GMIT
GALWAY-MAYO INSTITUTE OF TECHNOLOGY
INSTITIUID TEICNEOLAÍOCHTA NA GAILLIMHE-HAIGH ED

The Development of a Multi-Agent Design Information Management and Support System

In One Volume

Camelia Chira B.Sc. M.Sc.

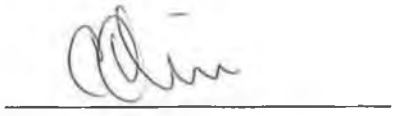
September 2005

Submitted for the degree of
Doctor of Philosophy

Submitted to	:	Galway-Mayo Institute of Technology, Ireland
Research Supervisor	:	Dr. Thomas Roche

Declaration

I hereby declare that the work presented in this thesis is my own and that it has not been used to obtain a degree in this university or elsewhere.



Camelia Chira

Dedication

To Erin Carla….

Prologue

The research presented in this thesis was developed as part of a project called Intelligent Agent Based Collaborative Design Information Management and Support Tools (I-DIMS). The I-DIMS project was funded by the Irish Research Council for Science, Engineering and Technology (IRCSET) as a partnership project between Galway-Mayo Institute of Technology and the Computer Integrated Manufacturing Research Unit (CIMRU), National University of Ireland, Galway. The project aimed to investigate the use of software agents to support the synthesis and presentation of information for distributed teams for the purposes of enhancing design, learning, creativity, communication and productivity.

Acknowledgments

Many people contributed in different ways to the development of this thesis.

I would like to thank Tom Roche, my project supervisor, for all the help and assistance particularly during the process of thesis write up. Thank you for offering me the opportunity to experience the world of research and academia during the last five years.

I would like to express my appreciation to all the members of the I-DIMS team for their contribution to the current research. David – for all the great discussions we had over the last three years, especially during the testing phase; all the best in your future career. Attracta – for your great ideas and fresh perspectives. Valerie – for all your support, good luck with your PhD.

I would also like to thank all my friends in Galway for just being here to make life easier – to Alex, Kati, Feri, Oana, Eli, Arpi, Anna, Dieter, Cosmina, Ionel, Gabi, Cosmin.

Special thanks to my husband, Ovi, for all the encouragement, for always having the patience to read chapters, listen and explain his views and re-read, re-listen and re-explain over and over. I am grateful for everything.

Many thanks to my parents and mother-in-law for all their help especially during the last year.

Finally, I have to thank my daughter, Carla, for being such a good and happy girl allowing me to concentrate on my work when I needed to.

Published Work Associated with this Thesis

Camelia Chira, David Tormey, Ovidiu Chira, Thomas Roche, Attracta Brennan, "A Multi-Agent Design Information Management and Support System", Advanced Engineering Informatics, 2005, Submitted.

Camelia Chira, Ovidiu Chira, Thomas Roche, "Multi-Agent Support for Distributed Engineering Design", IEA/AIE 2005, 18th International Conference on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems, Bari, Italy, June 22-25, 2005.

Ovidiu Chira, Camelia Chira, David Tormey, Attracta Brennan, Thomas Roche, "An agent-based approach to knowledge management in distributed design", Special Issue of Journal of Intelligent Manufacturing, 2005.

Camelia Chira, David Tormey, Thomas Roche, "An Ontological and Agent Based approach to Knowledge Management within a Distributed Design Environment", First International Conference on Design Computing and Cognition (DCC'04), MIT, Cambridge, USA, July 19-21, 2004.

Camelia Chira, Ovidiu Chira, Thomas Roche, Attracta Brennan, "Semantic Tools for Knowledge Management in Distributed Engineering Design", 10th International Conference on Concurrent Enterprising Escuela Superior de Ingenieros, Seville, Spain, June 14-16, 2004.

David Tormey, Camelia Chira, Thomas Roche, Jim Browne and Attracta Brennan, "Ontology oriented knowledge management tool to support DFX activities", International IMS Forum, Villa Erba - Cernobbio - Lake Como, Italy, May 2004.

Ovidiu Chira, Camelia Chira, David Tormey, Attracta Brennan, Thomas Roche, "A Multi-Agent Architecture for Distributed Design", International Conference on Applications of Holonic and Multi-Agent Systems HoloMAS 2003, Prague, September 1-3, 2003.

David Tormey, Camelia Chira, Ovidiu Chira, Thomas Roche, Attracta Brennan, "Development of Engineering Design Methodologies and Software Tools to Support the Creative Process of Design in a Distributed Environment", International Conference on Engineering Design ICED 03, Stockholm, August 19-21, 2003.

David Tormey, Ovidiu Chira, Camelia Chira, Attracta Brennan, Thomas Roche, "The Use of Ontologies for Defining Collaborative Design Processes", 32nd International Conference on Computers and Industrial Engineering, University of Limerick, August 11-12, 2003.

Ovidiu Chira, Camelia Chira, David Tormey, Attracta Brennan, Thomas Roche, "An agent-based approach to knowledge management in distributed design", 10th ISPE International Conference on Concurrent Engineering: Research and Applications, Madeira Island, Portugal, July 26-30, 2003.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1. Problem Statement

Emerging enterprise models involve multiple design teams with heterogeneous skills cooperating together in order to achieve global 'optima' in design. Moreover, both the human and the information resources involved in the design process are geographically, temporally, functionally and/or semantically distributed in a virtual environment. A crucial element of the success of distributed collaborative design is the cooperation process (i.e. communication, co-location, coordination and collaboration) among participants dispersed across the enterprise. If this cooperation process is well supported, collaborative design becomes highly beneficial to the successful location of the optimal design solution. The main benefits can be summarised as follows:

- Cooperating multidisciplinary design teams create a beneficial distributed cognition by sharing their skills and expertise.
- The generation of new insights, new ideas and new artefacts is supported and enhanced.
- Design solutions are enriched by the multiple skills of the designers engaged in the design task and by easier access to multiple sources of information.

However, the availability of the software infrastructure to support the cooperation process among distributed participants to the design process and to facilitate the communication of information in a virtual environment remains the key success factor of distributed engineering design. There is a need to study the cooperation process in a distributed design environment and to build software applications (particularly to support knowledge management activities in the enterprise) that facilitate and, if possible, enhance the creative process of design.

1.2. Research Objectives

This thesis aims to design, implement, test and validate an intelligent system for distributed and collaborative engineering design. The proposed system is called Multi-Agent Design Information Management and Support System (MADIS) and aims to efficiently facilitate the management of the data-information-knowledge value chain in a distributed design environment. MADIS employs multi-agent systems to enable interoperation among distributed resources and information ontologies for knowledge sharing, reuse and integration. Consisting of a collection of autonomous software agents able to cooperate with each other, the proposed system supports the designer's decision-making process in a distributed environment by

facilitating the storage, retrieval, exchange and presentation of data, information and knowledge. Furthermore, cooperation among multidisciplinary design teams is aided by flexible graphical user interfaces, a common shared knowledge base and easy access to relevant and timely information. The goals of the proposed multi-agent architectural framework can be summarised as follows:

- Minimise the effect of resources dispersion (particularly temporal and geographical dispersion) and the misunderstandings that might be generated by the functional and semantic distribution.
- Minimise the time spent for searching and retrieval of information, the effort of information translation between different tools and the administrational and organisational efforts not directly related to the design process (e.g. revision control).
- Maximise the quality of information, knowledge sharing and reuse within the extended enterprise.
- Maximise the flexibility of the user interfaces, designer's learning curve and creativity.

By addressing all of these factors, the multi-agent system presented in this thesis aims to ultimately optimise design process operation and management in the virtual enterprise.

In order to deliver the proposed multi-agent design information management architecture and system, the following objectives have been set for the current research and development work:

- *Objective 1*

  Research distributed engineering design in terms of definition, characteristics, potential benefits and problematic aspects.
- *Objective 2*

  Investigate the current approaches to support the process of design in a distributed environment.
- *Objective 3*

  Review state-of-the-art AI technologies including software agents, agent-based systems, multi-agent systems, ontologies and semantic web.
- *Objective 4*

  Specify and analyse the requirements of a computational system intended to support distributed engineering design.
- *Objective 5*

Design the architectural framework of the proposed MADIS system (e.g. architectural components, role, structure, properties, interrelationships, enabling technologies).

- *Objective 6*

  Develop a MADIS prototype by implementing the main components of the proposed system (proof-of-the-concept model).

- *Objective 7*

  Test and validate the proposed architecture and system.

The overall objective of the current thesis is to present the whole process of the MADIS software model construction showing how it supports the optimisation of the solution space of the collective dispersed design team, working within a distributed extended enterprise design environment.

1.3. Approach to Work

The research associated with this thesis started by defining distributed engineering design in a virtual collaborative environment. Problems associated with the domain were identified and analysed in order to generate a set of initial requirements for a software system intended to support distributed design teams during interoperation in the virtual enterprise. Moreover, related work in the area of developing (intelligent) software applications for distributed design has been examined and many deficiencies have been identified. This process significantly aided the identification of the necessary requirements for MADIS. Furthermore, software agents and multi-agent system have been extensively reviewed in order to inform the design and implementation phases of MADIS. Based on current results from the agent research community and informed by the MADIS requirements analysis phase, a detailed specification of the proposed system's architecture has been created to include the structure and properties of the components that comprise the core multi-agent system and the interrelationships among all architectural components. This design phase of the proposed system was followed by the implementation of all comprising components (e.g. software agents, communication strategies, ontologies).

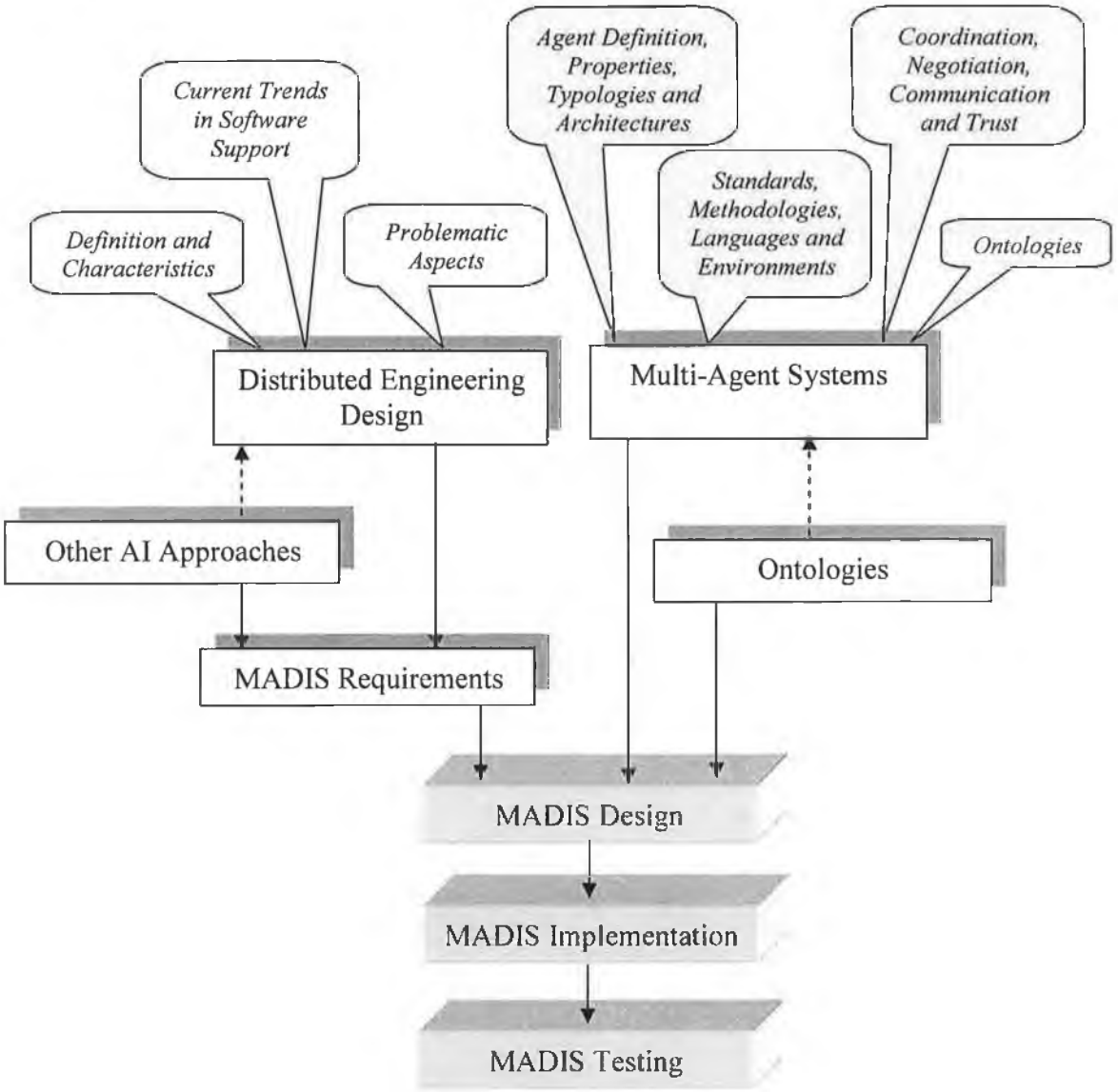


Figure 1.1. Approach to research

Conducted after the implementation process was completed (see Figure 1.1), the testing and validation phase of the proposed multi-agent system aims to demonstrate that agent properties such as autonomy, pro-activeness and cooperation address the problematic issues of distributed collaborative engineering design.

1.4. Thesis Structure

Chapter 1: Introduction

This chapter briefly introduces the problem addressed by the current research and presents the objectives and structure of the thesis.

Chapter 2: Distributed Collaborative Engineering Design

Chapter two reviews the distributed engineering design domain and identifies the main problems associated with collaborative design in a virtual environment. Related work in the area of the design and implementation of a software infrastructure for distributed collaborative design is presented. The computer-based systems analysed are those primarily based on software agents, agent architectures, multi-agent systems and/or information ontologies. The main characteristics of the currently available software systems for distributed design are outlined and used to feed the specification process of the intelligent system proposed by this thesis. The chapter concludes with a set of requirements for an intelligent system to support distributed collaborative design.

Chapter 3: Multi-Agent Systems

Chapter three offers an extensive literature review on software agents and multi-agent systems as this is the technology used to build the proposed software infrastructure for distributed engineering design. After the definition of software agent is given and numerous agent typologies are presented, the chapter continues by addressing the issues of designing and creating agent-based systems. Agent architectures an current agent standards are presented and then multi-agent systems are defined and detailed. Research in the area of trust, coordination, negotiation and communication within a multi-agent system is presented and analysed. Next, agent-oriented methodologies and agent languages and environments are examined. Chapter three concludes this comprehensive literature review by presenting some of the most successful applications of agents and multi-agent systems.

Chapter 4: Multi-Agent Design Information Management and Support

This chapter starts by presenting a detailed description of the multi-agent architecture which stays at the core of the proposed MADIS software infrastructure for distributed design. The design of this architecture started with the requirement analysis process and developed gradually into a detailed specification of the agents, agent subsystems and the relationships among them. The second part of this chapter describes the implementation phase of the multi-agent system offering the concrete solution for each aspect of the proposed architecture. The chapter also presents the engineering design ontology developed as part of the current

research. The proposed ontology is intended to capture the concepts and relations of the engineering design domain and therefore to facilitate knowledge sharing and reuse among all participants to the distributed design environment.

Chapter 5: MADIS Evaluation

Chapter five presents the MADIS testing and validation phase. Based on the Protocol Analysis method, MADIS was tested in a virtual environment by various designers asked to complete a set of predetermined tasks. The data analysis phase is supported by the think-aloud, communication and mouse tracking protocols registered during the test. The main results of the testing phase conclude this chapter.

Chapter 6: Conclusions and Future Work

This chapter contains the conclusions of the thesis and presents recommendations for further development of the multi-agent system for distributed engineering design.

Chapter 2

Distributed Collaborative Engineering Design

2.1. Introduction

Emerging as a response to market demands and competitive pressures, the distributed engineering design organization involves multiple design teams with heterogeneous skills cooperating together in order to achieve global 'optima' in design. Moreover, both the human and the information resources involved in the design process are geographically, temporally, functionally and/or semantically distributed in a virtual environment. Key aspects of this new organization of engineering design that need to be addressed include the support of the cooperation process among participants dispersed across the enterprise and the efficient management of the design related information structures circulated in the distributed design environment. Because the communication of information, the coordination of engineering design participants and team collaboration takes place in a computer based medium, the availability of the software infrastructure to support cooperation and facilitate the management of data, information and knowledge remains the key success factor of distributed design. After reviewing the distributed engineering design domain, this chapter aims to establish what are the necessary requirements of such a software infrastructure and to describe its high-level specification.

The structure of this chapter is as follows. After engineering design is defined, the distributed engineering design organization is introduced and described with an emphasis on its problematic aspects. Next, the current approaches to support distributed design are reviewed and the systems based on Artificial Intelligence techniques are detailed. Based on the main problems of distributed engineering design and the analysis of existing collaborative engineering design systems, a high-level specification for an intelligent computational system to support distributed design is generated at the end of the chapter.

2.2. Distributed Collaborative Engineering Design

A very broad area not entirely explored yet (Love 2002), design science "comprises a collection (a system) of logically connected knowledge in the area of design, and contains concepts of technical information and of design methodology" while also being concerned with "deriving from the applied knowledge of the natural sciences appropriate information in a form suitable for the designer's use" (Hubka and Eder 1987).

2.2.1. Engineering Design

Table 2.1 reflects the evolution of the design concept in time by presenting the various definitions of design resulted from research carried out by both academia and industry over the last forty years.

