based program from the same or another vendor, on almost any other computer, operating system, programming language, and network (URL1).
2. OMG (“ Object Management Group”) Messaging Services	OMG is an international trade association incorporated as a non-profit in the United States (URL2). The OMG is currently specifying a new messaging service (URL3).
3. JAVA Messaging Service	It is the standard API for sending and receiving messages (URL4).
4. RMI (“Remote Method Invocation”)	It defines and supports a distributed object model for the Java language hiding the ORB from the programmer and providing an API for the development of distributed applications (Bracho *et al.*, 1999). Java Remote Method Invocation (Java RMI) enables the programmer to create distributed Java technology-based to Java technology-based applications, in which the methods of remote Java objects can be invoked from other Java virtual machines, possibly on different hosts (URL5).
5. DCOM (“Distributed Component Object Model”)	Microsoft® Distributed COM (DCOM) extends the Component Object Model (COM) to support communication among objects on different computers—on a LAN, a WAN, or even the Internet (URL6). With DCOM, your application can be distributed at locations that make the most sense to your customer and to the application (URL6).
6. Enterprise Java Beans Events	The newest Java component model is Enterprise Java Beans (URL7). Besides its name, the Java language and the idea of component based software re-use; it has little or no similarities with the Java Beans standard (URL7). Enterprise Java Beans are located on the server and they support a distributed programming model that could be described as a flexible, two-way, object-oriented version of traditional client-server programming (URL7).

Table 6: Implementations of agent transportation mechanism

3.10. Ontology Languages and Editors

Besides an ACL, a common understanding of the concepts used among agents is necessary for a meaningful agent communication. A common ontology is required for representing the knowledge from various domains of discourse (OMG Agent Working Group, 2000). The ACL remains just syntax without a shared common ontology containing the terms used in agent communication and the knowledge associated with them (Nwana & Wooldridge, 1996). Table 7 enumerates and describes the most elaborated examples of such languages according to Weíβ (2002).

Ontology languages	Description
1. Ontolingua and Frame Logic	They are frame-based languages (Weíβ, 2002). Both of them extend first-order predicate logics (Weíβ, 2002). The key modelling primitive of these languages are frames as known from artificial intelligence (Weíβ, 2002).
2. CLASSIC and LOOM	They are description logics that allow an intentional definition of concepts (Weíβ, 2002).
3. CycL	It extends first-order predicate logic and was developed to enable the specification of large common-sense ontologies (Weíβ, 2002).

Table 7: Most elaborated examples of ontology languages

Table 8 enumerates and describes the most prominent ontology specification languages that are conform to syntactic and semantic Web standards according to Weíβ (2002).

Ontology languages	Description
1. SHOE ("Simple HTML Ontology Extension")	It is a language that slightly extends HTML and enables a hierarchical classification of HTML documents and the specification of relationships among them (Weíβ, 2002).
2. XOL ("Ontology Exchange Language")	It is an XML- and frame-based language for the exchange of ontologies (Weíβ, 2002).
3. OIL ("Ontology Inference Layer")	It aims at unifying formal semantics as offered by description logics, rich modelling primitives as offered by frame-based languages, and the XML and RDF web standards (Weíβ, 2002). OIL can be seen as an extension of XOL offering both an XML-based and an RDF-based syntax (Weíβ, 2002).
4. DAML-ONT and DAML-OIL	They are the DAML (DARPA Agent Markup Language) languages (Weíβ, 2002). DAML-OIL, which replaces DAML-ONT and represents the state of the art in the field, has well defined model-theoretic and axiomatic semantics (Weíβ, 2002).

Table 8: Ontology languages that are conform to syntactic and semantic Web standards

Table 9 enumerates and describes three good examples of editors for ontology creation and maintenance according to Weíβ (2002).

Editor	Description
1. Protégé	It supports single-user ontology acquisition (Weíβ, 2002).
2. Webonto	It supports multiple-user ontology acquisition over the Web (Weíβ, 2002).
3. OntoEdit	It supports multilingual development of ontologies and multiple inheritance (Weíβ, 2002).

Table 9: Editors for ontology creation and maintenance

3.11. Languages for constructing Agent-based systems

Most agent systems are probably written in Java and C/C++ (Weíβ, 2002). Apart from these standard languages, several prototype languages for implementing agent-based systems have been proposed that all aim at enabling a programmer to better realize agent-specific conceptions (Weiß, 2002). Three paradigms for implementing agent systems have been proposed: agent-oriented programming, market-oriented programming and interaction-oriented programming (Weiß, 2002). Weíβ (2002) lists some of most prominent and best understood prototype languages following the agent oriented paradigm (references to these languages are provided in (Weiß, 2002). Table 10 enumerates and describes the above mentioned prototype languages.

Constructing agent-based systems' languages	Description
1. AGENT-0, PLACA and AGENT-K	AGENT-0 realizes the basic ideas of the agent-oriented programming paradigm as formulated by Shoham (Weíβ, 2002). A language that

	extends AGENT-0 toward planning is PLACA, and a language that aims at integrating AGENT-0 and KQML is AGENT-K (Weíβ, 2002).
2. Concurrent MetateM	It allows specifying the intended behaviour of an agent based on temporal logics (Weíβ, 2002).
3. AgentSpeak(L)	It is a rule-based language that has a formal operational semantics and that assumes agents to consist of intentions, beliefs, recorded events, and plan rules (Weíβ, 2002). AgentSpeak(L) is based on an abstraction of the PRS architecture (Weíβ, 2002).
4. 3APL	It incorporates features from imperative and logic programming (Weíβ, 2002). 3APL has a well defined operational semantics and supports monitoring and revising of agent goals (Weíβ, 2002).
5. ConGolog	It is a concurrent logic-based language initially designed for high-level robot programming (Weíβ, 2002).
6. April ("Agent PRocess Interaction Language"), MAIL/MAI2L ("Multiagent Interaction and ImplementationLanguage"), and VIVA	Other examples of languages following the agent-oriented programming paradigm (Weíβ, 2002).