Autor(s)	Definition	Keywords	Reference
Feilden	"Engineering Design is the use of scientific principles, technical information and imagination in the definition of a mechanical structure, machine or system to perform prespecified functions with the maximum economy and efficiency".	Information transformation process Decision making process Cognitive activity Design objectives	(Feilden 1963)
Caldecote	"...the basic design function... to design a product which will meet the specification, to design it so that it will last and be both reliable and easy to maintain, to design it so that it can be economically manufactured and will be pleasing to the eye."	Design objectives	(Caldecote 1963)
Gregory	"Design science is concerned with the study, investigation and accumulation of knowledge about the design process and its constituent operations. It aims to collect, organize and improve those aspects of thought and information which are available concerning design, and to specify and carry out research in those areas of design which are likely to be of value to practical designers and design organizations."	Information transformation process Decision making process Cognitive activity Design objectives	(Gregory 1966)
Finkelstein	"Design is the creative process which starts from a requirement and defines a contrivance or system and the methods of its realisation or implementation, so as to satisfy the requirement. It is a primary human activity and is central to engineering and the applied arts."	Information transformation process Decision making process Cognitive activity Design objectives	(Finkelstein and Finkelstein 1983)
Luckman	"Design is a man's first step towards the mastering of his environment ... The process of design is the translation of information in the form of requirements, constraints, and experience into potential solutions which are considered by the designer to meet required performance characteristics ... some creativity or originality must enter into the process for it to be called design."	Information transformation process Decision making process Cognitive activity Design objectives	(Luckman 1984)
Archer	"...design involves a prescription or model, the intention of embodiment as hardware, and the presence of a creative step."	Cognitive activity Design objectives	(Archer 1984)
Hubka	"Design science comprises a collection (a system) of logically connected knowledge in the area of design, and contains concepts of technical information and of design methodology. Design science addresses the problem of determining and categorizing all regular phenomena of the systems to be designed, and of the design process. Design science is also concerned with deriving from the applied knowledge of the natural sciences appropriate information in a form suitable for the designer's use."	Information transformation process Decision making process Design objectives	(Hubka and Eder 1987)

Coyne	"Design is a pragmatic discipline concerned with providing a solution within the capacity of the knowledge available to the designer. The design may not be corrector ideal and may represent a compromise, but will meet the original intentions to some degree."	Information transformation process Decision making process Cognitive activity	(Coyne, Rosenman et al. 1990)
Gasparski Strzalecki	"The science of design (should be) understood, just like the science of science, as a federation of sub disciplines having design as the subject of their cognitive interests."	Cognitive activity	(Gasparski and Strzalecki 1990)
Evbuomwan	Design is "the process of establishing requirements based on human needs, transforming them into performance specifications and functions, which are then mapped and converted (subject to constraints) into design solutions (using creativity, scientific principles and technical knowledge), that can be economically manufactured and produced."	Information transformation process Decision making process Cognitive activity Design objectives	(Evbuomwan, Sivalogan athan et al. 1996)
Eder	Eder views design science "as a system of knowledge" that investigates designing.	Information transformation process Decision making process Cognitive activity Design objectives Life cycle information	(Eder 1998)
Roche	"Design is a systematic problem solving process that uses the creativity, knowledge, experience, imagination and originality of humans to transform customer requirements into design specifications, from design specifications into functional requirements, from functional requirements to concepts and there from into detailed design representations of a product whilst optimising aggregate life cycle properties throughout all design phases and across many specialist domains."	Information transformation process Decision making process Cognitive activity Design objectives Life cycle information	(Roche 1999)

Table 2.1. Definitions of design

The process of design consists of a series of actions that can be sequential or parallel conducing to one or more solutions to a design problem. Summarizing the definitions presented in table, the design activity emerges as a threefold process as follows:

- An *information transformation process* from abstract statements of requirements into detailed specifications of a product usually in the form of graphic and textual representations (Feilden 1963; Finkelstein and Finkelstein 1983; Luckman 1984; Coyne, Rosenman et al. 1990; Hubka and Eder 1996; Eder 1998; Roche 1999).
- A *problem solving process* aided by specific methods, methodologies and tools in order to establish a path from the initial requirements to a desirably 'optimal' design solution (Luckman 1984; Pugh 1991; Hubka and Eder 1996; Roche 1999; Gero 2000). The dominant approach of engineers is to "solve problems by

synthesising the 'available knowledge[1]' and make decisions based on that information" (Roche 1999).

- A *decision making process* based on the designer's *implicit knowledge* e.g. experience, knowledge, imagination, originality, creativity and *explicit knowledge* e.g. case bases, personal information databases, colleagues, handbooks, technical reports, vendors, suppliers, design catalogues (Finkelstein and Finkelstein 1983; Luckman 1984; Coyne, Rosenman et al. 1990; Hubka and Eder 1996; Eder 1998; Roche 1999).

Besides the successful achievement of the initial requirements under specific design constraints, the objectives of the engineering design activity also refer to some significant characteristics the final product should have e.g. satisfy consumer demands, fulfil a human need, economically manufactured (Feilden 1963; Pugh 1991; Lang, Dickinson et al. 2002). Furthermore, many researchers emphasize the cognitive activity of the designer when defining engineering design. Indeed, designer characteristics such as skills, experience, knowledge, imagination, originality and creativity have a significant impact on the construction of an 'optimal' solution to a design problem. (Gero indicates that many decision sequences in design are almost totally derived from individual experience and are largely inexplicit (Gero 2000).) Closely linked to concepts such as creativity, problem solving and decision-making, learning is another essential aspect in the process of design (Kolb 1984; Roche 1999). Viewing design as a highly complex skill learned and practised, Roche identifies a synergy between design and learning as shown in figure 2.1 (Roche 1999).

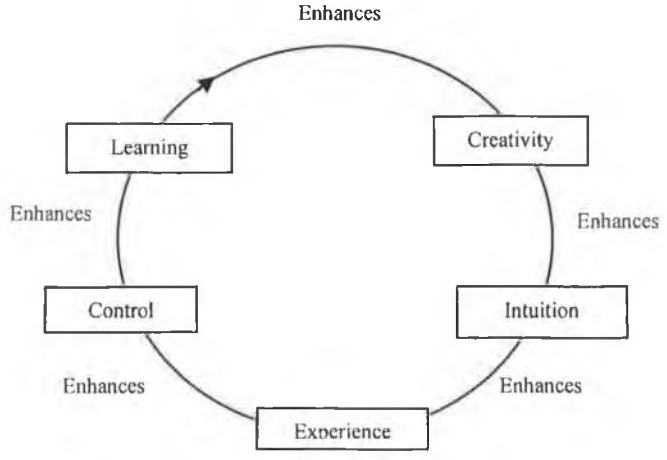


Figure 2.1. Synergy between design and learning (Roche 1999)

[1] The author believes that the word '*knowledge*' should actually refer to '*information*' as the latter one is synthesised in order to attain knowledge (see section 2.2.3 for the meanings associated with the data-information-knowledge value chain).

Moreover, the author believes that the synergy presented in figure 2.1 can be enriched by adding a set of crossed links to illustrate further relationships such as the enhancement of control and experience through the process of learning.

Integrating the design process into the broader process of product realization, some researchers highlight the life cycle information aspect associated with engineering design (Eder 1998; Roche 1999; Lang, Dickinson et al. 2002). The product life cycle consists of a series of generic stages including raw material extraction, manufacture, use and end of life. Figure 2.2 takes a holistic view of the product life cycle showing how life cycle information is acquired through a set of life cycle design information loops (Roche 1999; Man, Diez-Campo et al. 2002).

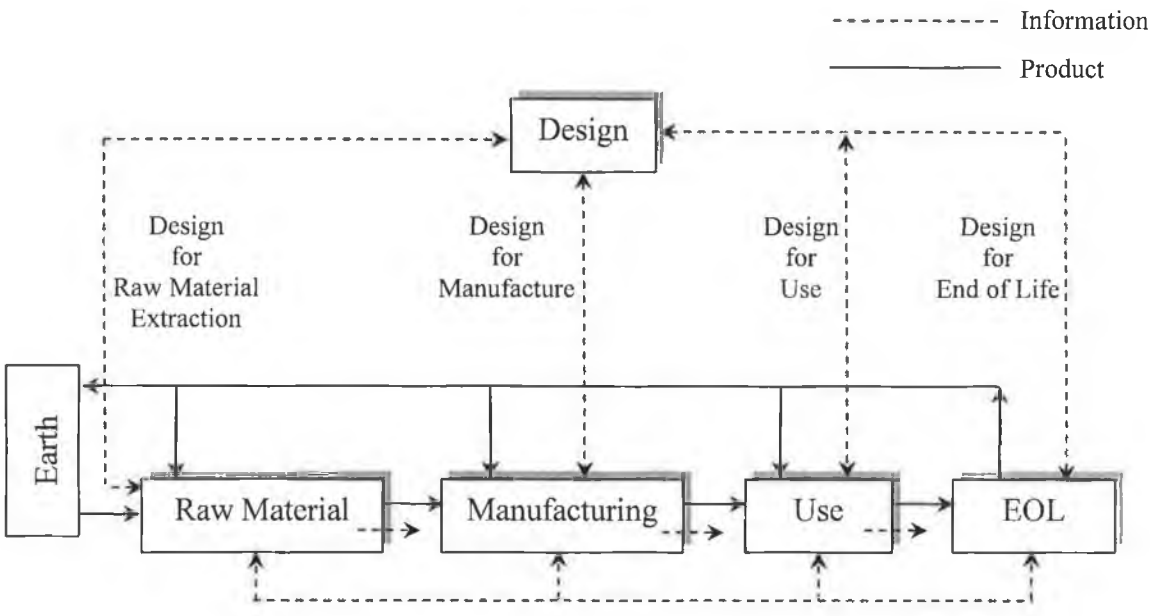


Figure 2.2. Design information loops (Roche 1999)

In this context, the accomplishment of the design goals highly depends on the performance of interrelated activities and processes within the life cycle of the product (Roche 1999). For the successful location of the global 'optima', the engineering design activity needs to be informed and assisted by the other phases of the product realization process by means of knowledge sharing along the supply chain (Eder 1998; Roche 1999).

To summarize, engineering design is a human cognitive activity that transforms information from initial requirements, needs and constraints into detailed specifications of a final product capable of fulfilling these demands and those generated by all stages of the product life cycle. The successful completion of this process highly depends on the problem solving skills of the design engineer and the decisions that he/she makes along the

path to a solution, which in turn are influenced by the implicit and explicit knowledge of the designer.

2.2.2. Distributed Engineering Design

The distributed engineering design organization emerged as a response to market constraints, legislative requirements and business competition (Pahng, Senin et al. 1997; Gammack and Poon 1999; Lang, Dickinson et al. 2002; MacGregor 2002). While the design goal is still "to find a good solution that leads to a quality product with the least commitment of time and resources" (Ullman 1996), the way in which the design activity is carried out and the resources involved in the design process adapted to the complex demands of today's product markets (Hirsch 2000; MacGregor 2002; Thoben 2002). These demands include quick time to market, low cost, high quality, low environmental impact and increased customization (Tomiyama 1994; Kimura 1997; Lang, Dickinson et al. 2002). The new organization of engineering design "distributes its work to the best locations for their execution based on the criteria of people skills, costs and resources" (Gammack and Poon 1999). The distributed engineering design activity involves multiple engineers with heterogeneous skills dispersed over a computer network and requiring concurrent access to multiple system resources. These engineers have to collaborate in a distributed design environment in order to achieve the 'optimal' solution to the current design problem. Indeed, "complex design problems require more knowledge than any single person possesses because the knowledge relevant to a problem is usually distributed among stakeholders" (Arias, Eden et al. 2000).

The distribution of the teams of people involved in the design process can be described on four levels as follows (Weiss 1999; Chira, Chira et al. 2003):

- *Geographical distribution* – design participants are dispersed in different geographic locations;
- *Temporal distribution* – design participants within a distributed environment can be located at different time zones;
- *Functional distribution* – design participants are structured in clusters defined by specific perceptual, effectual and intellectual capabilities;
- *Semantic distribution* – design participants are structured in clusters defined by specific languages and conceptual realities.

Moreover, these levels of distribution also apply to the data, information and knowledge resources supporting the decision-making and problem solving processes of design (Cross 1994; Pahl and Beitz 1996; Bertola and Teixeira 2003).

With participants and information resources distributed over the enterprise, teamwork is becoming increasingly important as design problems are growing to be more complex (Patel, D'Cruz et al. 1997). The primary elements to cooperative work teams are as follows (Pena-Mora, Hussein et al. 2000):

- *Communication* – refers to the exchange of information, events and activities between participants;
- *Co-location* – focuses on the infrastructure to provide a smooth communication among distributed participants;
- *Coordination* – refers to the management of the workflow, resources and communication process;
- *Collaboration* – describes the process of creation of a shared understanding in a distributed environment.

Although an effective communication is a necessity, it is not a sufficient condition to a meaningful cooperation in a distributed environment. Efficient coordination and collaboration are of significant importance while communication is an integral component in the problem solving process (Pena-Mora, Hussein et al. 2000). In order to achieve global 'optima' in distributed design, there is an increasing need for design teams to establish and maintain a cooperative work through a good communication, co-location, coordination and collaboration. Indeed, because design is nowadays a team effort of multidisciplinary groups of participants, "close collaboration among them will accelerate the product development by shortening the development cycle, improving the product quality and reducing investment" (Liu, Tang et al. 2002). One immediate benefit of collaborative work is the coming together of participants with heterogeneous skills (Edmonds, Candy et al. 1994), who, on sharing their skills, expertise and insight, create what is known as distributed cognition. "Distributed cognition emphasizes that the heart of intelligent human performance is not the individual human mind in isolation but the interaction of the mind with tools and artefacts as well as groups of minds in interaction with each other" (Arias, Eden et al. 2000). The collaboration of individuals with different insights, implicit knowledge and skills generally results in the generation of new insights, new ideas and new artefacts. Thus, collaborative designs generally result in work products which are enriched by the multiple personalities of the designers engaged in the design task.

A significant integral component of distributed engineering design is represented by the computer whose primary role evolved from being a tool (in collocated design) to being a medium (in distributed design) "through which two or more geographically separated individuals communicate" (MacGregor 2002). The focus of this new role of the computer

is now on communication, collaboration and the process of designing. Acting as a virtual workplace, the computer represents "a suite of tools, necessary to support the human designer, both for actual design work and communication" (MacGregor 2002).

Characterised by distribution, cooperation, teamwork and being computer supported, the new organization of engineering design aims to achieve benefits such as savings in project life-cycle and costs, added value to team efforts, access to a comprehensive knowledge-based system, reliable communication among design teams and members, flexible access and retrieval of information and timely connectivity with global experts (Pena-Mora, Hussein et al. 2000; Laure 2001; Iheagwara and Blyth 2002). Therefore, "the use of globally distributed engineering design teams continues to increase as companies aim to boost profits and decrease lead times by effectively leveraging knowledge and communication from dispersed locations" (MacGregor, Thomson et al. 2001).

To summarize, the main characteristics of distributed engineering design are as follows:

- The human and physical resources involved in the design process can be geographically, temporally, functionally and semantically distributed over the enterprise.
- The human designers or teams of designers are highly heterogeneous (they may have different intent, background knowledge, area of expertise and responsibility).
- Teamwork is playing a significant role in design projects becoming increasingly large, complex and long in duration.
- The cooperation process among distributed teams of people is crucial for the successful location of the 'optimal' design solution.
- The role of the computer for distributed design is that of a medium facilitating cooperation among distributed designers and also supporting the design process through various applications.

Because of the importance of collaboration for distributed engineering design, the following terms are considered synonyms within this thesis: *distributed engineering design*, *collaborative engineering design* and *distributed collaborative engineering design*. However, the most complete term would be *distributed collaborative engineering design*, where *distributed* refers to the fact that engineering design is performed by a multidisciplinary team of engineers distributed in separate locations (even across various time zones) often working in parallel with different engineering tools and *collaborative* refers to the fact that the product is designed through the collective and joint efforts of

multidisciplinary team of engineers distributed in separate locations (even across various time zones) often working in parallel with different engineering tools and *collaborative* refers to the fact that the product is designed through the collective and joint efforts of many designers supported by the cooperation (communication, co-location, coordination and collaboration) process among them.

2.2.3. Problematic Aspects

The potential benefits of distributed collaborative engineering design are often marginalized by the problems inherent in the process (Huang 1999). Information related problems, coordination and communication problems, knowledge sharing problems and information technology support are among the main issues of distributed collaborative design that need to be addressed.

One of the significant problematic aspects of distributed collaborative design refers to the information resources needed by distributed engineers to support the successful location of the global 'optima' in design. Research shows that engineering design is an information intensive process as all along the design process designers need information to complete their tasks (Baya 1996; Hubka and Eder 1996; Roche 1999). Inherent to the design process, information handling activities include generating, capturing, accessing, transforming, indexing, structuring and analyzing information in order to create an artefact (Baya 1996). A study on the information requirements of engineering designers shows that designers spend in average 18% of their time searching for information, 23% dealing with paperwork, 16% in meetings and only 43% of their time designing (Court, Culley et al. 1993; Court, Culley et al. 1997). Hales indicates that designers spent 53% of their time for information retrieval, planning, cost estimating, reviews and social contact (Hales 1987). Baya gives an information perspective on the design process showing how "old design information can be used in satisfying information needs during a design process resulting in new design information" (Baya 1996), therefore reducing cost and time (see figure 2.3).

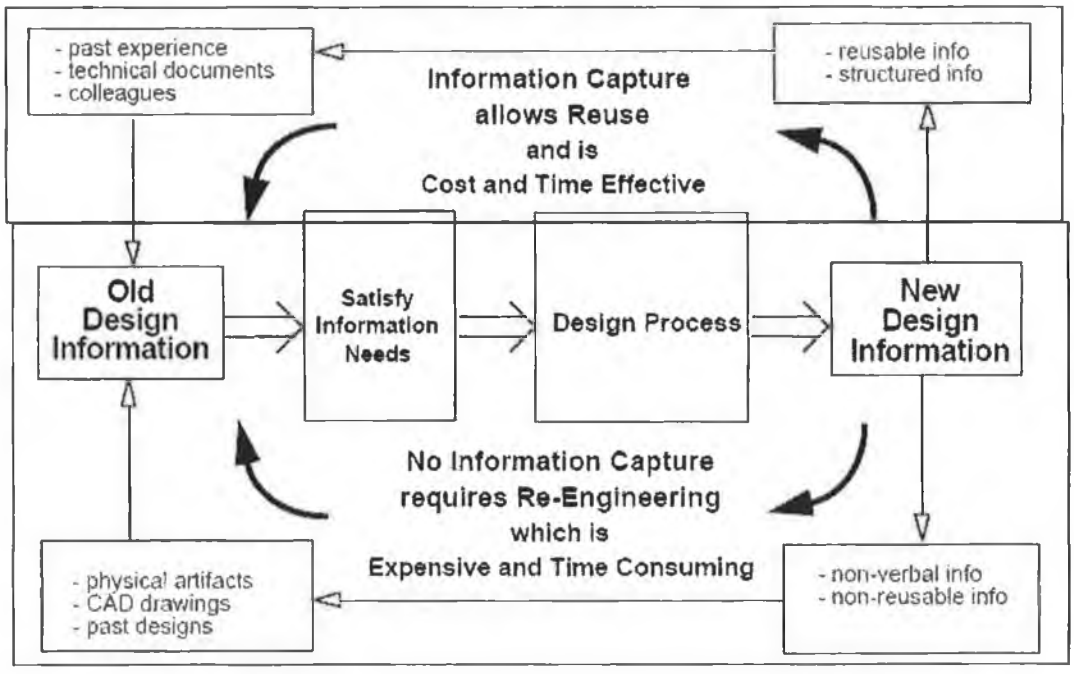


Figure 2.3. An information perspective on the design process (Baya 1996)

Because of the critical importance of information for the success of the design process, the support for access to relevant information is vital. However, the unprecedented growth of information and knowledge[2] has lead to a situation whereby the designer cannot handle such vast quantities of information (VanCuilenburg, Scholten et al. 1991; Fischer 2002), which has the potential of slowing down the design process rather than supporting it to faster reach an 'optimal' solution. Another problem generated by information overloads is finding the information which is relevant for the task at hand in the design process (Viano 2000). If "computers are to be helpful to us at all, it must not be in producing more information – we already have enough to occupy us from dawn to dusk – but to help us to attend to the information that is the most useful or interesting or, by whatever criteria you use, the most valuable information" (Simon 1996). Because engineers conduct the problem-solving process of design based on the available knowledge (that they have ready access to), it is important to ensure that the appropriate information is available at the correct time in the process (Lawson 1990; Cross 1994; Hubka and Eder 1996; Pahl and Beitz 1996). However, "obtaining pertinent, consistent and up-to-date information across a large company is a complex and time-consuming process" (Liang and Huang 2002). Without support for the access of the relevant information at the required time, designers are more likely to generate local rather than global 'optimal' solutions (Coyne, Rosenman

[2] Research shows that the overall amount of information that the world produces is in the range of one to two exabytes (a billion gigabytes) per year (Ho and Tang 2001).

et al. 1990; Lawson 1990; Roche 1999). Furthermore, the distribution of the information resources and the inherent dynamic nature of knowledge in a virtual collaborative design environment add another dimension to the complexity of the management of design data, information and knowledge (Jagdev and Browne 1998; Roche 1999; Pena-Mora, Hussein et al. 2000).

Playing a critical role within the data/information/knowledge hierarchy[3], knowledge must be organised and managed so that human access to it is supported. The philosopher Michael Polanyi and the Japanese organization-learning theorist Ikijuro Nonaka indicate that knowledge has two forms: implicit knowledge and explicit knowledge (Polanyi 1966; Nonaka and Takeuchi 1995; Nonaka and Konno 1998). *Implicit* or *tacit* knowledge represents personal knowledge stored in the bearer's mental structures, having its roots in the private psychological baggage of the individual (e.g. subjective insights, intuitions and hunches). This kind of knowledge cannot be easily formalised, hence it cannot be straightforwardly communicated or shared. *Explicit knowledge* is the knowledge codified and systematically expressed in formal structures compatible with human language (e.g. libraries, archives, databases). Hence, the explicit knowledge represents the kind of knowledge that is communicated and shared. Indeed, these two types of knowledge are crucial to the design process as a whole. Internalised knowledge is more likely to influence the designer subliminally in making good design decisions and is involved in the learning process, which also has strong impacts on the decisions made by the designer (Kolb 1984). Therefore, collaborative design tools and systems need to support the acquisition of relevant information related at levels beyond surface similarities, efficiently manage explicit knowledge as well as to support, promote and enable human's implicit knowledge (as this will affect creativity).

Another problematic aspect of distributed collaborative design relates to the cooperation process among dispersed multidisciplinary teams, which plays a crucial role in the successful location of the global 'optima' in design (Pena-Mora, Hussein et al. 2000; Liu, Tang et al. 2002). Involving collaboration and communication among distributed engineers with different intent and background knowledge as well as coordination of design activities, this cooperation process proves hard to be accomplished by large,

[3] *Data* represents simple facts or individual entities, which organised and structured in a meaningful context generate *information* (Nonaka and Takeuchi 1995). The analysis, synthesis and interpretation of information create meaning and therefore *knowledge* (Tuomi 1999; Shaw 2003; Srinivas 2003).

multidisciplinary projects (Cutkosky, Englemore et al. 1997). Research shows that "team members have difficulty in communicating design intents, decisions and problems across disciplines" (Fruchter, Reiner et al. 1996). The error-prone and time consuming cooperation tasks as well as the unresolved conflicts among different discipline design proposals result in communication difficulties that often "have an impact upon the quality of the final device and the time required to achieve design consensus" (Fruchter, Reiner et al. 1996). Moreover, this communication problem is amplified by the heterogeneous tools used by different engineers. Indeed, "communication between people, organisations and software systems is difficult due to the fact that each of these actors speaks a different language" (Roche 2000). Involving activities such as exchanges of information (communication) and the creation of a shared understanding in a distributed environment (collaboration), the cooperation process among dispersed designers raises difficulties strongly related with those created by the information problematic aspects of collaborative engineering design. Furthermore, the cooperation problem also connects to the heterogeneous software environment used nowadays in distributed design since cooperation also focuses on the infrastructure for a smooth communication (co-location) and on the management of distributed resources and of the workflow (coordination).

Connected to both the information and cooperation problematic aspects of distributed collaborative design, the low level of awareness of other designers and their work within the virtual enterprise forms another facet of the distributed design problem (Nakakoji, Yamamoto et al. 1998; Sclater, Grierson et al. 2001; MacGregor 2002; Thoben, Weber et al. 2002). "Designers generally have a limited awareness and understanding of how the work of other designers within the project - or in similar projects - is relevant to their own part of the design task. The large and growing discrepancy between the amount of such relevant knowledge and the amount any one designer can possibly remember imposes a limit on progress in design" (Nakakoji, Yamamoto et al. 1998).

Closely related to the cooperation problem, the knowledge sharing problem in distributed design refers to the difficulties associated with the exchange of not only data but also knowledge among different actors (Roche 2000). "Although the technology to support exchange of information between participants is available, more content related support is not. [...] Knowledge level models of collaborative distributed design are needed as the situation in which an individual designer contributes to a collaborative distributed design process differs significantly from the situation in which an individual completes an entire design project on his/her own" (Brazier, Moshkina et al. 2001). Because of the different

languages, backgrounds, experience and expertise of distributed design engineers, meaning is particularly difficult to transfer and communicate (Snow 1993; Harvey and Koubek 1998; Brazier, Moshkina et al. 2001; MacGregor 2002; Thoben, Weber et al. 2002). Thus, there is a need for content related support for the exchange of information (Brazier, Moshkina et al. 2001) by establishing compatible understandings of the meanings of the terms exchanged between dispersed distributed design participants.

Finally, the software infrastructure currently employed by distributed collaborative environments plays a crucial role for a domain in which the computer is the workplace but can raise further difficulties for design engineers, some of them in a close relation with problems already identified. A distributed design environment means the existence of different hardware, different operating systems (e.g. Windows, Unix, Linux), different network protocols and architectures (e.g. TCP/IP, FTP, HTTP), different programming languages and compilers (e.g. Java, C, C++), different applications or tools (e.g. CAD/CAM/CAE, PDM, ERP) from different vendors, different databases (e.g. Oracle, Sybase, Microsoft) and multidisciplinary knowledge and Web-based services (Zhao, Deng et al. 2001). Because of this heterogeneity of the distributed design environment, "any collaborative communication or coordination between such diverse and different models, languages and system architectures may prove difficult" (Chao, Norman et al. 2002). Therefore, the integration of all these distributed and heterogeneous resources with the aim of interoperation is imperative (McGuire, Kuokka et al. 1993; Zhao, Deng et al. 2001; Wang, Shen et al. 2002; Anumba, Ren et al. 2003). Anumba et al consider that the facilitation of the flow of information across heterogeneous software tools is a "key aspect of collaborative working between multidisciplinary teams" (Anumba, Ren et al. 2003). However, the syntactic integration of various software tools into the distributed design environment is reduced (Crabtree, Fox et al. 1997; Siemieniuch and Sinclair 1999; Pena-Mora, Hussein et al. 2000) causing a more difficult sharing of knowledge and information in an environment where the "tools are developed by and for experts" (Cutkosky, Englemore et al. 1997). Because "the software tools used in concurrent engineering, requiring specific and dedicated representations, are more concurrent than collaborative" (Roche 2000), a shared understanding among the participants to the distributed design environment (both human and software) needs to be defined. This relates the software infrastructure problem to the cooperation and knowledge sharing problematic aspects of distributed design already mentioned.

To summarize, distributed collaborative engineering design characteristics with detrimental potential include the following:

- The big volume and dispersion of design data, information and knowledge makes the management process more difficult and impacts on the relevance of the information required for different design tasks.
- The cooperation process in a distributed design environment is burdened by the inherent distribution and multidisciplinarity of the design teams involved in a project and by the heterogeneity of the resources supporting the decision making process.
- There is a limited awareness and understanding of other designers and their work within the same project.
- Information and knowledge sharing among dispersed participants to the design process is difficult in a heterogeneous environment.
- Current supporting software infrastructure of distributed design adds another dimension to the complexity of the problematic aspects of collaborative design due to high heterogeneity and low integration.

As already indicated, these problems are highly interconnected by the distributed design data, information and knowledge that needs to be managed, shared and understood by human and machines within a collaborative environment. Computational design support is needed for communications and accessibility to design knowledge, past records and histories. Because design teams "increasingly need to make extraordinary efforts to establish and maintain a sense of communication, co-location, coordination and collaboration [...] software tools and hardware solutions that support such distributed design teams have therefore become a necessity rather than a fad" (Pena-Mora, Hussein et al. 2000). Indeed, "corporations have been seeking to develop a number of information technology (IT) systems to assist with the information management of their business processes" (Liang and Huang 2002) in order to address complex distributed design characteristics such as diverse forms of information, interdisciplinary collaboration and heterogeneous software tools (Wang, Shen et al. 2002). The aim of these IT systems is "to improve the way in which information is gathered, managed, distributed and presented to people" (Liang and Huang 2002). Furthermore, a shared understanding among distributed design participants needs to be created in order to support interoperation and integration of distributed resources. The overall goal of a computational design support system should be

to help multidisciplinary design teams achieve the global 'optimal' design solution in a virtual collaborative environment.

2.3. Current Trends in Software Support

Since the design activity is becoming increasingly computer dependent and there are so many issues concerning the design process in a distributed environment, many researchers have already explored the need for intelligent computational support of collaborative engineering design. Traditional approaches such as the development of integrated sets of tools and the establishment of data standards "are becoming insufficient to support collaborative design practices, because of the highly distributed nature of the design teams, diversity of the engineering tools and the complexity and dynamics of the design environments" (Wang, Shen et al. 2002). Advances in the field of Artificial Intelligence (AI) and successful results in other domains (e.g. medicine, commerce) justify the investigation of intelligent problem-solving methods to support the domain of engineering design. Many of the relevant research studies indicate that the complex activity of distributed cooperative design may be effectively supported by the provision of a collection of interacting autonomous software components incorporating AI specific problem-solving mechanisms. Coordinating the expertise, knowledge and activities of several agents in order to achieve a common goal, such systems (particularly multi-agent systems) are considered suitable for supporting collaborative work in a distributed design environment. Besides distributed agents and multi-agent systems, other emerging technologies used to enable the functionality of new collaborative design systems include ontologies, the Internet and Web technologies. Existing research, projects and applications in the domain of distributed collaborative design mainly based on emerging AI technologies are reviewed next.

2.3.1. Computational Support Systems based on Artificial Intelligence Techniques

PACT: An Experiment in Integrating Concurrent Engineering Systems (Cutkosky, Englemore et al. 1997)

Jointly developed by several research groups (including Stanford University and Hewlett-Packard), the Palo Alto Collaborative Testbed (PACT) is an infrastructure for computer-aided concurrent engineering based on interacting agents. The approach is to integrate existing concurrent engineering systems into a common framework in order to cooperatively solve engineering problems based on knowledge sharing. Able to encapsulate preexisting engineering tools and frameworks, the PACT agents exchange

information and services through an explicit shared model of the design. The PACT architecture uses facilitators to support communication and coordination among agents ("federation architecture"). A facilitator is responsible for routing messages received from various agents to agents able to handle them. Agent interoperability is based on a communication and control language (i.e. KQML) and an interlingua based on first order logic (i.e. KIF). Common ontologies defined for the shared application domain allow knowledge sharing across disciplines. All these PACT ideas and concepts (e.g. knowledge sharing, interoperability, agent-based architectures for concurrent engineering) were tested through the PACT experiments on a robotic manipulator system looking at cooperative design refinement, distributed simulation and distributed redesign. Having a clear beneficial potential for concurrent engineering, PACT serves as a testbed for cooperative research, knowledge sharing and computer-aided engineering.

SHARE: A Methodology and Environment for Collaborative Product Development (Toye, Cutkosky et al. 1993)

The SHARE project provides an open, heterogeneous, network-oriented environment for concurrent engineering aimed at helping design teams to gather, organize, re-access and communicate design information. Team design is viewed as "a process of reaching a "shared understanding" of the domain, the requirements, the artefact, the design process itself and the commitments it entails". The SHARE architecture is a set of interacting agents able to exchange information and services over the Internet using simple commands and an interlingua. These agents represent common engineering tools e.g. designer's CAD tools, a database or other information service, a computational service supporting the engineering process. The ultimate objective of SHARE is to establish a "shared understanding" of the design and design process by facilitating real-time capture, annotation and structuring of information, supporting communication and collaboration among distributed engineers and enabling interoperation among various specialized tools.

SHADE: Technology for Knowledge-Based Collaborative Engineering (McGuire, Kuokka et al. 1993)

The SHADE (SHAred Dependency Engineering) infrastructure aims to support information sharing among engineering tools within multidisciplinary design environments. Supported by this knowledge-based medium, designers are allowed to accumulate and share engineering knowledge through their tools. The three main components of SHADE are a shared ontology, a set of communication protocols and a set

of facilitation services. Considered a direct result of the multidisciplinary nature of engineering, the first component of SHADE (i.e. shared ontology) represents a commonly understood representation and vocabulary for design information. This shared ontology allows tools to exchange design information and aims to support different degrees of knowledge sharing. An agent-based approach is adopted to provide the other two components of SHADE. Firstly, communication protocols are required to allow exchanges of information among tools. Secondly, facilitation services are necessary to make possible the communication and coordination among agents. The validation of the SHADE approach focused on applications such as PACT (Palo Alto Collaborative Testbed), MACE (Mid-deck Active Controls Experiment) and SBD (Simulation Based Design). Part of early research in the area of intelligent computational support for concurrent engineering, the SHADE project was involved in community-wide standards efforts on providing systems and techniques for building ontologies (e.g. KIF – Knowledge Interchange Format) as well as defining a common agent communication language to allow knowledge sharing and exchange (e.g. KQML – Knowledge Query and Manipulation Language).

SINE: Support for Single Function Agents (Brown, Dunskus et al. 1995)

Brown et al propose the SiFA (Single Function Agents) model to investigate design-related primitive problem-solving and interaction patterns. Multiple agents with limited functions cooperate together in order to produce a solution. Each agent is defined by three parameters i.e. function, target and point-of-view. Incorporating design knowledge, conflict resolution knowledge and communication knowledge, the SiFA system also addresses issues such as negotiation and conflict detection, notification and resolution between the various types of agents (e.g. selector, advisor, estimator, evaluator, critic, praiser, suggestor).

An experimental multi-agent environment for engineering design (Shen and Barthes 1996)

Shen and Barthes propose a Distributed Intelligent Design Environment (DIDE) for supporting cooperation among the existing engineering tools as well as information sharing and coordination of the design activities of multidisciplinary design teams in an open design environment. DIDE adopts the multi-agent system architecture consisting of a population of asynchronous cognitive agents each with its own representation of the situation independent from that of other agents. The internal structure of an agent includes a network interface, a communication interface, symbolic agent models, an internal knowledge base and information about the task to be performed. The implementation of

the DIDE system contains agents such as project manager, monitor, database of engineering standard, object-oriented distributed database, graphical tools and design tools. Especially useful for large design projects of complex systems (e.g. automobile, harbour, aircraft), the DIDE multi-agent system features modularity, flexibility, extensibility and transportability.

Collaborative Mechatronic System Design (Fruchter, Reiner et al. 1996)

Developed at Stanford University, the Interdisciplinary Communication Medium (ICM) is an integrated software framework that supports conceptual design by enabling multidisciplinary teams to communicate, share and capture design information (e.g. form models, multi-criteria semantics, project specific information, critique results, explanations, change notifications). ICM formalizes an iterative communication cycle for collaborative teamwork consisting of "propose – interpret – gather – information – critique – explain – change and route notifications". The ICM architecture is based on a shared graphic modelling environment serving as the central interface among designers and as the gateway to network based services. ICM incorporates a Semantic Modeling Extension to facilitate the creation of Interpretation Objects, which capture specific discipline perspectives and annotate features with semantic meaning. Additionally, ICM includes a World Wide Web (WWW) Design Coach agent to explore the large amount of information available on the web by providing mechanisms for WWW document gathering, organizing and reuse. Finally, a Change Notification mechanism enables the creation and organization of design changes linked to the graphic models and routes change notifications.

Discourse Model for collaborative design (Case and Lu 1996)

Intended for use in software environments that offer computational support for collaborative design, the Discourse Model is both a structure and a process for collaborative engineering design that allows groups of designers to cooperate in a distributed and asynchronous way. The proposed model views any interaction between designers as a process of discourse. Supporting interactions between humans and software agents in a blackboard-based workspace, the Discourse Model includes functional requirements for rich modelling environment, agents, distributed and asynchronous information exchange and conflict management. The three distinct phases included in the model are analysis and synthesis, information sharing and conflict management. The workspace described by the model incorporates frames, constraints, semantic networks, libraries of shareable design objects, software agent modules and an electronic messaging

system. Tackling the complexity of large design projects depending on the collaboration among distributed groups of people, the Discourse Model is "implementation independent and applicable to many research and commercial design environments currently available".

Agent-based collaborative design of parts in assembly (Mori and Cutkosky 1998)

Mori and Cutkosky propose an agent-based architecture to support collaborative design by enabling a better coordination of the actions of geographically distributed design teams working on the same design. This is achieved through engineering design agents able to interact with each other, exchange design information and keep track of design changes, dependencies and conflicts. Characterised by reactivity (rather than autonomy), a design agent maintains rule-based knowledge and "consists of a software tool, a wrapper that encapsulates the software tool to communicate with other agents, and a human designer who controls the agent itself". A prototype of the proposed architecture is implemented using a commercial 3D-CAD system (i.e. AutoCAD R14) and the C++ version of JATLite for the agent infrastructure implementation and applied to the design of a CD player.

A-Design: an agent-based approach to conceptual design in a dynamic environment (Campbell, Cagan et al. 1999)

Combining aspects of multi-objective optimization, multi-agent systems and automated design synthesis, A-Design provides a new design generation methodology and search strategy for the conceptual stages of engineering design. A-Design has four basic subsystems i.e. an agent architecture, a representation of the conceptual design problem, a scheme for multi-objective decision making and an evaluation-based iterative algorithm. The collaboration among several different agents (e.g. manager-agents, maker-agents, modification-agents) with individual strategies and preferences supports the generation of a population of design alternatives. An adaptive selection of designs is also incorporated in A-Design to divide the population of design candidates into three categories: pareto optimal, good and poor. In the next iteration, the best design alternatives are selected to be further improved based on multiple objectives, constraints and user preferences while the remaining ones are discarded. The authors also present a series of tests and experiments demonstrating the effectiveness of the A-Design system in locating the design solutions that best meet user specifications.