Table 10: Constructing agent-based systems' languages following the agent oriented paradigm

Nwana & Wooldridge (1996) classify constructing agent application languages according to a typology. Table 11 depicts the above mentioned classification (Nwana & Wooldridge, 1996).

Constructing agent application languages	Typology	Description
1. Actors	Collaborative agents	Actor language
2. Agent-0 and Placa	Collaborative agents	Agent-oriented programming languages
3. TCL/Tk, Safe-TCL, Safe-Tk, Java, Telescript, Active web tools, Python, Obliq, April and Scheme-48	Interface, Information and mobile agents	Scripting languages
4. RTA/ABLE	Reactive agents	Reactive language

Table 11: Typology classification of constructing agent application languages

However traditional languages are still used to construct agent applications (Nwana & Wooldridge, 1996). It is possible to implement agent-based systems in languages like Pascal, C, Lisp, or Prolog (Nwana & Wooldridge, 1996). But as a rule, one would not choose to do so because such languages are not particularly well-suited to the job (Nwana & Wooldridge, 1996). Typically, object-oriented languages such as Smalltalk, Java, or C++ lend themselves more easily for the construction of agent systems (Nwana & Wooldridge, 1996). This is because the concept of an "agent" is not too distant from that of an "object": agents share some properties with objects such as encapsulation, and frequently, inheritance and message passing (Nwana & Wooldridge, 1996). However, agents differ distinctly from objects vis-à-vis polymorphism (Nwana & Wooldridge, 1996).

3.12. Tools and Platforms

A number of tools and platforms are available that support activities or phases of the process of agent-oriented software development (Weiβ, 2002). Most of them are built on top of and integrated with Java (Weiβ, 2002). While almost all available tools and platforms have their focus on implementation support, some of them do also support analysis, design, and test/debugging activities (Weiβ, 2002).

Weíβ (2002) makes a list of such tools and platforms separating them into often sited academic and research prototypes and into commercial products for development support. Table 12 depicts the above mentioned classification. References to the following tools and a brief description as well can be found in (Weiβ, 2002).

Tools and platforms	Focusing area of interest

1. ZEUS, JADE ("Java Agent DEvelopment framework"), LEAP ("Lightweight Extensible Agent Platform"), agenTool, RETSINA, JATLite ("Java Agent Template, Lite"), FIPA-OS, MADKIT, SIM_AGENT, JAFMAS ("Java-based Agent Framework for Multi-Agent Systems"), ABS ("Agent Building Shell"), OAA ("Open Agent Architecture"), and Agentis	Academic and research activity
2. AgentBuilder, JACK, Intelligent Agent Factory and Grasshopper	Commercial activity

Table 12: Classification of tools and platforms supporting activities or phases of the process of agent-oriented software development

Serenko & Detlor (2002) state about the term agent toolkit that each vendor uses its own explanation of the term and for the needs of their report define an agent toolkit as any software package, application or development environment that provides agent builders with a sufficient level of abstraction to allow them to implement intelligent agents with desired attributes, features and rules. Some toolkits may offer only a platform for agent development, whereas others may provide features for visual programming (Serenko & Detlor, 2002). Serenko & Detlor (2002) categorize the available agent toolkits on the market into four major categories. Table 13 depicts the four categories and the representative toolkits of each category (Serenko & Detlor, 2002).

Agent toolkits	Category
1. Concordia, Gossip, FarGo, and IBM Aglets	Mobile agent toolkits
2. MadKit, Zeus, JADE, JATLite, and MAST	Multi-agent toolkits
3. FIPA-OS and Ascape	General purpose toolkits
4. Microsoft Agent, Voyager and NetStepper	Internet agent toolkits

Table 13: Categorization of agent toolkits according to their focusing area

3.13. Agent-Oriented Software Engineering (AOSE) Methodologies

Agent researchers have produced methodologies to assist engineers to create agent-based systems (URL8). Some researchers have taken agent theory as their starting point and have produced methodologies that are rooted in that theory (URL8). Other researchers have taken object techniques as their point of departure and have enriched them to be suitable for agents (URL8). Others have taken knowledge engineering concepts and extended them (URL8). Researchers also have tried to assemble methodologies by combining features from different methodologies (URL8). Yet other researchers have produced methodologies based on both agent and object technologies (URL8).

Methologies having as background the agent and multi-agent technology are characterized by a clear focus on capturing social-level abstractions such as agent, group, or organization, that is, on abstractions that are above the conventional object level (Weiβ, 2002). Methologies having as background the object orientation are characterized by the attempt to appropriately extend existing object-oriented techniques such that they also capture the notion of agency (Weiβ, 2002). Methologies having as background Knowledge engineering are characterized by an emphasis on the identification, acquisition and modelling of knowledge to be used by the agent components of a software system (Weiβ, 2002). Table 14 depicts the most popular approaches of each disciplinary background (Weiβ, 2002).