CAIRO: a concurrent engineering meeting environment for virtual design teams (Pena-Mora, Hussein et al. 2000)

Developed at Massachusetts Institute of Technology, the CAIRO (Collaborative Agent Interaction and synchROnization) system is a software architecture that supports virtual collaboration among geographically separated designers and engineers. The system provides an environment for concurrent engineering which allows distributed designers to collaborate in order to solve a problem. Having access to the Internet and the CAIRO software, design engineers can meet in a virtual meeting room where they can find synchronous communication support, coordinated interaction support and a multi-user interface for collaboration. To support distributed collaboration processes, the meeting control strategies implemented in CAIRO include chairman meeting, free style meeting and lecture meeting. The main component of the CAIRO system is a collection of software agents with the goal of learning "to work with the user to make the meetings as effective as possible". Thus, one software agent tracks the agenda for the user while another agent(s) monitors the collaboration activity and communication requests and can automatically change agenda stages or floor control strategies. Supporting multi-media interactions over computer networks, the CAIRO system can enhance the distributed design process reducing time, personnel and training expenses.

Knowledge level model of an individual designer as an agent in collaborative distributed design (Brazier, Moshkina et al. 2001)

Brazier et al conducted a study to understand and specify the types of reflective reasoning and knowledge involved in distributed collaborative design compared to single agent design. They describe a knowledge-level model of an individual designer as an agent by combining two existing models of agents i.e. a generic model of a design agent and a generic model of a co-operative agent. The proposed model is extended to include explicit knowledge of other participants and the design environment. Furthermore, the extended model includes reasoning about other agents (their knowledge, experience and results) and the need for interaction during a design process. The study also includes an example of a distributed design process (i.e. the distributed design of a web site) to illustrate the types of knowledge and reasoning processes included in the resulting knowledge-level model.

A framework for distributed agent-based engineering design support (Lees, Branki et al. 2001)

Lees et al advocate an agent-based approach to distributed engineering design support in order to provide computational support for concurrent engineering. They propose an agent framework aimed at supporting distributed cooperative design, assisting to maintain design constraints, compiling design histories and supporting the reuse of designs. Various agent types are used to achieve these goals: user interface agents, design critics, service agents and agent communications server. Using several mechanisms for maintaining the contact with the user, the interface agents manage and monitor all user interactions and support communication between agents and the user tailoring the dialogue to the user's level of expertise and performance history. Design critics are intelligent agents which can assist the designer to achieve his/her goal e.g. maintenance of design constraints, recognition of patterns of design activity. Service agents are included in the proposed framework to perform detailed processing operations such as providing an interface to a CAD database. The role of the agent communications server is to support collaborating agents distributed in the design environment to share data and knowledge. The aid of web technology is considered as one of the possibilities for the realisation of the proposed agent-based design environment. The research concludes that intelligent software agents represent a potential effective method to provide the necessary support for the various tasks involved in the complex process of design.

CLOVER: an agent-based approach to systems interoperability in cooperative design systems (Zhao, Deng et al. 2001)

CLOVER is a multi-agent cooperative design environment based on various standards (including ISO standards and widely accepted standard propositions) for agent communication (e.g. FIPA ACL, KQML), shared ontology (e.g. Enterprise Ontology, STEP) and common format for the content of communication (e.g. KIF, EXPRESS, XML). The proposed architecture consists of four types of design agents according to design activities: process management agents (PMA), design task agents (DTA), tool agents (TA) and product data agents (PDA). Based on a general design model, PMAs manage the design process and provide services such as searching, registering and managing of the other three types of agents. DTAs incorporate knowledge on process, product data and existing resources and are able to process some part of product according to related rules in an autonomous way. TAs and PDAs wrap legacy applications (e.g. CAD tools, knowledge-based applications, Web-based services) and legacy engineering

databases respectively to provide interoperability among heterogeneous resources. After also implementing a prototype Web-based CLOVER environment, the authors conclude that agent technology improves interoperability among applications and furthermore can "support higher level dynamic and autonomous cooperation among applications".

An agent-based approach to engineering design (Chao, Norman et al. 2002)

Chao et al propose an agent-based approach to engineering design aimed at reducing redundant design activities and improving coordination among distributed/multidisciplinary design teams (by enabling real-time knowledge sharing between different design tools). They argue that agent attributes such as autonomy and proactiveness can overcome these limitations improving the effectiveness and efficiency of engineering design. The proposed framework includes communication mechanism, mental model, observation mechanism and application. Using an Agent Communication Language, the communication mechanism transports the agent's message to the recipient. This message passing system is built upon Object Request Broker. The mental model reasons with the message content and sends the task to the underlying application which generates the appropriate reply and forwards it to the request agent. The Belief, Desire, Intention model supports the observation mechanism to allow agents to act proactively (to determine which other objects in agents they need to observe and the action to take). A case study on the design of a petrochemical plant with three scenarios was conducted to look at an initial CORBA-based framework for concurrent engineering proposed by the authors and the new agent framework (proposed to overcome the limitations of the first one) and shows that the agent-based approach enhances concurrency in the design process.

The agent-based collaboration information system of product development (Liang and Huang 2002)

Liang and Huang propose an agent-based system called intelligent collaborative agent (ICA) system to support the product development process. The main intended characteristics of the proposed system are intelligence (agents automatically adapt to user preferences and environment changes), autonomy (agents are able to take the initiative) and cooperation (agents are able to cooperate with other agents and the user e.g. make suggestions to modify requests, ask questions for clarification). The architecture of any ICA inside the agent-based system includes an agent body that manages the agent's activities and interactions and an agency indicating the solution resources for the product

development problems. A simple personal computer design is presented as an example operation and scenario to demonstrate the potential of the proposed ICA system.

Supporting evolution in a multi-agent cooperative design environment (Liu, Tang et al. 2002)

Liu et al present a multi-agent cooperative design environment that supports evolutionary design by the means of software agents. A collection of asynchronous semi-autonomous cooperating agents integrate design and engineering tools and human specialists in an open design environment. The proposed multi-agent architecture includes specifications for design agents, knowledge base agents, design tool agents, database maintenance agents, communication agents, process monitor agents, conflict mediation agents, task decomposition agents and management agents. Forming the majority of agents in the design environment, the design agents can demonstrate capabilities such as capturing designers' interests and habits, guiding the designer towards a suitable candidate solution, checking design constraints, maintaining and interpreting knowledge. The knowledge base of a design agent incorporates design knowledge (e.g. rules, functions, methods, algorithms) and social knowledge (e.g. designer's profile, knowledge about other agents). Furthermore, the authors discuss a genetic algorithm based approach to support evolutionary and innovative design abilities. This research presents AI-based techniques, particularly multi-agent systems, as a promising solution for supporting distributed teams of designers with different background knowledge, expertise and responsibility in an open environment.

WebBlow: a Web/agent-based multidisciplinary design optimization environment (Wang, Shen et al. 2003)

WebBlow is a distributed multidisciplinary design optimization (MDO) software environment that integrates composition, coordination, cooperation and adaptation into a design project (i.e. "a collection of multidisciplinary design tools and experts that can be integrated to serve the objective of a design project"). The WebBlow software aims to allow project managers and designers working in a distributed design environment to share product information and knowledge and to support collaboration and coordination of their activities within the context of the design project. To achieve these goals, the proposed software framework integrates emerging technologies such as software agents, Internet/Web, Java and XML. The components of the WebBlow architecture include web-based interface agents, directory facilitator agent, engineering data management agents,

problem solving agents, servlets and XML databases. The research and developmental work carried out by the authors already incorporates system requirements definition, system design and implementation and will continue with the finalization of the prototype implementation and the testing and validation through an industrial partner. Also, ontologies are considered for future work to assure that agents agree on the terminology they use to describe a common domain.

Negotiation within a multi-agent system for the collaborative design of light industrial buildings (Anumba, Ren et al. 2003)

Anumba et al propose a multi-agent system framework to support the interaction and negotiation between specialist design team members with different areas of expertise. The proposed system is called Agent-Based Support for The Collaborative Design of Light Industrial Buildings (ADLIB). The components of the conceptual framework include interface/architecture agents and task agents e.g. structural design agent, building services agent, costing and constructability agent, safety advisory agent. These agents are organised using the peer to peer model with each self-interested agent having the same priority to negotiate design issues. After highlighting the negotiation protocol and strategies adopted by the ADLIB system, the authors present a working prototype (developed using the ZEUS agent development toolkit) which was successfully tested in different design scenarios. Based on a detailed review of multi-agent systems and the design, implementation and evaluation of the ADLIB system, Anumba et al conclude that multi-agent systems "have great potential to improve efficiency of construction collaborative design" and can tackle problems such as "facilitating supply-chain management, procurement management, knowledge management, site management and claims management".

2.3.2. Discussion

Table 2.2 summarizes this extensive literature review of related work in the area of distributed collaborative engineering design support.

Generally, the main issues addressed by researchers concern one or more of the following key characteristics of collaborative design:

- Interdisciplinary collaboration and cooperation among geographically, functionally and semantically distributed designers
- Sharing of diverse and irreducible representations of design data, information and knowledge
- Integration of heterogeneous software tools used in the engineering design process

Research shows that AI-based techniques, particularly software agents and multi-agent systems, are a potential successful solution for supporting distributed multidisciplinary design teams collaborating in a virtual environment to achieve global 'optima'. Characterised by cooperation, autonomy (or semi-autonomy), reactivity and desirably intelligence, agents are mostly used for "supporting cooperation among designers, providing a semantic glue between traditional tools, or for allowing better simulations" (Wang, Shen et al. 2002).

The technological issues associated with the design and development of systems supporting distributed design can be summarised as follows:

- Agent technology is extensively employed in providing computational (and maybe intelligent) support for distributed design process operation and management.
- Specifying content specific agreements, ontologies are used by some of the reviewed systems to allow knowledge sharing and reuse.
- Web technology is sometimes incorporated in the proposed framework to significantly extend the access to information structures.

If addressed, the implementation phase of the proposed systems generally employs traditional programming languages such as Java, C++ or Lisp but agent development toolkits (e.g. ZEUS, JATLite) are sometimes used.

System	Year	Proposer(s)	Objectives	Features	Technologies	Reference
PACT	1993	Stanford University Lockheed Hewlett-Packard Enterprise Integration Technologies	Knowledge sharing; Interoperability; Integration of design tools via shared design models.	Federation architecture; Wrapper for legacy system integration	Agents Ontologies PDES/STEP KQML KIF	(Cutkosky, Englemore et al. 1997)
SHARE	1993	Stanford University Enterprise Integration Technologies	Design information sharing (access, organization and communication); Communication and collaboration among designers; Interoperation among design tools.	Federation architecture; Asynchronous communication; Web-based tools for information management.	Agents Internet/Web E-mail	(Toye, Cutkosky et al. 1993)
SHADE	1993	Stanford University Hewlett-Packard Enterprise Integration Technologies	Information sharing; Knowledge-based medium for distributed designers.	Shared knowledge representation; Protocols supporting information exchange; Facilitation services for communication and coordination among agents.	Agents Ontologies KQML KIF Ontolingua	(McGuire, Kuokka et al. 1993)
SiFA	1995	Worcester Polytechnic Institute	Design-related problem solving and interaction patterns	Single Function Agents	Agents Internet CLIPS	(Brown, Dunskus et al. 1995)
DIDE	1996	Université de Technologie de Compiègne	Information sharing; Integration of engineering tools; Coordination of design activities.	Autonomous agents approach; Conflicts detection and resolution; Legacy system integration.	Agents Ontologies OSACA LISP MOSS MATISSE EDBMS	(Shen and Barthes 1996)
ICM	1996	Stanford University	Capturing, communication and sharing of design information; Integration of multi-criteria and multi-disciplinary representation and reasoning.	Shared graphical modeling; Iterative communication model.	Agents Semantic Modeling AutoCAD Prokappa C, Mosaic Internet, E-mail	(Fruchter, Reiner et al. 1996)
Agent-based collaborative design	1998	Toshiba Corporation Stanford University	Support collaborative design; Coordination of the actions of designers working on a common design.	Reactive agent approach; Pareto optimality.	Agents AutoCAD R14 ObjectARX C++, JATLite	(Mori and Cutkosky 1998)
A-Design	1999	Carnegie Mellon	Methodology for design generation;	Multi-objective decision making;	Agents	(Campbell, Cagan et

		University	Search strategy for conceptual design.	Automated design synthesis	Internet Lisp	al. 1999)
CAIRO	2000	Massachusetts Institute of Technology Oracle Corporation	Virtual collaboration among distributed designers; Information exchange across the Internet.	Synchronous communication support; Coordinated interaction support; System modularity and extensibility; Multimedia communication infrastructure; Multi-user interface for collaboration.	Agents Internet Java	(Pena-Mora, Hussein et al. 2000)
CLOVER	2001	Beijing University National Research Council Canada	Design process management; Interoperability among applications; Autonomous cooperation.	Multi-agent system approach (with autonomous agents); Middle-wares between general agents and legacy applications; Use of ISO standards.	Agents Ontologies KQML STEP EXPRESS XML, JATLite	(Zhao, Deng et al. 2001)
Agent-based framework for engineering design	2002	Coventry University	Improve coordination among distributed/multidisciplinary design teams; Reduce redundant design activities;	Multi-agent systems; Autonomous and pro-active agent approach; Mobile agents; Belief, Desire, Intention mental model;	Agents CORBA Java/C++ STEP AP231	(Chao, Norman et al. 2002)
WebBlow	2003	National Research Council Canada University of Western Ontario	Sharing of product information and knowledge; Collaboration and coordination of designers and their activities.	Multidisciplinary design optimization; Process and performance simulation; Web-based user interfaces; Agent-oriented approach for Web-based collaborative design systems.	Agents Applets Servlets Java, C/C++ Apache Tomcat XML Ontologies (future work)	(Wang, Shen et al. 2003)
ADLIB	2003	Loughborough University University of Hong Kong	Representation of activities and processes involved in collaborative design; Interaction and negotiation between designers.	Multi-agent system approach; Peer to peer agent organization.	Agents Ontologies ZEUS	(Anumba, Ren et al. 2003)

Table 2.2. Summary of proposed systems supporting distributed collaborative design based on Artificial Intelligence techniques

Started more than ten years ago, research in the area of distributed collaborative systems includes a large number of projects which only propose an architecture or framework without providing a viable implementation and testing phase. Difficulties associated with the development of collaborative design systems include the creation of a shared ontology that would enable knowledge-level communication in a distributed environment, the integration of the various design tools and the provision of a cooperation model among interacting participants with different needs and diverse areas of expertise. Furthermore, knowledge management in distributed design needs to be better addressed in order to "capture and reuse the existing designs, help them to adapt to new requirements, and maintain the design knowledge as a corporate asset" (Wang, Shen et al. 2002).

To conclude, related research shows that agent technology is a promising approach for collaborative design systems although most of the proposed systems, tools or applications are highly theoretical rather than practical proposals being still under proof-of-the-concept prototype development stage.

2.4. High-level Specification of an Intelligent Architecture to support Distributed Collaborative Engineering Design

Any computational system intended to support distributed collaborative engineering design should address the main problems designers have when collectively working in a distributed environment in order to achieve global 'optima' (see section 2.3). Therefore, the requirements of such a system can be summarised as follows:

- The system should efficiently manage the design data, information and knowledge circulated in a distributed environment in order to support the designer in finding, accessing and retrieving the information needed in the various design stages.
- The system should aid distributed and multidisciplinary design teams in establishing and maintaining cooperation through an effective communication, co-location, coordination and collaboration.
- The system should offer content related support for the exchange of data, information and knowledge in order to enable knowledge sharing and reuse in a distributed design environment.
- The system should address the integration of heterogeneous software tools used by designers by enabling the flow of information in the distributed environment.

The literature review in the area of collaborative design systems (see section 2.4) emphasizes the need for intelligent forms of technological support for distributed design.

Moreover, software agents and multi-agent systems represent an effective method for providing support for the various tasks of distributed design (Shen and Barthes 1996; Lees, Branki et al. 2001; Zhao, Deng et al. 2001; Chao, Norman et al. 2002; Liu, Tang et al. 2002; Wang, Shen et al. 2003). It is intended to demonstrate that, through features such as autonomy, cooperation, reactivity and learning, agent technology is a potential solution for distributed design issues such as:

- Interdisciplinary cooperation among distributed designers
- Exchange of design data, information and knowledge
- Integration of heterogeneous software tools

Furthermore, cooperation among software agents is crucial for the efficient functionality of any collaborative design system. In a multi-agent system, the agents must coordinate their activities, negotiate if a conflict occurs and be able to communicate with other agents. A meaningful agent interoperation highly depends on the availability of a common vocabulary for all design-related aspects which can be offered by a common shared *ontology*. Ontologies define content specific agreements to facilitate knowledge sharing and reuse among systems that submit to the same ontology/ontologies by the means of ontological commitments (Gruber 1995; Spyns, Meersman et al. 2002). They describe concepts and relations assumed to be always true independent from a particular domain by a community of humans and/or agents that commit to that view of the world (Guarino 1997). Therefore, the specification of an architecture to support distributed design should include the design of an ontology library representing the knowledge from the collaborative design domain.

Figure 2.4 summarizes these specification requirements of an intelligent collaborative design system in a high-level view of the architecture.