Agent-oriented software engineering methology	Disciplinary background on which the different approaches are based	Description
1. Gaia ("Generic Architecture for Information Availability")	Agent and multi-agent technology	This is a method that distinguishes between analysis and design and associates different models with these two phases (Weiβ, 2002). Gaia focuses on organizational aspects in terms of concepts such as roles, interactions, and acquaintances (Weiβ, 2002).
2. SODA ("Societies in Open and Distributed Agent spaces")	Agent and multi-agent technology	This is another good example of an analysis and design method that concentrates on the social (inter-agent) aspects of agent systems and that employs the concept of coordination models (Weiβ, 2002).

3. Cassiopeia	Agent and multi-agent technology	This is a design method that distinguishes three levels of behaviour - elementary, relational, and organizational - and aims at capturing both structural and dynamic aspects of the target system (Weiβ, 2002).
4. Aalaadin	Agent and multi-agent technology	This is a general analysis and design framework that has its focus on the organizational level of multi-agent systems and is built on the three core concepts of agents, groups, and roles (Weiβ, 2002).
5. KGR	Object-oriented technology	This is a design and specification method for a particular class of agents, namely, BDI agents (Weiβ, 2002).
6. MaSE ("Multiagent Systems Engineering")	Object-oriented technology	This method covers design and initial implementation through two languages called AgML ("Agent Modeling Language") and AgDL ("Agent Definition Language") and builds upon OMT and UML (Weiβ, 2002).
7. MASSIVE ("MultiAgent SystemS Iterative View Engineering")	Object-oriented technology	This method covers analysis, design and code generation, and combines standard software engineering techniques such as multi-view modelling, round-trip engineering, and iterative enhancement (Weiβ, 2002).
8. AOAD ("Agent-Oriented Analysis and Design")	Object-oriented technology	This analysis and design method proposes the use of extended class responsibility cards (CRCs) and the use of both the Object Modelling Technique (OMT) and the Responsibility Driven Design (RDD) method known from object-oriented development (Weiβ, 2002).
9. MASB ("Multi-Agent Scenario-Based").	Object-oriented technology	MASB is an analysis and design method that covers issues of both objects and agents via behaviour diagrams, data models, transition diagrams, and object life cycles (Weiβ, 2002).
10. CoMoMAS ("Conceptual Modelling of Multi-Agent Systems")	Knowledge engineering technology	This is an elaborated extension of the CommonKADS methodology, supporting analysis, design, and automated code generation (Weiβ, 2002).
11. MAS-CommonKADS ("Multi-Agent System CommonKADS")	Knowledge engineering technology	This is another extension of CommonKADS that supports analysis and design of agent-oriented systems (Weiβ, 2002).

Table 14: Categorization of AOSE methologies according to their focusing disciplinary background

Other agent-oriented software engineering methologies (AOSE) are Tropos, Agent-Oriented Analysis and Design, Agent Modelling Technique for Systems of BDI agents, Agent Oriented Methodology for Enterprise Modelling, PASSI (a Process for Agent Societies Specification and Implementation), Prometheus , AOR, ROADMAP, OPM /MAS, Ingenias, DESIRE, AAII methodology, Cooperative Information Agents design, Adept, AUML, ADELFE, MESSAGE /UML, The Styx Agent Methology, SABPO, EXPAND ("Expectation-oriented analysis and design") and ODAC (URL8; Iglesias *et al.,* 1999*;* Cernuzzi *et al.,* 2004; Wooldridge & Ciancarini, 2000;URL9; Weiβ, 2002).

4. Agents in Virtual Enterprises (VE)

4.1 What is a Virtual Enterprise?

The term, and the concept, "Virtual Enterprise" emerged already in the beginning of nineties and could be seen as the further optimization and perfection of the basic ideas about dynamic networking (Putnik, 2004). Although the virtual enterprise research represents a growing and multidisciplinary area it still lacks a precise definition of the concepts and an agreement on the used terminology (Camarinha-Matos, 2002). So far, there is no unified definition for this paradigm and a number of terms are even competing in the literature while referring to different aspects and scopes of virtual enterprises (Camarinha-Matos, 2002). Akin concepts are supported by Gijsen *et al.* (2002), Freitas Mundim *et al.* (2000), Putnik (2004), and Petersen *et al.* (2001).

The definitions range from the virtual enterprise as a simple subcontracting network to the virtual enterprise as a dynamic network, in which the partners share that share resources, risks and even markets, and which operates in a virtual environment or with virtual agents (Putnik, 2004). According to Do *et al.* (2000) a virtual enterprise is a form of cooperation of independent market players (enterprises, freelancers, authorities etc.) which combine their core competencies in order to manufacture a product or to provide a service. Marík & McFarlane (2005) conclude that a virtual enterprise represents a cluster of organizations collaborating to achieve one or more goals. Katzy & Schuh (1999) define that the virtual enterprise is based on the ability to create temporary co-operations and to realize the value of a short business opportunity that the partners cannot (or can, but only to lesser extent) capture on their own. Other attempts at defining virtual enterprises are listed in (Petersen *et al.*, 2001).

In our opinion, an interesting definition that we adopt is the following: '*a goal-oriented constellation of (semi)autonomous distributed entities. Each entity, which can be an organization and/or an individual, attempts to maximize its own profits as well as contribute to defining and achieving the overall goals of the virtual enterprise. Virtual enterprises are not rigid organizational structures within rigid frameworks, but rather (heterogeneous) ensembles, continuously evolving over time*' (p. 2) (Petersen *et al.*, 2001).

4.2 Why to use Agents in Virtual Enterprises?

Marík & McFarlane (2005) state that a virtual enterprise might address problems ranging from simple membership to distributed inventory management and synchronization of supply, production, and distribution schedules. They also support that these problems are inherently distributed, with each organization willing to share only limited information and having its own business goals in conjunction with the overall goal. All the above statements orientate to an agent technology solution.

According to Jennings *et al.* (1998), considering a virtual enterprise, the domain involves an inherent distribution of data, problem solving capabilities, and responsibilities. In addition, the integrity of the existing organizational structure and the autonomy of its sub-parts needs to be maintained (Jennings *et al.*, 1998). Moreover, interactions are fairly sophisticated, including negotiation, information sharing, and coordination (Jennings *et al.*, 1998). Besides, the problem solution cannot be entirely prescribed (Jennings *et al.*, 1998). According to Jennings *et al.* (1998), all the above observations motivate the choice of agents as a technology solution as well.

According to Fox & Gruninger (1998), the entrepreneurial and virtual nature of the agile enterprise coupled with the need for people and information to have a strategic impact entails a greater degree of communication, coordination and cooperation within and among enterprises. In other words, the agile organisation must be integrated (meaning by the term integrated the structural, behavioural and information integration of the enterprise) (Fox & Gruninger, 1998). Petersen *et al.* (2001) support that cooperation is required both to perform work and to adapt the constellation to the varying needs of the environment. They state that goal-oriented and distributed nature of virtual enterprises implies that there is no central control; rather, the control is decentralized. According to their opinion, the distributed and goal-oriented nature of the virtual enterprise provides a strong motivation for the use of agents to model virtual enterprises.

The following parallelism demonstrates remarkable interest. According to Rahwan *et al.* (2001) the virtual enterprise creation could be viewed as a Cooperative System design problem. A Cooperative System is a system in which a set of autonomous agents (computational and human) interact with each other through sharing their information, decision making capabilities and other resources, and distributing the corresponding workload among themselves, in order to achieve common and complementary goals (Camarinha-Matos & Afsarmanesh, 1998). The above parallelism motivates as well the agents as a technology solution.

The nature of agents, by definition, enables decentralized control of the enterprise, which is desirable in a dynamic and flexible environment, and the behaviour of the complete enterprise emerges as a result of the behaviours of the individual agents (Petersen *et al.*, 2001).

Another strong point in favour of the adoption of agents is their versatility (Petersen *et al.*, 2001). They can play two main roles (Petersen *et al.*, 2001). First, they provide a flexible means of modelling the virtual enterprise in terms of cooperative work among the agents (Petersen *et al.*, 2001). Second, they can be used to provide active support to the members of the virtual enterprise (Petersen *et al.*, 2001). Thus, agents being computational entities, the resulting model provides an easy and efficient passage to the computational support that is required by virtual enterprises (Petersen *et al.*, 2001).

According to Marík & McFarlane (2005), MASs and relevant technologies consider each company as an agent able to carry out specific (usually quite complex) functions. The agents are registered with a certain platform and communicate in a standard agent communication language (Marík & McFarlane, 2005). Virtual enterprise formation, as well as the joint planning and scheduling activities, is based on jointly known negotiation rules and scenarios (Marík & McFarlane, 2005). These are very similar (or identical) to protocols or auctions in the MAS domain (Marík & McFarlane, 2005). The highly specialized members of a virtual enterprise, such as brokers or professional network organizers, can easily find their counterparts in the MAS community—for example, various middle agents and brokers (Marík & McFarlane, 2005). The negotiation and brokering algorithms that have proven useful for the MAS domain can serve to formalize (and later automate) the corresponding virtual enterprise processes (Marík & McFarlane, 2005). Specialized agents called meta-agents could also serve as tools both to help detect the network's less efficient parts or bottlenecks and to provide advice supporting the virtual enterprise's self-evolution in the desired direction (Marík & McFarlane, 2005). Virtual enterprise creation is analogous to coalition formation in the MAS domain (Marík & McFarlane, 2005).

They also support that MAS concept of knowledge sharing, which classifies knowledge as public, private, and semiprivate, has high potential for virtual enterprises. Requirements for keeping agents' knowledge confidential and preventing knowledge disclosure, as well as specific security principles used with MASs, can be reused for virtual enterprises (Marík & McFarlane, 2005).

4.3 Current research activity focusing on Agents in Virtual Enterprises (VE)

Virtual enterprises have recently received increasing attention. Due to the advancement of distributed information technology and the changing needs of the business community, enterprises are expected to be more agile and responsive (Petersen *et al.*, 2001). Many current developments in multi-agent systems (MAS) are more and more focused on the production of robust development environments (Camarinha-Matos, 2002). Considerable efforts are also being put on standardization of architectures and communication languages, which are important requirements for the industrial application of the paradigm (Camarinha-Matos, 2002). We have observed that there is a remarkable body of literature that studies the application of agent technology to virtual enterprises as researchers pay enough attention to this scientific area of activity. In continuance of the study, some prominent research efforts follow.