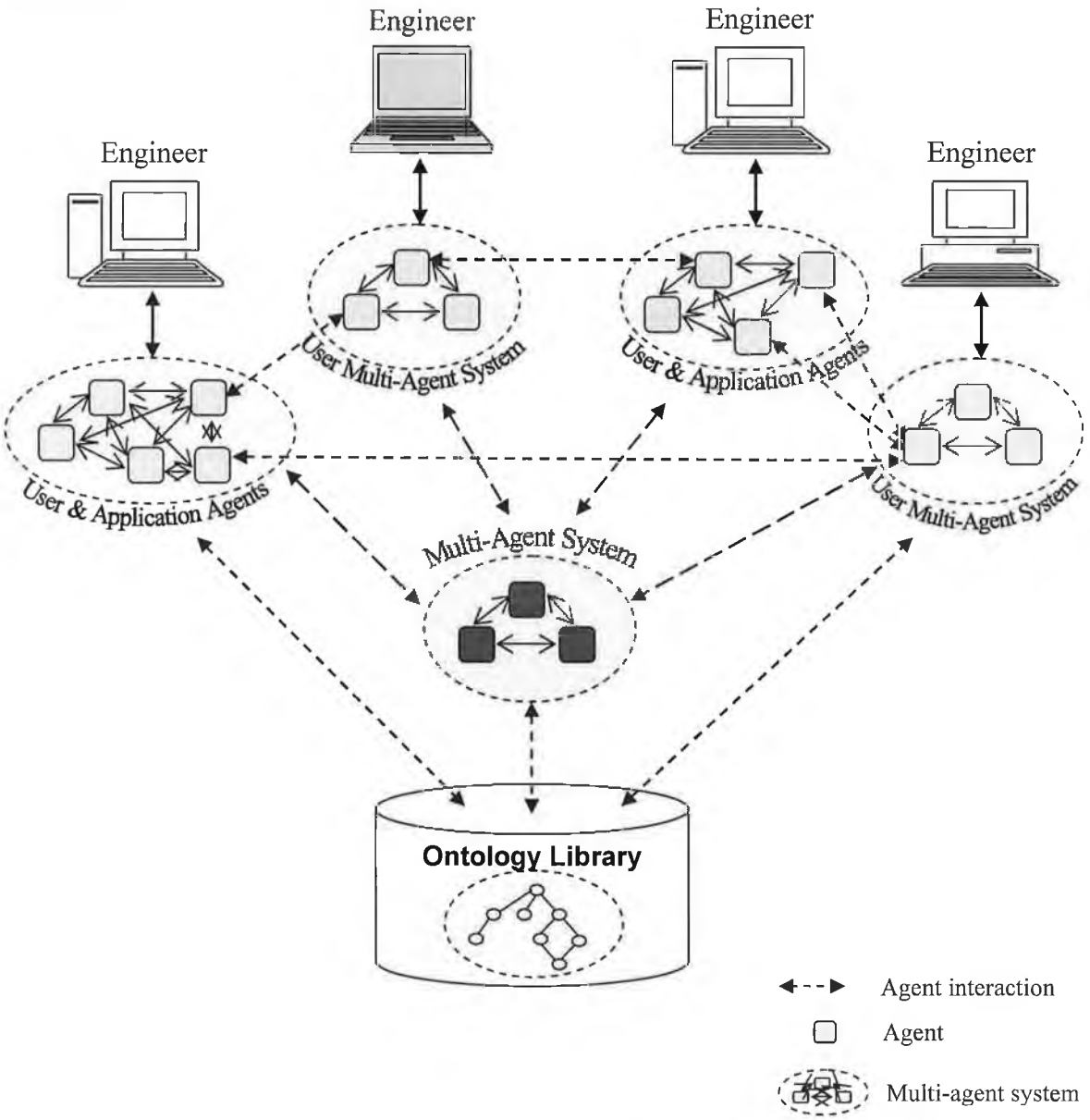


Figure 2.4. High-level view over the architecture of the proposed intelligent collaborative design system

Software agents and multi-agent systems are used to facilitate interoperation among distributed resources and to support the designer in accessing the information he/she needs for the task at hand in a preferred and suitable format. The proposed architecture employs ontologies (more specifically, an ontology library) to support knowledge sharing, reuse and integration in the distributed design environment. Types of software agents needed in the distributed collaborative design system include user specific agents and application agents as well as agents working closely with the ontology library in order to translate information from one format to another and to store, update, access and maintain knowledge. Of course, the cooperation process among these agents actually supports the designer in his/her task and ultimately aids the distributed design process. A close investigation of

agents and multi-agent systems is necessary in order to provide a detailed specification of the proposed system and to feed the design and implementation process of the architecture. The definition of an agent and a multi-agent system, agent typology, agent architectures, agent-oriented methodologies, languages and environments are among the issues to be further clarified.

2.5. Conclusions

Distributed collaborative engineering design is an information intensive activity based on the cooperation process of dispersed and multidisciplinary design teams with the aim of achieving a global 'optimal' design solution. Problematic aspects of this activity such as diverse and complex forms of information, interdisciplinary collaboration and heterogeneous software tools emphasize the need for computational support of distributed design teams working together in a computer-based medium. Highlighting the benefits of *intelligent* technological support, the review of the current approaches to aid distributed collaborative engineering design indicates the suitability of multi-agent systems and ontologies for supporting such a computational system. Agent properties such as autonomy, pro-activeness and cooperation can overcome current distributed engineering design limitations by enabling interoperation among distributed resources. Representing techniques to manage the complexity inherent in software systems, agents and multi-agent systems are appropriate for domains in which data, control, expertise and/or resources are inherently distributed (Jennings, Sycara et al. 1998; Oliveira, Fischer et al. 1999). Adding a semantic link between distributed and heterogeneous resources, ontologies can support collaborative engineering design by enabling knowledge sharing, reuse and integration. Indeed, research shows that ontologies are currently very popular mainly within fields that require a knowledge-intensive approach to their methodologies and system development, such as knowledge engineering (Gruber 1993; Uschold and Gruninger 1996; Gaines 1997), knowledge representation (Artala, Franconi et al. 1996), qualitative modeling, language engineering, database design (Van de Riet 1998), information modeling (Weber 1997), information integration (Bergamaschi, Castano et al. 1998; Mena 1998), knowledge management and organization and agent-based design (Nwana 1996; Odell 2000; Chaib-draa and Dignum 2002).

Based on the problematic aspects of distributed collaborative engineering design that need to be addressed and on the literature review of related work, a high-level specification of a supporting computational system for distributed engineering design has been proposed. The main supporting technology of the proposed system is represented by software agents

and multi-agent systems. However, the need of other emerging technologies such as ontologies has been already identified to fully support the functionality of the system. A review of agent issues such as definitions, architectures, languages and environments is necessary to further detail and complete the specification of the proposed architecture and to implement the proposed system.

References

Anumba, C. J., Z. Ren, A.Thorpe, O. O. Ugwu and L.Newnham (2003). "Negotiation within a multi-agent system for the collaborative design of light industrial buildings." Advances in Engineering Software 34: 389-401.

Archer, L. (1984). Systematic Method for Designers. Developments in Design Methodology. N. Cross. London, John Wiley & Sons Ltd: pp 57 - 82.

Arias, E., H. Eden, G. Fischer, A. Gorman and E. Scharff (2000). " Transcending the Individual Human Mind - Creating Shared Understanding through Collaborative Design." ACM transactions on Computer-Human Interaction Vol. 7, No. 1: 84 - 113.

Artala, A., E. Franconi, N. Guarino and L. Pazzi (1996). "Part-Whole Relations in Object-Centred Systems: an Overview." Data and Knowledge Engineering 20(3): 347-383.

Baya, V. (1996). Information handling behavior of designers during conceptual design: three experiments. Department of Mechanical Engineering, Stanford University.

Bergamaschi, S., S. Castano, S. D. C. d. Vimercati and M. Vincini (1998). An Intelligent Approach to Information Integration. Formal Ontology in Information System. N. Guarino. Amsterdam, IOS Press.

Bertola, P. and J. C. Teixeira (2003). "Design as a knowledge agent. How design as a knowledge process is embedded into organizations to foster innovation." Design Studies 24(2): 181-194.

Brazier, F. M. T., L. V. Moshkina and N. J. E. Wijngaards (2001). "Knowledge level model of an individual designer as an agent in collaborative distributed design." Artificial Intelligence in Engineering 15: 137-152.

Brown, D. C., B. V. Dunskus, D. L. Grecu and I. Berker (1995). SINE: Support For Single Function Agents. Proceedings of the International Conference on Artificial Intelligence in Engineering, Udine, Italy.

Caldecote, V. (1963). "The Design Team in Relation to The Individual Designer." The Practice of and Education for Engineering Design 178(Part B): 16-19.

Campbell, M., J. Cagan and K. Kotovsky (1999). "A-Design: An Agent-Based Approach to Conceptual Design in a Dynamic Environment." Research in Engineering Design 11(3): 172-192.

Case, M. P. and S. C.-Y. Lu (1996). "Discourse Model for collaborative design." Computer-Aided Design 28(5): 333-345.

Chaib-draa, B. and F. Dignum (2002). "Trends in Agent Communication Language." Computational Intelligence 18(2).

Chao, K.-M., P. Norman, R. Anane and A. James (2002). "An agent-based approach to engineering design." Computers in Industry 48: 17-27.

Chira, O., C. Chira, D. Tormey, A. Brennan and T. Roche (2003). An agent-based approach to knowledge management in distributed design. 10th ISPE International Conference on Concurrent Engineering: Research and Applications, Madeira Island, Portugal.

Court, A. W., S. J. Culley and C. A.McMahon (1997). "The influence of information technology in new product development: Observations of an empirical study of the access of engineering design information." International Journal of Information Management 17(5): 359-375.

Court, A. W., S. J. Culley and C. A. McMahon (1993). The Information Requirements of Engineering Designers. International Conference on Engineering Design, The Hague.

Coyne, R. D., M. A. Rosenman, M. A. Radford, M. Balachandran and J. S. Gero (1990). Knowledge based Design Systems, Addison Wesley.

Crabtree, R. A., M. S. Fox and N. K. Baid (1997). "Towards an Understanding of Collaborative Design Activities." Research in Design Engineering 9: 70-84.

Cross, N. (1994). Engineering Design Methods, J. Wiley & Sons.

Cutkosky, M. R., R. S. Englemore, R. E. Fikes, M. R. Genesereth, T. R. Gruber, W. S. Mark, J. M. Tenenbaum and J. C. Weber (1997). PACT: An Experiment in Integrating Concurrent Engineering Systems. Readings in Agents. M. N. Huhns and M. P. Singh. San Francisco, CA, USA, Morgan Kaufmann: 46-55.

Eder, W. E. (1998). "Design Modelling - A Design Science Approach (And Why Does Industry Not Use It?)." Journal of Engineering Design 9(4).

Edmonds, E. A., L. Candy, R. Jones and B. Soufi (1994). "Support for Collaborative Design : Agents and Emergence." Communications of the ACM 37(7).

Evbuomwan, N., S. Sivaloganathan and A. Jebb (1996). "A survey of design philosophies, models, methods and systems." Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture 210: 301-319.

Feilden, G. B. R. (1963). Engineering Design. London, Report of Royal Commission - HMSO.

Finkelstein, L. and A. C. W. Finkelstein (1983). Review of Design Methodology. IEE Proceedings.

Fischer, G. (2002). "Knowledge Management : Problems, Promises, Realities and Challenges." IEEE Intelligent Systems.

Fruchter, R., K. A. Reiner, G. Toye and L. J. Leifer (1996). "Collaborative Mechatronic System Design." Concurrent Engineering: Research and Applications 4(4): 401-413.

Gaines, B. (1997). "Editorial: Using Explicit Ontologies in Knowledge-based System Development." International Journal of Human-Computer Systems 46: 181.

Gammack, J. and S. Poon (1999). Communication Media for Supporting Distributed Engineering Design. 32nd Hawaii International Conference on System Sciences, Hawaii.

Gasparski, W. and A. Strzalecki (1990). "Contributions to design science: Praxeological perspective, Design Methods and Theories." Journal of DMG 24(2): 1186-1194.

Gero, J. (2000). "Computational Models of Innovative and Creative Design Process." Technological Forecasting and Social Change 64: 183-196.

Gregory, S. (1966). The Design Method. London, Butterworth & Co Ltd.

Gruber, T. R. (1993). "A Translation Approach to Portable Ontology Specification." Knowledge Acquisition 5(2): 199-220.

Gruber, T. R. (1995). "Toward Principles for the Design of Ontologies Used for Knowledge Sharing." International Journal of Human and Computer Studies 43(5/6): 907-928.

Guarino, N. (1997). Semantic Matching: Formal Ontological Distinctions for Information Organization, Extraction, and Integration. Summer School on Information Extraction, Frascati, Italy, July 14-19.

Hales, C. (1987). Analysis of the Engineering Design Process in an Industrial Context. Department of Engineering. Cambridge, University of Cambridge.

Harvey, C. M. and R. J. Koubek (1998). "Toward a Model of Distributed Engineering Collaboration." Computers & Industrial Engineering 35(1-2): 173-176.

Hirsch, B. (2000). Extended Products in Dynamic Enterprises", E-Business: Key Issues, Applications and Technologies,: 622-628.

Ho, J. and R. Tang (2001). "Towards an Optical Resolution to Information Overload : An Infomediary Approach." ACM.

Huang, J. (1999). "Knowledge sharing and innovation in distributed design: implications of internet-based media on design collaboration." International Journal of Design Computing: Special Issue on Design Computing on the Net (DCNet'99).

Hubka, V. and E. Eder (1987). "A Scientific Approach to Engineering Design." Design Studies 8(3): 123-137.

Hubka, V. and E. Eder (1996). Design Science, Springer-Verlag.

Iheagwara, C. and A. Blyth (2002). "Evaluation of the performance of ID systems in a switched and distributed environment the RealSecure case study." Computer Networks.

Jagdev, H. and J. Browne (1998). "The Extended Enterprise-A context for Manufacturing." Production Planning and Control 9(3): 326-339.

Jennings, N. R., K. P. Sycara and M. Wooldridge (1998). "A Roadmap of Agent Research and Development." Journal of Autonomous Agents and Multi-Agent Systems 1(1): 7-36.

Kimura, F. (1997). Inverse manufacturing: From Products to Services. Managing Enterprises - Stakeholders, Engineering, Logistics and Achievement First International Conference Proceedings, MEP Ltd, London,.

Kolb, D. (1984). Experiential Learning: Experience as the Source of Learning and Development, Prentice-Hall.

Lang, S. Y. T., J. Dickinson and R. O. Buchal (2002). "Cognitive factors in distributed design." Computers in Industry 48: 89-98.

Laure, E. (2001). "OpusJava: A Java framework for distributed high performance computing." Future Generation Computer Systems 18: 235-251.

Lawson, B. (1990). How Designers Think 2nd Ed.

Lees, B., C. Branki and I. Aird (2001). "A framework for distributed agent-based engineering design support." Automation in Construction 10: 631-637.

Liang, W.-Y. and C.-C. Huang (2002). "The agent-based collaboration information system of product development." International Journal of Information Management 22: 211-224.

Liu, H., M. Tang and J. H. Frazer (2002). "Supporting evolution in a multi-agent cooperative design environment." Advances in Engineering Software 33: 319-328.

Love, T. (2002). "Constructing a coherent cross-disciplinary body of theory about designing and designs: some philosophical issues." Design Studies 23(3): 345-361.

Luckman, J. (1984). An Approach to the Management of Design. Developments in Design Methodolgy. N. Cross. London, John Wiley & Sons Ltd: 83-97.

MacGregor, S. P. (2002). "New Perspectives for Distributed Design Support." The Journal of Design Research 2(2).

MacGregor, S. P., A. L. Thomson and N. P. Juster (2001). Information sharing within a distributed, collaborative design process: a case study. Proceedings of Design Engineering Technical Conferences and Computers (DETC'01) and Information in Engineering Conference, Pittsburgh, Pennsylvania.

Man, E., J. E. Diez-Campo, C. Chira and T. Roche (2002). Product Life Cycle Design using the DFE Workbench. 5th IFIP International Conference on Information Technology for Balanced Automation Systems in Manufacturing and Services (BASYS), Cancun, Mexico.

McGuire, J. G., D. R. Kuokka, J. C. Weber, J. M. Tenenbaum, T. R. Gruber and G. R. Olsen (1993). "SHADE: Technology for knowledge-based collaborative engineering." Concurrent Engineering: Research and Applications 1(3).

Mena, E., Kashyap, V., Illarramendi, A., Sheth, A. (1998). Domain Specific Ontologies for Semantic Information Brokering on the Global Information Infrastructure. Formal Ontology in Information Systems. N. Guarino. Amsterdam, IOS Press.

Mori, T. and M. R. Cutkosky (1998). Agent-based collaborative design of parts in assembly. Proceedings of Design Engineering Technical Conference '98, Atlanta, Georgia, USA.

Nakakoji, K., Y. Yamamoto, T. Suzuki, S. Takada and M. Gross (1998). "From Critiquing to Representational Talkback: Computer Support for Revealing Features in Design." Knowledge-Based Systems Journal 11(7-8): 457-468.

Nonaka, I. and N. Konno (1998). "The Concept of "Ba": Building a Foundation for Knowledge Creation." California Management Review 40(3): 40-54.

Nonaka, I. and H. Takeuchi (1995). The Knowledge Creating Company: How Japanese Companies Create the Dynasties of Innovation. New York, Oxford University Press.

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

Odell, J. (2000). Agent Technology - Green Paper, OMG - Agent Platform Special Interest Group.

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

Pahl, G. and W. Beitz (1996). Engineering a Systematic Approach, Springer.

Pahng, F., N. Senin and D. Wallace (1997). Modelling and Evaluation of Product Design Problems in a Distributed Design Environment. DETC'97: 1997 ASME Design Engineering Technical Conferences, Sacramento, California.

Patel, U., M. J. D'Cruz and C. Holtham (1997). "Collaborative Design for Virtual Team Collaboration : A Case Study of Jostling on the Web." ACM.

Pena-Mora, F., K. Hussein, S. Vadhavkar and K. Benjamin (2000). "CAIRO: a Concurrent Engineering Meeting Environment for Virtual Design Teams." Artificial Intelligence in Engineering 14: 202-219.

Polanyi, M. (1966). The Tacit Dimension, Doubleday & Co.

Pugh, S. (1991). Total Design: Integrated Methods for Successful Product Engineering, Addison-Wesley Publishing UK.

Roche, C. (2000). "Corporate ontologies and concurrent engineering." Journal of Materials Processing Technology 107: 187-193.

Roche, T. (1999). Development of a Design for the Environment Workbench. CIMRU, Industrial Engineering Dept. Galway, UCG.

Sclater, N., H. Grierson, W. J. Ion and S. MacGregor (2001). "Online Collaborative Design Projects: Overcoming Barriers to Communication." International Journal of Engineering Education 17(2): 189-196.

Shaw, N. C. (2003). Knowledge Management Basics, ICASIT - International Centre for Applied Studies in Information Technology. 2003.

Shen, W. and J.-P. A. Barthes (1996). "An experimental multi-agent environment for engineering design." International Journal of Cooperative Information Systems 5(2-3): 131-151.

Siemieniuch, C. E. and M. Sinclair (1999). "Real-time collaboration in design engineering: an expensive fantasy or affordable reality?" Behaviour & Information Technology 18(5): 361-371.

Simon, H. A. (1996). The Sciences of the Artificial. Cambridge Mass., MIT Press.

Snow, C. P. (1993). The Two Cultures. Cambridge, Cambridge University Press.

Spyns, P., R. Meersman and M. Jarrar (2002). Data Modelling versus Ontology Engineering, ACM SIGMOD Record. 31.

Srinivas, H. (2003). Knowledge Management, THE GLOBAL DEVELOPMENT RESEARCH CENTER. 2003.

Thoben, K.-D. (2002). Extended Products: Evolving Traditional Product Concepts. 7th International Conference on Concurrent Enterprising.

Thoben, K.-D., F. Weber and M. Wunram (2002). "Barriers in Knowledge Management and Pragmatic Approaches." Studies in Informatics and Control 11(1).

Tomiyama, T. (1994). The Technical Concept of Intelligent Manufacturing Systems (IMS). Tokyo, University of Tokyo.