According to Yonghe & Biqing (1999), decision and control processes within the domain of virtual enterprises have not received deserved attention till now. Based on agent technology, they bought up an architecture for control and decision-making during the dynamic creation and operation of a virtual enterprise. An approach for integrating different business units is presented (Yonghe & Biqing, 1999). A prototype software simulating the design of a new product in a virtual enterprise is developed (Yonghe & Biqing, 1999).

Petersen *et al.* (2003) present the virtual enterprise formation process as an agent interaction protocol and an approach to its implementation. They have focussed on the selection of partners within the formation process in order to understand these interactions and the contents of the messages that are exchanged between the agents. Based on this, they describe how the AGORA multi-agent architecture can be used to support the formation of a virtual enterprise.

Gou *et al.* (2001) propose an agent-based virtual enterprise model and provide the agent collaboration mechanisms under the model, thereby achieving the agent based virtual enterprise modeling and operation control. Their agent-based approach achieves distributed control over the whole business process execution of the virtual enterprise.

According to Fankhauser & Tesch (1999), negotiations encourage agents to reason about the interests of their opponents. Thus, negotiations suffer from counter speculations (Fankhauser & Tesch, 1999). Auctions apply to asymmetric trading only; they either favor the auctioneer or the bidders (Fankhauser & Tesch, 1999). Both mechanisms do not promote agents to tell the truth (Fankhauser & Tesch, 1999). Therefore, they propose to use a trustbroker to mediate between the agents. They introduce three symmetric, negotiation free one-step protocols to carry out a sequence of decisions for agents with possibly conflicting interests. The protocols achieve substantially better overall benefit than random or hostile selection, and they avoid lies (Fankhauser & Tesch, 1999). They analyze the protocols with respect to informed vs. uninformed lies, and with respect to beneficial vs. malevolent lies, and show that agents are best off to know and announce their true interests.

Gong's & Wang's (2000) research is a contribution to the model of multi-agent system (MAS) for supporting the dynamic enterprise model (DEM). It separates the business process from the organizational structure (organizational structure tier and business process tier), models each of them as MAS, and coordinates agents by the 'yellow page" mechanism (Gong's & Wang's, 2000). This model not only can regulate itself in terms of DEM, but also centered on the coordination strategies between agents composing it (Gong's & Wang's, 2000). It is believed that the model of MAS is a practical way to build flexible enterprise information system (Gong's & Wang's, 2000).

Chrysanthis *et al.* (1999) view the establishment of a virtual enterprise as a problem of dynamically expanding and integrating workflows in decentralized, autonomous and interacting workflow management systems. They focus on the idea of mobile agents called adlets and their use in establishing virtual enterprises that involves advertising, negotiating and exchanging control information and data as well as its management.

Szirbik *et al.* (2000) propose a systematisation of the monitoring and control aspects in a virtual enterprise. As an instrument, they use the mobile agent paradigm, defining the concept of a mobile agent web (MA-web). According to them, one of the roles of the agents in this environment is to mediate negotiations between the parties of the virtual enterprise. They make some assumptions about the new behaviour and code of conduct in the MA-web, such as the willingness to share data and knowledge.

Based on the analysis of why agent-based mechanism is suitable and only suitable for cross-domain cooperation of virtual enterprise, Zhang et al. (2004) propose a framework to implement it. In their framework, there is a service information supply-demand center which is in charge of service information management and agent is responsible for cooperative partner selecting before cooperation and interaction during cooperation. The relevant key strategies and basic interaction models are also described (Zhang et al., 2004).

Ouzounis & Tschammer (2001) discuss concepts and technologies that are considered to satisfy key requirements of dynamic virtual enterprises, and propose DIVE, a framework for the specification, execution and management of shared business processes in dynamic virtual enterprises.

Suh *et al.* (2005) describe an open .and flexible infrastructure to support dynamic collaboration among companies through the entire lifecycle of the virtual enterprise. The proposed approach is an agent-enhanced architecture on which the conversation model is grafted (Suh *et al*., 2005). The collaboration among enterprises is modeled by a collaboration policy which is a machine-readable specification of a pattern of message exchange among agents participating in the collaboration (Suh *et al*., 2005).

5. Future trends

Luck *et al.* (2006) stated a thorough and outstanding approach about the future of multi-agent systems. As we consider their point of view extremely prominent, we appose at this point of the chapter some parts of their findings (for a more complete investigation consult. Luck *et al.* (2006) extrapolated future trends in multi-agent systems by classifying them into four broad phases (current, short-term future, medium-term future and long-term future) of development of multi-agent system technology over the next decade.

At first phase, multi-agent systems are typically designed by one design team for one corporate environment, with participating agents sharing common high level goals in a single domain (Luck *et al.*, 2006). These systems may be characterised as closed (Luck *et al.*, 2006). The communication languages and interaction protocols are typically in-house protocols, defined by the design team prior to any agent interactions (Luck *et *al.*, 2006). Design approaches, as well as development platforms, tend to be ad hoc, inspired by the agent paradigm (Luck *et al.*, 2006). There is also an increased focus on taking methodologies out of the laboratory and into development environments, with commercial work being done on establishing industrial-strength development techniques and notations (Luck *et al.*, 2006).

In the short-term future, multi-agent systems will increasingly be designed to cross corporate boundaries, so that the participating agents have fewer goals in common, although their interactions will still concern a common domain, and the agents will be designed by the same team, and will share common domain knowledge (Luck *et al.*, 2006). Standard agent communication languages will be used, but interaction protocols will be mixed between standard and non-standard ones (Luck *et al.*, 2006). Development methodologies, languages and tools will have reached a degree of maturity, and systems will be designed on top of standard infrastructures such as web services or Grid services, for example (Luck *et al.*, 2006).

In the medium term future, multi-agent systems will permit participation by heterogeneous agents, designed by different designers or teams (Luck *et al.*, 2006). Any agent will be able to participate in these systems, provided their (observable) behaviour conforms to publicly-stated requirements and standards (Luck *et al.*, 2006). However, these open systems will typically be specific to particular application domains (Luck *et al.*, 2006). The languages and protocols used in these systems will be agreed and standardised (Luck *et al.*, 2006).

In the long-term future, we will see the development of open multi-agent systems spanning multiple application domains, and involving heterogeneous participants developed by diverse design teams (Luck *et al.*, 2006). Agents seeking to participate in these systems will be able to learn the appropriate behaviour for participation in the course of interacting, rather than having to prove adherence before entry (Luck *et al.*, 2006). Selection of communications protocols and mechanisms, and of participant strategies, will be undertaken automatically, without human intervention (Luck *et al.*, 2006).

The above mentioned aspect about future is enhanced with the AOSE Technical Forum Group's (2004) perception of the future trends in the area of agent-oriented software engineering. According to AOSE Technical Forum Group (2004), the research in the area of agent-oriented software engineering is still in its early stages, and several challenges need to be faced before agent-oriented software engineering becoming a widely accepted and a practically usable paradigm for the development of complex software systems. One possible way to identify and frame the key research challenges in the area of agent-oriented software engineering is to recognize that such challenges may be very different depending on the "scale of observation" adopted to model and build a software system (AOSE Technical Forum Group, 2004).

At one extreme, the micro scale of observation is that where the system to be engineered has to rely on the controllable and predictable behaviour of (a typically limited number of) individual agents, as well as on their mutual interactions (AOSE Technical Forum Group, 2004). There, the key engineering challenges are related to extending traditional software engineering approaches toward agent-oriented abstractions (AOSE Technical Forum Group, 2004). Brand new modelling and notational tools, as well as possibly brand new software process models may be needed (AOSE Technical Forum Group, 2004).

At the other extreme, the macro scale of observation is the one where a multi-agent system is conceived as a multitude of interacting agents, for which the overall behaviour of the system, rather than the mere behaviour of individuals, is the key of interest (AOSE Technical Forum Group, 2004). In this case, a discipline of agent-oriented software engineering should focus on totally different problems, and should be able to develop novel "systemic" approaches to software engineering, possibly getting inspiration from areas such as complex systems sciences and systemic biology (AOSE Technical Forum Group, 2004).

In between, the meso scale of observation is that where the need of predictability and control typical of the micro scale clashes with the emergence of phenomena typical of the macro scale (AOSE Technical Forum Group, 2004). Therefore, any engineering approach at the meso scale requires accounting for problems that are typical of both the micro and the macro scale, and possibly for new problems specific to the meso scale (AOSE Technical Forum Group, 2004). These include: identifying the boundaries of a systems – which may be challenging in the case of open multi-agent systems; electing trust as a primary design issue; identifying suitable infrastructures for multi-agent systems support (AOSE Technical Forum Group, 2004).

As concerns the virtual enterprises' scientific domain, we believe that agent technology has much to offer with respect to the formation and the operation of a virtual enterprise. According to Camarinha-Matos (2002), several challenges remain open for MAS requiring further research, such as support for the full life cycle of the virtual enterprise, adoption of contract-based coordination models, necessary integration of MAS with several other paradigms, interoperation with legacy systems and enterprise applications, inclusion of specialized protocols and standards, and support of robust safety mechanisms.

There is a need to integrate ACL with mechanisms for safe communications (cryptography, digital signature, certification, etc.) that have been developed for virtual enterprises and e-commerce (Camarinha-Matos, 2002). The development of advanced simulation tools to support planning, optimization, and assessment of operation of virtual enterprises and distributed business processes is another open challenge that can benefit from a MAS approach (Camarinha-Matos, 2002). Finally it is important to stress that in order to be accepted by the industrial community, MAS applications need to be successfully demonstrated in complex real world pilot systems (Camarinha-Matos, 2002).

5. Conclusion

The area of software agents is vibrant and rapidly developing. A number of fundamental advances have been made in the design and the implementation of software agents as well as in the interaction between software agents. In this brief chapter, we have tried to convey some of the key concepts of the active field of software agents and make a reference point to a large body of literature outlining essential issues. We were limited to enumerate our findings of our survey regarding software agent technology, instead of judge them, aiming to provide a synoptic review of the basic aspects. It is up to the reader to judge how successful we have been in meeting our goal in this chapter. In addition, we have argued the issue of applying the agent technology to virtual enterprise. Our purpose was to offer a brief introduction of the application of agent technology to virtual enterprises and to provide some useful hints for further studying concerning the above mentioned theme.

References

Nwana, H., & Ndumu, D. (1996). A Brief Introduction to Software Agent Technology. Unicom Seminar on “Real-World Applications of Intelligent Agent Technology”, London UK, 6/11-13/96, 278-292.

Wooldridge, M., & Jennings, N. R. (1995). Intelligent agents: theory and practice. The Knowledge Engineering Review 10(2), 115-152.

Oliveira, E., Fischer, K., & Stepankova, O. (1999). Multi-agent systems: which research for which applications. Robotics and Autonomous Systems, 27, 91-106.