Toye, G., M. R. Cutkosky, L. J. Leifer, J. M. Tenenbaum and J. Glicksman (1993). SHARE: A Methodology and Environment for Collaborative Product Development. Post-Proceedings of the IEEE Infrastructure for Collaborative Enterprises.

Tuomi, I. (1999). Data Is More Than Knowledge: Implications of the Reversed Knowledge Hierarchy for Knowledge Management and Organizational Memory. The 32nd Hawaii International Conference on System Sciences, Maui, Hawaii.

Ullman, D. G. (1996). Mechanical Design Process, McGraw-Hill.

Uschold, M. and M. Gruninger (1996). "Ontologies:Principles, Methods and Applications." The Knowledge Engineering Review 11(2): 93-136.

Van de Riet, R., Burg, H., Dehne, F. (1998). Linguistic Issues in Information System Design. Formal Ontology in Information System. G. Nicola. Amsterdam, IOS Press.

VanCuilenburg, J. J., O. Scholten and G. Noomen (1991). Stiinta Comunicarii.

Viano, G. (2000). Adaptive User Interface for Process Control based on Multi-Agent approach. AVI 2000, Palermo, Italy.

Wang, L., W. Shen, H. Xie, J. Neelamkavil and A. Pardasani (2002). "Collaborative conceptual design - state of the art and future trends." Computer Aided Design 34: 981-996.

Wang, Y. D., W. Shen and H. Ghenniwa (2003). "WebBlow: a Web/agent-based multidisciplinary design optimization environment." Computers in Industry 52: 17-28.

Weber, R. (1997). Ontological Foundations of Information Systems. Melbourne, Coopers and Lybrand.

Weiss, G. (1999). Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. London, MIT Press.

Zhao, G., J. Deng and W. Shen (2001). "CLOVER: an agent-based approach to systems interoperability in cooperative design systems." Computers in Industry 45: 261-276.

Chapter 3

Multi-Agent Systems

3.1. Background

Over the last years, autonomous agents have been the focus of researchers and developers from disciplines such as Artificial Intelligence (AI), object-oriented programming, concurrent object-based systems and human-computer interface design (Jennings, Sycara et al. 1998). Indeed, software agents and Multi-Agent Systems (MAS) represent an important and fast growing area of AI and more generally of Computer Science (Lesser 1995; Nwana 1996; Bradshow 1997; Green, Hurst et al. 1997; Jennings 2000). The introduction of agents in AI "is partly due to the difficulties that have arisen when attempting to solve problems without regard to a real external environment or to the entity involved in that problem-solving process" (Luck, McBurney et al. 2003).

MAS form one the three broad areas of a relatively youthful field of AI called Distributed Artificial Intelligence (DAI), the other two research areas being Distributed Problem Solving and Parallel AI (see Figure 3.1) (Nwana and Ndumu 1999). Dealing with collections of interacting, coordinated knowledge-based processes (Gasser 1998), DAI demonstrates a distinct feature through the communication and coordination among intelligent and autonomous agents during a problem solving process. This approach decomposes the complexity of the domain problem (agents work together in a problem solving team as opposed to a single agent dealing with a problem) and enhances the system's performance (Chu, Srihari et al. 1996). Inheriting characteristics from AI and DAI, MAS incorporate potential benefits such as modularity, speed, reliability, operation at knowledge level, easier maintenance, reusability and platform independence (Nwana 1996; Chaib-draa and Dignum 2002).

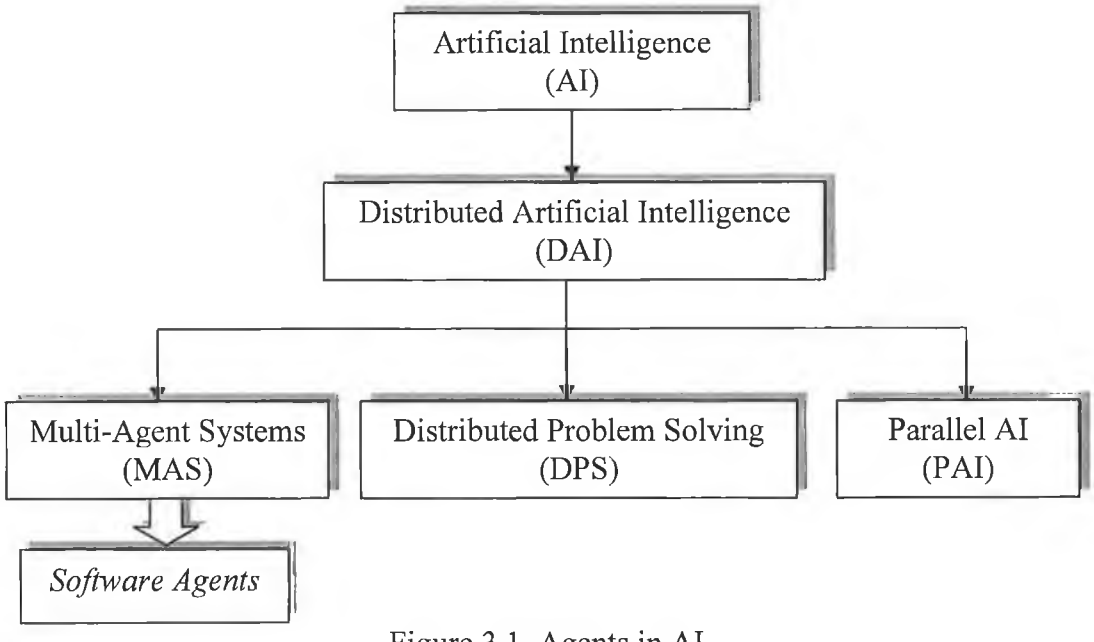


Figure 3.1. Agents in AI

Started in the early eighties, the research in the area of software agents and MAS evolved into what is now "one of the most active areas of research and development activity in computing generally" (Wooldridge and Ciancarini 2001). Nwana splits the research and development work on software agents into two main strands as follows (Nwana 1996):

1. *Strand 1* spans the period 1977 to the current day. This strand "concentrated mainly on deliberative-type agents with symbolic internal models". The emphasis is on macro issues such as "the interaction and communication between agents, the decomposition and distribution of tasks, coordination and cooperation, conflict resolution via negotiation, etc" and agent theories, architectures and languages.
2. *Strand 2* spans the period 1990 to the current day. The emphasis is on the "diversification in the types of agents being investigated" which indicates that software agents are becoming mainstream (Nwana 1996; Bradshow 1997).

The second strand identified by Nwana appeared (at least partly) because "everybody is now calling everything an agent". Indeed, over the last years, there has been an explosion in the use of the term "agent" without a good reason. This was favoured by the lack of a consensus definition for the term agent among AI researchers. Bradshow demonstrates the proliferation of many varieties of "agents" by listing some reasons why a number of programs are called agents e.g. some because they can be scheduled in advance to perform tasks on a remote machine, some because they perform the role of an "intelligent assistant", and some because they manifest characteristics of distributed intelligence (Bradshow 1997).

This chapter aims to describe and define software agents and multi-agent systems by comparing various definitions and taxonomies, identifying common properties proposed by different authors and reviewing agent-oriented methodologies, architectures, standards, languages and environments. Some applications of agents and multi-agent systems are briefly presented at the end of the chapter.

3.2. Software Agents

One of the most dynamic and exciting areas in Computer Science, software agents (characterised by autonomy and flexibility) have the potential to play a crucial role in a large number of application domains including ambient intelligence, computing, electronic business, semantic web, bioinformatics and computational biology (Luck, McBurney et al.

2003). In a recent issue of the IEEE Intelligent Systems publication[1], the editors indicate that software agent technology can be used to "create distributed systems that reason about and dynamically alter their own configurations to maximize their overall dependability" (Greaves, Stavridou-Coleman et al. 2004).

Jennings emphasizes the need for autonomous agents to address the complexity inherent in software systems using the following two arguments (Jennings 2000):

1. *The Adequacy Hypothesis*: "Agent-oriented approaches can significantly enhance our ability to model, design and build complex, distributed software systems" since "the agent-based approach can be viewed as a natural next step in the evolution of a whole range of approaches to software engineering".
2. *The Establishment Hypothesis*: "As well as being suitable for designing and building complex systems, the agent-oriented approach will succeed as a mainstream software engineering paradigm". Furthermore, "agent-based techniques are the ideal computational model for developing software for open, networked systems".

Moreover, Wooldridge and Ciancarini believe that intelligent agents have the potential to form an important new direction in software engineering because agents are the *natural metaphor*[2] to address *distribution of data or control*[3], *legacy systems*[4] and *open systems*[5] (Jennings 2000; Wooldridge and Ciancarini 2001). Having clear potential benefits, software agents are described next in terms of definitions, properties, typologies and architectures.

3.2.1. Definition and Properties

A good sense of what an agent should eventually address is offered by the following typical intelligent agent scenario:

---

[1] http://www.computer.org/intelligent

[2] "Just as many domains can be conceived of consisting of a number of interacting but essentially passive objects, so many others can be conceived as interacting, active, purposeful agents" (Wooldridge and Ciancarini 2001).

[3] In order to effectively address the development of systems composed of different computing nodes that can be geographically and temporally dispersed, "these nodes must be capable of autonomously interacting with each other – they must be agents" (Wooldridge and Ciancarini 2001).

[4] "A natural way of incorporating legacy systems into modern distributed information system is to agentify them" (Wooldridge and Ciancarini 2001).

[5] In order to make open systems work effectively, "the ability to engage in flexible autonomous decision-making is critical" (Wooldridge and Ciancarini 2001).

> *"You are editing a file, when your PDA requests your attention: an email message has arrived, that contains notification about a paper you sent to an important conference, and the PDA correctly predicted that you would want to see it as soon as possible. The paper has been accepted, and without prompting, the PDA begins to look into travel arrangements, by consulting a number of databases and other networked information sources. A short time later, you are presented with a summary of the cheapest and most convenient travel options."* (Wooldridge and Jennings 1995)

A literature review in the area of agents and agent-based systems offers many and diverse definitions for the notion of agency. Nwana notes, "we have as much chance of agreeing on a consensus definition for the word agent as AI researchers have of arriving at one for artificial intelligence itself" (Nwana 1996). An extensive discussion among agent scientists about whether "some particular system is an agent, an intelligent agent or merely a program" generated as many definitions as there are researchers (Anumba, Ugwu et al. 2002) Moreover, "there is now a plethora of different labels for agents ranging from the generic autonomous agents, software agents, and intelligent agents to the more specific interface agents, information agents, mobile agents, and so on" (Luck, McBurney et al. 2003). Bradshow identifies two approaches to the definition of an agent as follows (Bradshow 1997):

1. *Agent as an ascription:* this approach is based on the idea 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".
2. *Agent as a description:* agents are defined by describing the attributes they should exhibit.

The first approach tends to define agents in a general manner offering the opportunity to many systems or components of software to be regarded as agents even though they do not present some minimal properties required by the notion of agency. Foner observes that there is "little justification for most of the commercial offerings that call themselves agents. Most of them tend to excessively anthropomorphise the software, and then conclude that it must be an agent because of that very anthropomorphization, while simultaneously failing to provide any sort of discourse or 'social contract' between the user and the agent. Most are barely autonomous, unless a regularly-scheduled batch job counts" (Foner 1993). Authors in software agent technology generally agree that, even if a complete definition of the term agent is not yet possible, a good description may be given by characterising/describing the space of agent types that would result through the

combination of possible attributes (Wooldridge and Jennings 1995; Wooldridge and Jennings 1995; Franklin and Graesser 1996; Nwana and Wooldridge 1996; Bradshow 1997). Therefore, the second approach, i.e. agent as a description, is considered appropriate for defining software agents in the context of the current thesis.

The most significant definitions of an agent proposed by different researchers and authors are summarised in Table 3.1.

Author(s)	Year	Definition	Reference
Y. Shoham	1993	An agent is an entity whose state is viewed as consisting of mental components such as beliefs, capabilities, choices, and commitments.	(Shoham 1998)
P. Maes	1995	Autonomous agents are computational systems that inhabit some complex, dynamic environment, sense and act autonomously in this environment, and by doing so realize a set of goals or tasks that they are designed for.	(Maes 1995)
B. Hayes-Roth	1995	Intelligent agents continuously perform three functions: perception of dynamic conditions in the environment; action to affect conditions in the environment; and reasoning to interpret perceptions, solve problems, draw inferences, and determine actions.	(Hayes-Roth 1995)
S. Russell P. Norvig	1996	An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors.	(Russell and Norvig 2003)
H.S. Nwana	1996	When 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.	(Nwana 1996)
S. Franklin A. Graesser	1996	An autonomous agent is a system situated within and part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future.	(Franklin and Graesser 1996)
N.R. Jennings M. Wooldridge K. Sycara	1998	An agent is a computer system that is situated in some environment, and that is capable of flexible autonomous action in this environment in order to meet its design objectives.	(Jennings, Sycara et al. 1998; Jennings and Wooldridge 1998; Wooldridge 1999)
FIPA[6] (standard)	2000	An agent is an encapsulated software entity with its own state, behavior, thread of control, and an ability to interact and communicate with other entities – including people, other agents, and legacy systems.	(Poslad, Buckle et al. 2000)
OMG[7]	2000	An agent is a computer program that acts autonomously on behalf of a person or organization.	(OMG 2000)
AgentLink[8] Roadmap	2003	An agent is a computer system capable of flexible autonomous action in a dynamic, unpredictable and open environment.	(Luck, McBurney et al. 2003)

Table 3.1. Agent definitions

[6] FIPA - Foundation for Intelligent Physical Agents (http://www.fipa.org) is a non-profit standard organization established in 1996, which promotes the creation of specifications of generic agent technologies.

[7] OMG - Object Management Group (http://www.omg.org) is a non-profit international corporation focusing on computer industry specifications for interoperable enterprise applications.

[8] AgentLink (http://www.agentlink.org) is a coordinating organisation for research and development activities in the area of agent-based computer systems on the behalf of the European Commission.

Although there is no universally accepted definition for an agent, researchers and scientists generally agree that an agent is characterised by the following (Nwana 1996; Wooldridge 1999; Jennings 2000; Luck, McBurney et al. 2003):

- An agent acts on behalf of its user (Maes 1995; Nwana 1996; Jennings and Wooldridge 1998).
- An agent is situated in an environment and is able to perceive that environment (Maes 1995; Franklin and Graesser 1996; Jennings and Wooldridge 1998; Poslad, Buckle et al. 2000).
- An agent has a set of objectives and takes actions so as to accomplish these objectives (Maes 1995; Franklin and Graesser 1996; Nwana 1996; Jennings and Wooldridge 1998).
- An agent is autonomous i.e. an agent can take decisions without the intervention of humans or other systems (based on the individual state and goals an agent has) (Maes 1995; Franklin and Graesser 1996; Nwana 1996; Jennings and Wooldridge 1998; Wooldridge 1999; Jennings 2000; Poslad, Buckle et al. 2000).

Agents receive inputs about the state of the environment they're situated in through sensors and they can perform actions through effectors (*situatedness*) (Jennings 2000). Having the potential to affect its environment, each action has a set of associated pre-conditions that specify the possible situations when it can be performed. Wooldridge demonstrates this with an action 'lift table' which will succeed only if the weight of the table is sufficiently small that the agent can lift it (Wooldridge 1999). Figure 3.2 shows the interaction between an agent and its environment, which is usually an ongoing and non-terminating one.

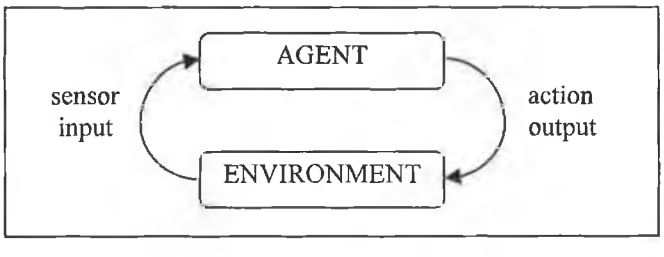


Figure 3.2. An agent in its environment (Wooldridge 1999)

Table 3.2 presents different types of environments that an agent can occupy based on various environment properties (Wooldridge 1999; Russell and Norvig 2003). The everyday physical world can be regarded as an inaccessible, non-deterministic, non-episodic, highly dynamic and continuous environment, which is in fact the most complex general class of environments (Russell and Norvig 2003). In most environments, an agent

will not have complete control but will enjoy partial control over its environment. Therefore, the same action may fail to have the desired effect even though performed in apparently identical circumstances (Wooldridge 1999).

Classification	**Explanation**	**Greater problem for the agent designer**
Accessible vs. Inaccessible	Agents can obtain complete, accurate, up-to-date information about the environment's state in an accessible environment. Most complex environments are inaccessible.	Inaccessible Environment
Deterministic vs. Non-deterministic	Any action of the agent has a single guaranteed effect in a deterministic environment as opposed to some uncertainty about the resulting state after an action is performed in a non-deterministic environment.	Non-deterministic Environment
Episodic vs. Non-episodic	The performance of an agent depends on a number of discrete episodes in an episodic environment. There is no link between the performance of an agent in different scenarios.	Non-episodic Environment
Static vs. Dynamic	A static environment can be changed only by the performance of the actions of the agent while a dynamic environment has other processes operating in it, which are not under the control of the agent.	Dynamic Environment
Discrete vs. Continuous	There are a fixed, finite number of actions and percepts in a discrete environment.	Continuous Environment

Table 3.2. Classification of environment properties (Russell and Norvig 2003)

Wooldridge and Jennings (Wooldridge and Jennings 1995) identify a weak notion of agency by which an agent is characterised by autonomy, social ability (cooperation), reactivity and pro-activeness. Additionally, a stronger notion of agency (widespread in AI) exists by which an agent enjoys all the properties associated with the weak notion and also uses mental components such as belief, desire, intention, knowledge and obligation.

The main properties that should characterize a software agent can be summarised as follows (Maes 1995; Franklin and Graesser 1996; Nwana 1996; Wooldridge 1999):

- *Autonomy:* An agent can operate on its own without the intervention of humans or other systems.
- *Reactivity:* An agent is situated in an environment and is able to perceive this environment and to respond to changes that occur in it.
- *Pro-activeness:* The ability to take the initiative in order to pursue its individual goals (goal-directed behaviour).

- *Cooperation:* The capability of interacting with other agents and possibly humans via an agent-communication language. Involves the ability of an agent to dynamically negotiate and coordinate.
- *Learning:* The ability to learn while acting and reacting in its environment. Learning can increase performance of an agent over time.
- *Mobility:* The ability to move around a network (even from one platform to another) in a self-directed way.

Furthermore, some authors identified more properties associated with the notion of agency such as (Franklin and Graesser 1996; Nwana 1996; Wooldridge and Jennings 1995; Bradshow 1997; Luck, McBurney et al. 2003):

- *Temporal continuity:* The actions of an agent are performed through a continuous running process (over long periods of time).
- *Personality:* A believable character and emotional state.
- *Veracity:* An agent should not knowingly communicate false information.
- *Benevolence:* Agents should not have conflicting goals and every agent should always try to accomplish its objective.
- *Rationality:* An agent should act so as to achieve its goals and not to prevent its goals from being achieved.