Nwana, H. S. (1996). Software Agents: An Overview. Knowledge Engineering Review, 11(3), 1-40.

Sycara, K. P. (1998). Multi-agent Systems. AI magazine, Intelligent Agents Summer, 19(2), 79-92.

Jennings, N. R., Sycara, K., & Wooldridge, M. (1998). A Roadmap of Agent Research and development. Autonomous Agents and Multi-Agent Systems Journal (Eds.), Boston, Kluwer Academic Publishers, 1(1), 7-38.

Wooldridge, M. (1998). Agent-based computing. Interoperable Communication Networks, 1(1), 71-97.

Weiß, G. (2002). Agent orientation in software engineering. Knowledge Engineering Review, 16(4), 349–373.

Jennings, N. R., Norman, T. J., & Faratin, P. (1998). ADEPT: An Agent-based Approach to Business Process Management. ACM SIGMOD Record, 27(4), 32-39.

Camarinha-Matos, L. M. (2002). Multi-agent systems in virtual enterprises. Proceedings of AIS’2002 – International Conference on AI, Simulation and Planning in High Autonomy Systems, SCS publication, Lisbon, Portugal, 27-36.

Petersen, S. A.., Divitini, M., & Matskin, M. (2001). An agent-based approach to modeling virtual enterprises. Production Planning & Control, 12(3), 224-233.

Camarinha-Matos, L. M., & Afsarmanesh, H. (1998). Cooperative Systems Challenges in Virtual Enterprises. Proceedings of CESA’98 - IMCAS Multiconference on Computational Engineering in Systems Applications, Nabeu - Hammamet, Tunisia.

Jennings, N., & Wooldridge, M. (1996). Software Agents. IEE Review, 17-20.

Bradshow, J. M. (1997). An Introduction to Software Agents. Software Agents, J. M. Bradshow, Cambridge, MIT Press.

Hayes, C. C. (1999). Agents in a Nutshell-A Very Brief Introduction. IEEE Transactions on Knowledge and Data Engineering, 11(1), 127-132.

Green, S., Hurst, L., Nangle, B., Cunningham, D. P., Somers, F., & Evans, D. R. (1997). Software agents: a review. Technical report, TCS-CS-1997-06, Trinity College Dublin, Broadcom Éireann Research Ltd..

Zambonelli, F., Jennings, N., & Wooldridge, M. (2003). Developing Multiagent Systems: the Gaia Methology. ACM Transactions on Software Engineering and Methodology, 12(3).

Chira, C. (2003). Software Agents, IDIMS Report, 2/21/03, http://pan.nuigalway.ie/code/docs/agents.pdf
Alonso, E. (2002). AI and agents: State of the art. AI Magazine, 23(3), 25-29.
Sycara, K. P. (1998). The many faces of agents. AI Magazine 19(2), 11-12
Odell, J. (2000). Agents: Technology and usage (Part 1). Executive Report 3(4).
Etzioni, O., & Weld, D. S. (1995). Intelligent agents on the Internet: Fact, Fiction and Forecast. IEEE Expert, 10(4), 44-49.
Horn, E., Kupries, M., & Reinke, T. (1999). Properties and Models of Software Agents and Prefabrication for Agent Application Systems. (32) Hawaii: International Conference on System Sciences (HICSS-32), Software Technology Track.
Goodwin, R. (1993). Formalizing properties of agents. Technical report, School of Computer Science, Carnegie-Mellon University, Pittsburgh.
Franklin, S., & Graesser, A. (1996). Is it an Agent, or just a Program? : A Taxonomy for Autonomous Agents. Proceedings of the Third International Workshop on Agent Theories, Architectures and Languages.
Wooldridge, M. (1999). Intelligent Agents, The MIT Press.
Kostakos, V., & Taraschi, C. (2001). Agents. http://www.cs.bath.ac.uk/~vk/files/agents.pdf
Bussink, D. (2004). A Comparison of Language Evolution and Communication Protocols in Multi-agent Systems. 1st Twente Student Conference on IT, Track C - Intelligent_Interaction, http://referaat.ewi.utwente.nl/
OMG Agent Working Group (2000), Agent Technology. Green paper, OMG Document ec/8/1/00, (1).
URL1: CORBA® BASICS, http://www.omg.org/gettingstarted/corbafaq.htm
URL2: www.omg.org
URL3: The CORBA Object Group Service. A Service Approach to Object Groups in CORBA, N° 1867 (1998), http://lsrwww.epfl.ch/OGS/thesis/
URL4: http://www.creativescience.com/Products/soa.shtml
Bracho, A., Matteo, A., & Metzner, Ch. (1999). A taxonomy fro comparing distributed objects technologies. CLEI Electronic Journal 2 (2).
URL5: http://java.sun.com/products/jdk/rmi/
URL6: DCOM Technical Overview, Microsoft Corporation (1996), http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dndcom/html/msdn_dcomtec.asp
URL7: Tuukka Vartiainen, Java Beans and Enterprise Java Beans, http://www.cs.helsinki.fi/u/campa/teaching/tukka-final.pdf
Nwana, H., & Wooldridge, M. (1996). Software Agent Technologies. BT Technology Journal 14(4), 68-78.
Serenko & Deltor, B. (2002). Agent Toolkits: A General Overview of the Market and an Assessment of Instructor Satisfaction with Utilizing in the Classroom. Working Paper #455, http://www.business.mcmaster.ca/msis/profs/detlorb/nserc/McMaster_Working_Paper_455.pdf
URL8: Agent-Oriented Software Engineering. "Research Area Examination" (2005), http://www.deg.byu.edu/proposals/ResearchAreaExamMuhammedJM.pdf
Iglesias, C.A., Garijo, M., & Gonzalez, J.C. (1999). A Survey of Agent-Oriented Methologies. Intelligent Agents V, Proceedings of the Fifth International Workshop on Agent Theories, Architectures, and Languages (ATAL-98), Lecture Notes in Artificial Intelligence. Springer-Verlag, Heidelberg, 1555, 317-330.
Cernuzzi, L., Cossentino, M., & Zambonelli, F. (2004). Process Models for Agent-Based Development. International Journal on Engineering Applications of Artificial Intelligence (EAAI), Elsevier, http://www.pa.icar.cnr.it/~cossentino/paper/eaai_zambonelli_draft.pdf
Wooldridge, M., & Ciancarini, P. (2000). Agent-Oriented Software Engineering: The State of the Art. First Int. Workshop on Agent-Oriented Software Engineering, P. Ciancarini and M. Wooldridge (ed.), Springer-Verlag, Berlin, 1957, 1-28.
URL9: AOSE Methologies, http://www.science.unitn.it/~recla/aose/
Putnik, G. D. (2004). Virtual Enterprise as a "flow" enterprise. Fourth Annual Meeting of the European Chaos and Complexity in Organisations Network ECCON - ECCON 2005, Driebergen, Netherlands.
Gijsen, J. W. J., Szirbik, N. B., & Wagner, G. (2002). Agent Technologies for Virtual Enterprises in the One-of-a-kind-Production Industry. International Journal of Electronic Commerce, 7(1), 9-26.
Freitas Mundim, A. P. Rossi, A.., & Stocchetti, A.. (2000). SME in Global Market: Challenges, Opportunities and Threats. Brasilian Electronic Journal of Economics, 3(1).
Do, V., Halatchev, M., & Neumann, D. (2000). A contextbased approach to support virtual enterprises. Proceedings of the 33rd Hawaii International Conference on System Sciences
Marík, V., & McFarlane, D. C. (2005). Industrial Adoption of Agent-Based Technologies. IEEE Intelligent Systems 20(1), 27-35
B. R. Katz, B. R., & Schuh, G. (1999). The Virtual Enterprise. Book, http://portal.cetim.org/file/1/68/KatzySchuh-1999-The_virtual_enterprise.pdf
Fox, M.S., & Gruninger, M., (1998). Enterprise Modelling. AI Magazine, AAAI Press, 109-121.

Rahwan, I., Kowalczyk, R., & Yang, Y. (2001). Virtual Enterprise Design - BDI Agents vs. Objects. Advances in Artificial Intelligence, Lecture Notes in Artificial Intelligence, 2112, 147-157

Yonghe, L., & Biqing, H. (1999). Virtual enterprise: an agent-based approach for decision and control. Systems, Man, and Cybernetics, IEEE SMC '99 Conference Proceedings, 1999 International Conference, 6, 451-456.

Petersen, S.A., Jinghai Rao, & Matskin, M. (2003). Virtual enterprise formation with agents - an approach to implementation. Intelligent Agent Technology. IAT 2003, IEEE/VVIC International Conference, 527-530.

Gou, H., Huang, B., Liu, W., & Li, Y. (2001). Agent-based virtual enterprise modeling and operation control. Systems, Man, and Cybernetics, 2001 IEEE International Conference, 3, 2058-2063.

Fankhauser, P., &Tesch, T. (1999). Agents, a broker, and lies. Research Issues on Data Engineering: Information Technology for Virtual Enterprises, RIDE-VE '99, Proceedings, Ninth International Workshop, 56-63

Gong, B., & Wang, S. (2000). Model of MAS: supporting dynamic enterprise model. Intelligent Control and Automation, Proceedings of the 3rd World Congress, 3, 2042-2046.

Chrysanthis, P. K., Znati, T., Banerjee, S., & Shi-Kuo Chang (1999). Establishing virtual enterprises by means of mobile agents. Research Issues on Data Engineering: Information Technology for Virtual Enterprises, RIDE-VE '99, Proceedings, Ninth International Workshop, 116-123.

Szirbik, N., Aerts, A.., Wortmann, H., Hammer, D., & Goossenaerts, J. (2000). Mediating negotiations in a virtual enterprise via mobile agents. Research Challenges, Proceedings, Academia/Industry Working Conference, 237-242.

Yan Zhang, Meilin Shi, & Shaohua Zhang (2004). An agent-based framework for cross-domain cooperation of virtual enterprise. Computer Supported Cooperative Work in Design, Proceedings, The 8th International Conference, 1, 291-296.

Ouzounis, V. K., & Tschammer, V. (2001). An agent-based life cycle management for dynamic virtual enterprises. Computer Supported Cooperative Work in Design, The Sixth International Conference, 451-459.

Luck, M., McBurney, P., & Gonzalez-Palacios, J. (2006). Agent-Based Computing and Programming of Agent Systems. Lecture Notes in Artificial Intelligence, Agent-Based Computing and Programming of Agent Systems, 3862, 23-37, Springer.

AOSE Technical Forum Group (2004). AL3-TF1 Report, 30 June- 2 July, Rome, http://www.pa.icar.cnr.it/~cossentino/al3tf1/docs/aose_tfg1_report.pdf