Autonomy, reactivity, pro-activeness and cooperation (or social ability) are the properties used by Wooldridge and Ciancarini (Wooldridge and Ciancarini 2001) in their definition of the term agent (these properties are not optional but actually *define* an agent). Jennings et al include three key concepts in their definition of an agent: situatedness, autonomy and flexibility. The latter concept is defined using three properties as follows: reactivity (an agent should be responsive), pro-activeness and social ability (Jennings, Sycara et al. 1998). Nwana considers that a truly intelligent agent (the ideal agent) should equally be characterised by three primary attributes i.e. autonomy, cooperation and learning while any system that does not exhibit these three properties (more or less emphasized) should not be considered an agent at all (Nwana 1996).

However, these agent properties are more challenging than they seem. While pursuing their goal, agents should cancel actions when it is clear that those actions will not work or when the goal of the action is not longer valid. In such a situation, reactivity should be demonstrated: the agent should react to the events that occur in its dynamic environment. While pro-activeness (in a system that exhibits goal-directed behaviour) and reactivity (in a purely reactive system) can be easily implemented independently, integrating goal-directed

and reactive behaviour within a system is a difficult task. This problem of achieving an effective balance between pro-activeness and reactivity represents one of the key problems of the agent designer and is basically still open to discussion (Wooldridge and Ciancarini 2001).

Table 3.3 summarizes the main properties associated with the notion of agency and provides an explanation for each.

Property	**Other name(s)**	**Description & comments**	**Reference(s)**
Autonomy		An agent can operate on its own without the intervention of humans or other systems. ➢ *Generally accepted.*	(Maes 1995; Franklin and Graesser 1996; Nwana 1996; Jennings and Wooldridge 1998; Wooldridge 1999)
Reactivity	Situatedness *or* Sensing and acting	An agent perceives its environment: it receives input from the environment and is able to change the environment by performing some actions.	(Franklin and Graesser 1996; Nwana 1996; Jennings and Wooldridge 1998; Wooldridge 1999; Jennings 2000)
Pro-activeness	Goal-directed behaviour	An agent has the ability to take the initiative in order to accomplish its design objectives. ➢ *Considered by Nwana a key element of autonomy*	(Franklin and Graesser 1996; Nwana 1996; Jennings and Wooldridge 1998; Wooldridge 1999)
Cooperation	Communicative *or* Social Ability	An agent is capable of interacting with other agents and/or humans in order to accomplish its design objectives. ➢ *Viewed by most researchers as a crucial attribute of an intelligent agent.*	(Franklin and Graesser 1996; Nwana 1996; Jennings and Wooldridge 1998; Wooldridge 1999)
Flexibility		Defined by Wooldridge and Jennings using three other properties i.e. reactivity, pro-activeness and cooperation ➢ *Flexibility is not a new property but instead incorporates three properties already defined.*	(Jennings, Sycara et al. 1998; Jennings and Wooldridge 1998; Wooldridge 1999)
Learning	Adaptivity	An agent can learn and improve with experience. ➢ *Considered by Nwana a key attribute of an intelligent agent.*	(Franklin and Graesser 1996; Nwana 1996; Bradshow 1997)
Mobility		An agent has the ability to move from one machine to another in a network.	(Franklin and Graesser 1996; Nwana 1996; Bradshow 1997; Luck, McBurney et al. 2003)
Temporal continuity		An agent persists over long periods of time.	(Franklin and Graesser 1996; Nwana 1996; Bradshow 1997)
Personality	Character	An agent demonstrates a "believable" personality and emotional state.	(Franklin and Graesser 1996; Nwana 1996; Bradshow 1997)

Table 3.3. Properties of an agent

Furthermore, research shows that the various properties of an agent are of differing importance for different domains. Therefore, learning can be considered very important for

some applications while others may consider it really detrimental (Wooldridge 1999). Table 3.4 presents the definition of an agent based on a list of the properties the agent should have proposed by different authors.

Author	**Reference**	**Properties used in the definition of an *agent***
H.S. Nwana	(Nwana 1996)	Autonomy (which includes pro-activeness) Cooperation Learning
S. Franklin A. Graesser	(Franklin and Graesser 1996)	Autonomy Reactivity Pro-activeness Temporal Continuity
N.R. Jennings M. Wooldridge	(Jennings, Sycara et al. 1998; Wooldridge 1999)	Autonomy Reactivity Pro-activeness } Flexibility Social Ability

Table 3.4. Various definitions of an agent using a list of properties

The current thesis identifies an agent as a *software system situated in an environment that autonomously acts on behalf of its user and is able to cooperate with other agents and/or humans in order to accomplish its objectives*. If necessary for a particular application domain, an agent should also be characterised by mobility. Autonomy is unquestionably the most important property of an agent without which the notion of agency would not exist. Furthermore, cooperation among different software agents may be very useful in achieving the objectives an agent has.

3.2.2. Agent Typologies

A review of the various typologies of agents proposed by different researchers can potentially aid the quest of understanding and describing the agent theory. The most straightforward classification of an agent would be along one of their properties such as (Nwana 1996):

- *Mobility:* static or mobile agents.
- *Reactivity:* deliberative or reactive agents.

A well-known agent taxonomy proposed in the agent research community is Gilbert's scope of intelligent agents (see Figure 3.3). Intelligent agents are described using the following three dimensions (Bradshow 1997):

- *Agency* refers to "the degree of autonomy and authority vested in the agent, and can be measured at least qualitatively by the nature of the interaction between the agent and other entities in the system. At minimum, an agent must run asynchronously.

The degree of agency is enhanced if an agent represents a user in some." (Gilbert et al. 1995 as cited by (Bradshow 1997)).

- *Intelligence* refers to the degree of reasoning and learned behaviour. Furthermore, intelligent agents should learn and adapt to their environment in terms of the user's objectives and the resources available.
- *Mobility* refers to the degree to which the agents travel through the network.

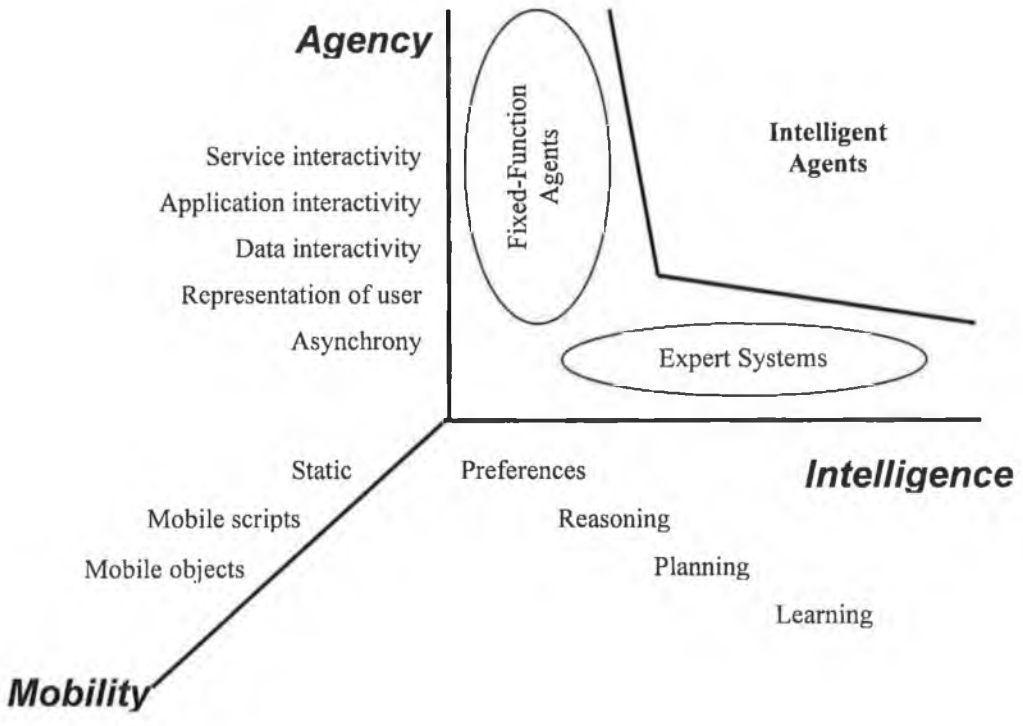


Figure 3.3. Scope of intelligent agents (adapted from Gilbert et al. by (Bradshow 1997))

Another well acknowledged agent taxonomy is Nwana's primary attribute dimension typoplogy (Nwana 1996). Nwana considers autonomy, cooperation and learning to be the minimal characteristics an agent should exhibit. These three properties are used to classify agents in four categories as follows (see Figure 3.4) (Nwana 1996):

- *Collaborative agents:* there is more emphasis on cooperation and autonomy than on learning.
- *Collaborative learning agents:* there is more emphasis on cooperation and learning than on autonomy.
- *Interface agents:* there is more emphasis on autonomy and learning than on cooperation.
- *Smart agents:* these agents equally implement autonomy, cooperation and learning.

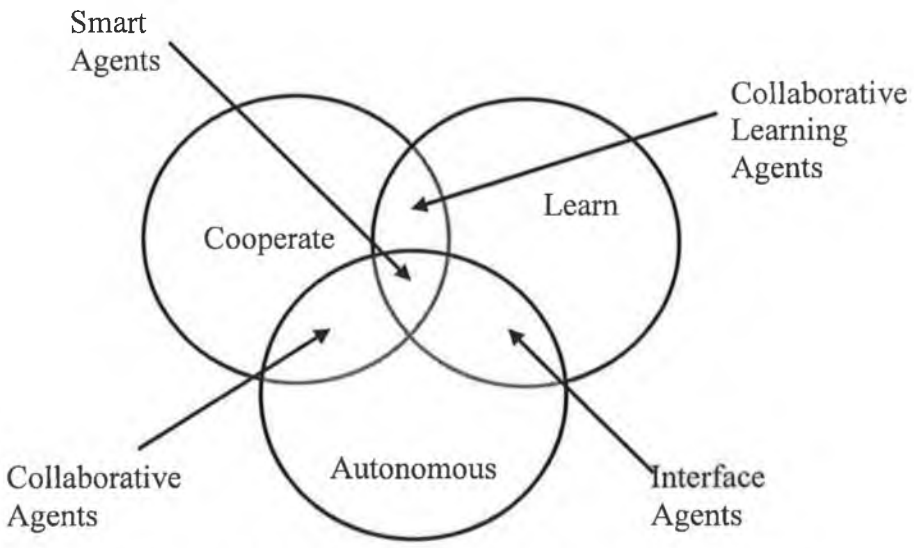


Figure 3.4. Nwana's agent typology (Nwana 1996)

As already indicated, mobility can also be used to classify agents in *static* or *mobile* while the presence of a symbolic reasoning model results in *deliberative* or *reactive* agents. Nwana (Nwana 1996) combines these types of agents with the ones already identified based on the ideal and primary attributes of an agent to produce other categories of agents such as static deliberative collaborative agents, mobile reactive collaborative agents, static deliberative interface agents, mobile reactive interface agents, etc. Another classification proposed by Nwana uses the roles of agents and is exemplified with *information* or *internet* agents. This category of agents manages large databases in wide area networks like the internet. The last category of agents identified by Nwana consists of *hybrid* agents, which combine two or more agent philosophies. Furthermore, Nwana uses these agent typologies to identify only seven types of agents as shown in table 3.5 (Nwana 1996).

No	Type of agent	Description	Key characteristics
1	*Collaborative agents*	"Able to act rationally and autonomously in open and time-constrained multi-agent environments".	Autonomy Social ability Responsiveness Pro-activeness
2	*Interface agents*	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.	Autonomy Learning Cooperation
3	*Mobile agents*	Autonomous software programs capable of roaming wide area networks (such as WWW) and cooperation while performing duties on behalf of its user.	Mobility Autonomy Cooperation
4	*Information/Internet agents*	Designed to manage, manipulate or collate the vast amount of information available from many distributed sources (information explosion). 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".	Mobility Cooperation Learning

5	*Reactive agents*	Act/respond to the current state of their environment based on a stimulus-response scheme. 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.	Autonomy Reactivity
6	*Hybrid agents*	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).	-
7	*Smart agents*	Are the ideal agents being equally characterised by autonomy, cooperation and learning.	Autonomy Cooperation Learning

Table 3.5. A classification of software agents proposed by Nwana (Nwana 1996)

Heterogeneous agent systems are obtained by combining agents from two or more of these categories. Unlike hybrid agent architectures, this agent category refers to an integrated set-up of at least two or more types of agents (including hybrid agents). Agent-based software engineering facilitates the interoperation of miscellaneous software agents. An agent communication language is necessary for the communication process among different agents (Nwana 1996). This category of agent systems is generally referred to (by most researchers) as multi-agent systems and is discussed in more detail in the next section of this chapter.

Franklin and Graesser proposed the taxonomy of autonomous agents presented in Figure 3.5 (Franklin and Graesser 1996).

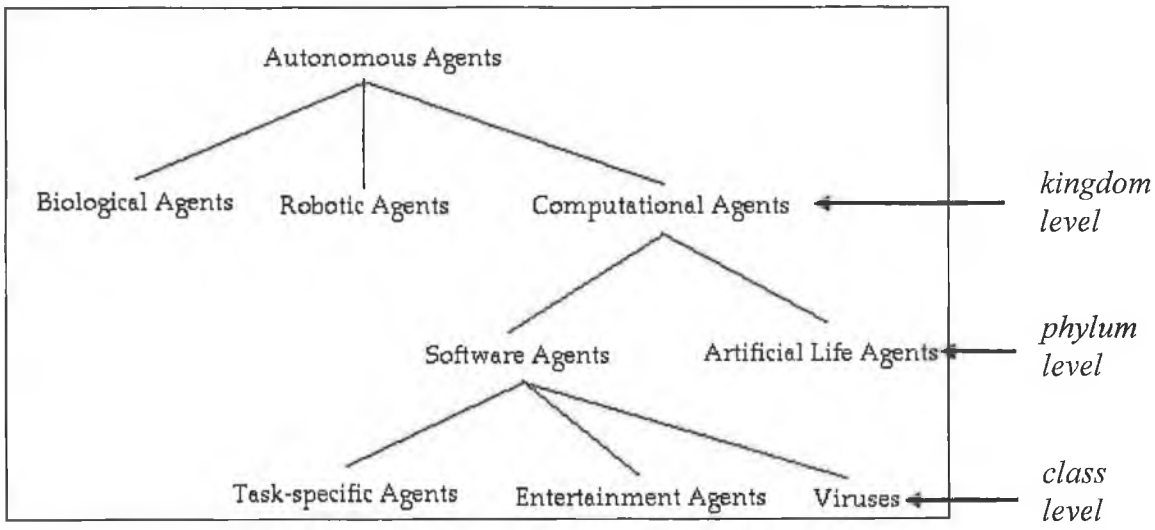


Figure 3.5. The taxonomy of agents proposed by Franklin and Graesser (after (Nwana 1996).)

Franklin and Graesser's agent taxonomy includes biological, robotic and computational agents at the kingdom level, software agents and artificial life agents at the phylum level

and task-specific agents, entertainment agents and computer viruses at the class level. A further taxonomy can be performed using schemes such as classification via the agent's control structures (e.g. regulation, planning and adaptive), via environments (e.g. database, file system, network, internet), via languages (in which the agent is written) and via applications. These subclassification schemes provide a collection of features for an agent and therefore a possible category of classification (Franklin and Graesser 1996).

From an architectural point of view (see section 3.2.3), Wooldridge identifies four classes of agents as follows (Jennings, Sycara et al. 1998; Wooldridge 1999; Devedzic 2001):

- Logic-based agents
- Reactive agents
- BDI (Belief-Desire-Intention) agents
- Layered architectures

Logic based agents are identified by Wooldridge as those agents that use logical formulae as the symbolic representations and logical deduction or theorem proving as the syntactic manipulation. In this approach to building agents, decision making is realized via logical deduction. The program of an agent is encoded as logical theory and the selection of an action is reduced to a problem of proof (Wooldridge 1999). The other types of agents identified by Wooldridge are further analysed in the 3.2.3 section of this chapter.

3.2.3. Agent Architectures

Agent architectures address the issues of designing and creating computer-based systems that satisfy the agent properties (proposed by agent theorists) e.g. autonomy, reactivity, pro-activeness, social ability (Wooldridge 1998). "An 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" (Wooldridge 1999). Wooldridge and Jennings indicate that agent architectures can be viewed as software engineering models of agents (Wooldridge and Jennings 1995). They identify the following classes of agent architectures (Wooldridge and Jennings 1995; Wooldridge and Jennings 1995; Wooldridge 1999):

1. Deliberative (or symbolic) architectures
2. Reactive (or behavioural or situate) architectures
3. Hybrid architectures

Deliberative architectures adopt the traditional AI (called symbolic AI) approach to designing intelligent systems by viewing them as a type of knowledge-based system. Wooldridge defines a deliberative agent as one that "contains an explicitly represented, symbolic model of the world" and "makes decisions (for example about what actions to perform) via symbolic reasoning" (Wooldridge 1999). The agent-based system that has to be designed receives a symbolic representation of its environment and its desired behaviour, which can be syntactically manipulated. Figure 3.6 shows how deliberative agents adopt the sense-plan-act problem-solving paradigm of classical AI planning systems (Helin 2003).

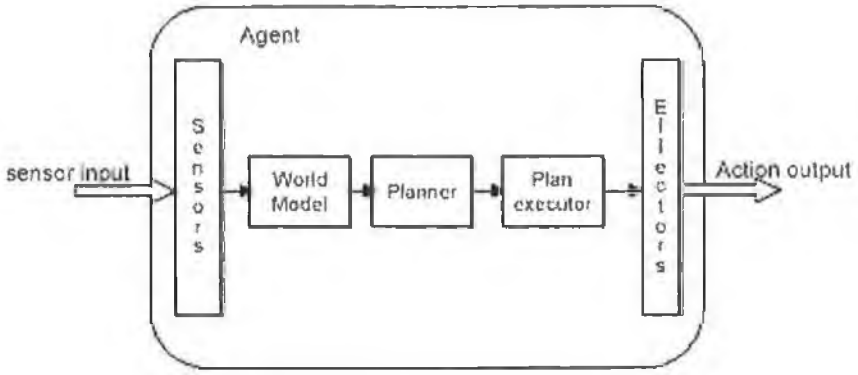


Figure 3.6. The basic architecture of a deliberative agent (Helin 2003)

The disadvantages associated with deliberative architectures can be summarised as follows (Wooldridge 1999; Helin 2003):

- *The transduction problem*

  It is time consuming to translate the information into its symbolic representation. The assumption of calculative rationality (i.e. "the assumption that the world will not change in any significant way while the agent is deciding what to do, and that an action which is rational when decision making begins will be rational when it concludes") (Wooldridge 1999) might result in an ineffective operation of agents in time-constrained environments.

- *The representation/reasoning problem*

  This problem refers to representing and reasoning about complex, dynamic, possibly physical environments (so as to achieve useful results).

Much of the research and development work on deliberative agents has focused on the agent-oriented programming paradigm (Wooldridge 1998). The state of an agent is characterised in terms of its mental attitudes of belief, desire and intention (Rao and Georgeff 1995). Agent-oriented programming uses these intentional notions to directly

program agents. Shoham developed an experimental language called AGENT0 (Shoham 1998) in order to demonstrate the agent-oriented programming paradigm. Other examples of logical approaches to agent programming include the ConGolog (Giacomo, Lespérance et al. 2000) and the Concurrent METATEM (Fisher 1994) programming languages. These agent languages are all described later in this chapter.

Reactive architectures are an alternative to the symbolic AI paradigm. They involve developing and combining individual behaviours of reactive agents situated in some environment (Wooldridge 1999). Reactive agents have a very simple representation of the world but provide tight coupling of perception and action. The behaviour-based paradigm informs the reactive approach to building agents. Each individual behaviour continually maps perceptual input to action output. Figure 3.7 presents the basic architecture of a reactive agent (Helin 2003).

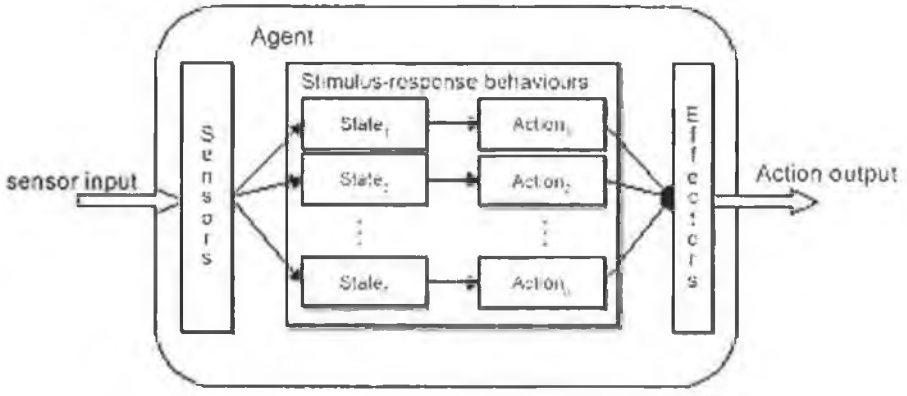


Figure 3.7. The basic architecture of a reactive agent (Helin 2003)

In the reactive approach, intelligent behaviour emerges from the interaction of various simpler behaviours as well as from the interaction between an agent and its environment. The main disadvantage of this architecture relates to the fact that agents do not employ models of their environment. This means that they need a great deal of local information to determine an acceptable action. Decision making is realised in the agent's local environment without necessarily taking into account non-local information (Wooldridge 1999). A well-known example of reactive agent architecture is the subsumption architecture developed by Brooks (Brooks 1986).

Hybrid architectures combine the deliberative and reactive approaches (see Figure 3.8). An agent consists of several subsystems that manifest characteristics of both deliberative and reactive approaches as follows (Helin 2003):

- *Deliberative component*: subsystems develop plans and make decisions using symbolic reasoning.
- *Reactive component*: subsystems are able to react quickly to events without complex reasoning.

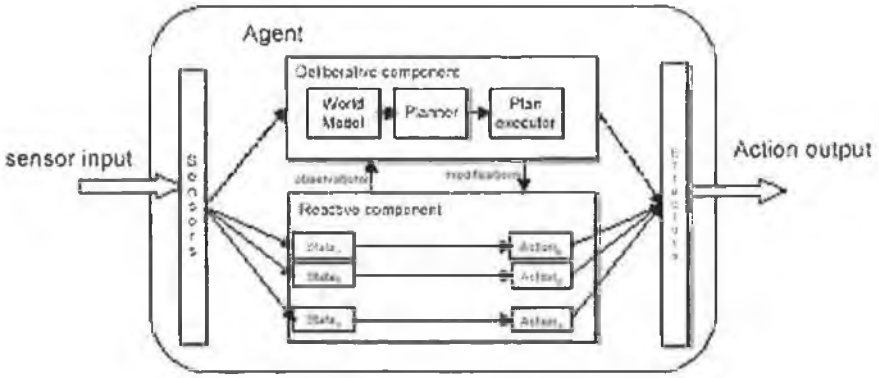


Figure 3.8. The basic architecture of a hybrid agent (Helin 2003)

A popular approach to the design of hybrid agents is the use of *layered architectures* (Wooldridge 1998). The various subsystems of the architecture are arranged into a hierarchy of interacting layers each of which is reasoning about the environment at different levels of abstraction. Two types of information and control flow have been identified within layered architectures i.e. horizontal and vertical. In horizontally layered architectures, each layer is directly connected to the sensory input and action output (see Figure 3.9).

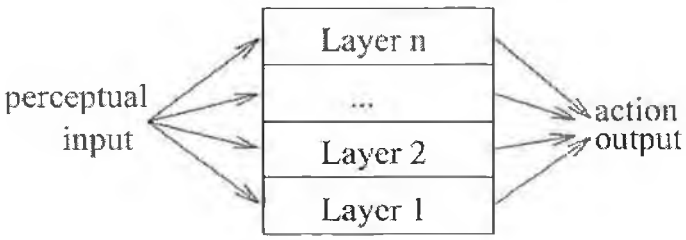


Figure 3.9. Control flow in horizontally layered agent architecture (Wooldridge 1999)

Acting like an agent, each layer produces action suggestions. However, this means that the layers are competing with one another creating the "danger that the overall behaviour of the agent will not be coherent" (Wooldridge 1999). The main advantage of this approach is the inherent conceptual simplicity allowing an agent to exhibit many different types of behaviour. In vertically layered architectures, at most one layer is connected to the sensory input and action output (see Figure 3.10).

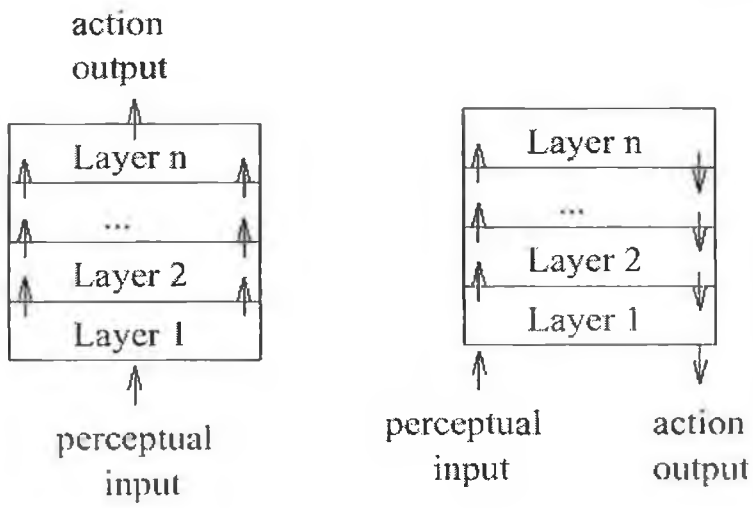


Figure 3.10. Control flow in vertically layered agent architecture (one pass control and two pass control) (Wooldridge 1999)

These architectures can be further classified into one pass architectures (information and control flows sequentially through each layer) and two pass architectures (information flows up through each layer and then control sequentially flows down). The lack of flexibility is the main disadvantage of vertical layering: control must flow through each different layer before a decision is made (Wooldridge 1999). Some examples of layered architectures include the TouringMachines (Ferguson 1992) and INTERRAP (Muller and Pischel 1993) architectures.

Another well-known agent architecture is the Procedural Reasoning System (PRS) (Ingrand, Georgeff et al. 1992) based on the *belief-desire-intention* (BDI) model (Rao and Georgeff 1995). Inspired by the philosophical tradition of understanding practical reasoning, BDI architectures have become very popular over the last years (Georgeff, Pell et al. 1999; Wooldridge 1999). The BDI architecture represents an agent in terms of its beliefs, desires (or goals) and intentions. The basic components of a BDI agent are data structures (that represent beliefs, desires and intentions) and functions for representing and reasoning about them. As shown in Figure 3.11, there are seven key components of a generic BDI architecture as follows (Wooldridge 1999):

- The agent's *beliefs* correspond to the information the agent has about the environment it occupies.
- The *belief revision function (brf)* determines new beliefs based on a perceptual input and the agent's current beliefs.
- The agent's *desires* correspond to the available actions (intuitively, allocated tasks).

- The agent's *intentions* correspond to those desires to which the agent has committed to achieving (the agent's current focus).
- The *option generation* function determines the agent's desires based on the agent's current beliefs and intentions.
- The *filter* function determines the agent's intentions based on the agent's current beliefs, desires and intentions.
- The *action* selection function determines the action to be performed based on the agent's current intentions.

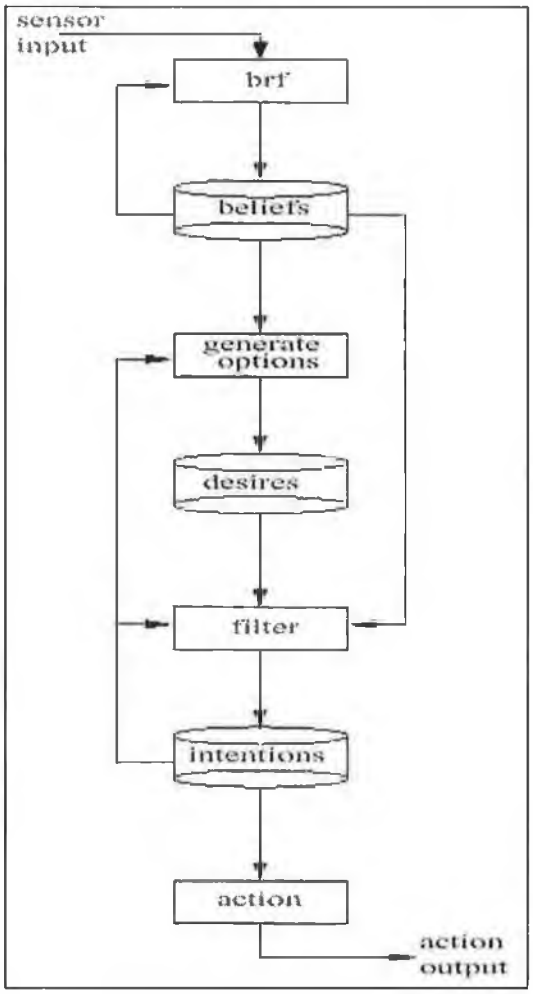


Figure 3.11. A generic architecture of a BDI agent (Wooldridge 1999)

The BDI architecture presents some attractive benefits such as intuitiveness and clear functional decomposition (Wooldridge 1999). Another positive aspect is that many researchers focused on the formalisation of the BDI model. Rao and Georgeff developed a family of BDI logics for the formal semantics of BDI architectures, which are mainly based on possible relationships between the three mental components of BDI agents or on

proof methods for restricted forms of the logics (Rao and Georgeff 1995). An important problem in BDI architectures is that of "striking a balance between being committed to and overcommitted to one's intentions: the deliberation process must be finely tuned to its environment, ensuring that in more dynamic, highly unpredictable domains, it reconsiders its intentions relatively frequently – in more static environments, less frequent reconsideration is necessary" (Wooldridge 1999). Also, the BDI model should consider systems that must learn and adapt their behaviour, which are becoming more and more important (Georgeff, Pell et al. 1999).

3.3. Multi-Agent Systems

As already indicated, systems composed of multiple agents are studied under the banners of Multi-Agent Systems (MAS) and Distributed Problem Solving (DPS), the two main fields of DAI (Green, Hurst et al. 1997; Sen 1997; Jennings, Sycara et al. 1998; Oliveira, Fischer et al. 1999). A DPS system incorporates interaction strategies in order to solve a particular given problem through cooperation (by dividing and sharing knowledge about the problem) among different modules. On the other hand, MAS researchers study the behaviour of a group of autonomous agents, which are working together towards a common goal (Jennings 2000; Wooldridge and Ciancarini 2001; Luck, McBurney et al. 2003). Another term often encountered in the literature is that of an *agent-based system* which can be defined as "one in which the key abstraction used is that of an agent" (Wooldridge and Ciancarini 2001). This means that a single agent can form an agent-based system (Jennings, Sycara et al. 1998) whereas the key characteristic of MAS is that of interoperation among two or more agents within the same system.

3.3.1. Potential Benefits

Representing a great potential of agent-based systems, MAS are ideal for solving complex problems with multiple problem solving methods, multiple perspectives and/or multiple problem solving entities (Jennings, Sycara et al. 1998). The potential benefits of employing MAS for developing complex software applications can be summarised as follows (Bradshow 1997; Green, Hurst et al. 1997; Gasser 1998; Jennings, Sycara et al. 1998; Martin, Plaza et al. 1998; Sycara 1998; Park and Sugumaran 2005):

- Ability to solve large and complex problems as opposed to a single centralised agent that might fail the same task (because of resource limitations for example).

- Interconnection and interoperation of multiple existing legacy systems (e.g. expert systems, decision support systems).
- Ability to provide solutions to efficiently manage domains in which the information resources are spatially distributed (e.g. sensor networks, seismic monitoring, Internet information gathering).
- Ability to handle domains in which the expertise is distributed (e.g. concurrent engineering, health care, manufacturing).

Furthermore, the MAS solution offers fundamental prospective features including the following (Sycara 1998; Wooldridge 1999; Jennings 2000):

- *Computational efficiency* – MAS exploit concurrent computation
- *Reliability* – if a component fails, an agent with redundant capabilities is dynamically identified.
- *Extensibility* – any number of agents with different capabilities can work in the same problem.
- *Robustness* – agents exchange suitable information.
- *Maintainability* – MAS can be easily maintained due to modularity.
- *Responsiveness* – anomalies can be managed locally without propagating them to the whole system.
- *Flexibility* – adaptivity allows agents with different abilities to interoperate in order to solve a problem.
- *Reuse* – an agent can be reused to solve another problem within a different system.

Addressing complex system development in distributed environments (Park and Sugumaran 2005), the MAS approach to building computational systems promotes conceptual clarity and simplicity of design (Martin, Plaza et al. 1998; Wooldridge 1999; Jennings 2000).

3.3.2. Definition

A MAS is a "loosely coupled network of problem solvers that work together to solve problems that are beyond the individual capabilities or knowledge of each problem solver" (Jennings, Sycara et al. 1998). The problem solvers from this definition are autonomous and possibly heterogeneous agents able to interact with each other in order to reach an overall goal (Green, Hurst et al. 1997; Jennings, Sycara et al. 1998). Moreover, each agent within the MAS has a limited set of capabilities or incomplete information to solve the problem. Additionally, the MAS approach implies that there is no global system control,

data is decentralized and computation is asynchronous (Jennings, Sycara et al. 1998; Sycara 1998; Oliveira, Fischer et al. 1999; Lazansky, Stepankova et al. 2001).

Clearly, the interoperation among autonomous agents of a MAS is essential for the successful location of a solution to a given problem. Agent-oriented interactions span from simple information interchanges to planning of interdependent activities for which cooperation, coordination and negotiation are fundamental. Jennings notes that these agent interactions differ from those that occur in other computational models from two perspectives as follows (Jennings 2000):

- An agent knows which goals should be followed and, therefore, agent-oriented interactions are taking place at the knowledge level.
- Agents are flexible entities in an environment over which they have only partial control and, therefore, they have to make run-time decisions about their interactions that were not foreseen at design time.

Jennings also identifies the organisational relationships inherent in an agent-based system because agents act towards a goal on behalf of individuals/companies or as part of a wider problem solving initiative. The organisational context needs to be explicitly represented as it defines the nature of the relationships between agents and influences their behaviour (Jennings 2000). All these concepts are represented by Jennings using a canonical view of an agent based system (see Figure 3.12).

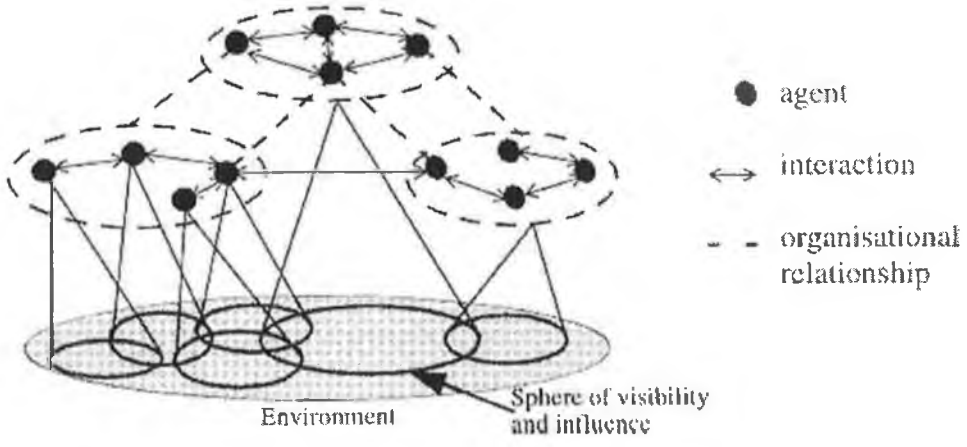


Figure 3.12. Canonical view of an agent-based system (Jennings 2000)

Depending on the degree of cooperation demonstrated by individual agents, two types of MAS have been identified by researchers as follows (Green, Hurst et al. 1997):

1. *Cooperative Multi-Agent Systems (CMAS)*

The general performance of the system is important and, therefore, all agents in the system act cooperatively in an appropriate manner. The designer of such a system is not concerned with the performance of individual agents.

2. *Self-Interested Multi-Agent Systems (SMAS)*

   Interested only on the benefit derived from individual agents, independent designers implement individually motivated agents. Such agents are considered self-interested, competitive or non-cooperative.

The situation of total cooperation known as the *benevolent agent assumption* is generally accepted in DAI research but agents may have conflicting goals resulting in this cooperative to antagonistic spectrum in a MAS (Green, Hurst et al. 1997). Cooperation is the primary element in a CMAS while negotiation is seen as the method for coordination and conflict resolutions in SMAS (Jennings, Sycara et al. 1998).

However, some inherent problematic issues to the design and implementation of MAS have been identified by researchers. These problems, which have intertwined solutions, can be summarised as follows (Gasser 1998; Jennings, Sycara et al. 1998; Oliveira, Fischer et al. 1999):

- Formulation, description, decomposition and allocation of the problem and synthesis of the results among a group of intelligent agents.
- Communication and interaction among agents: what communication languages or protocols to use in order to enable a meaningful agent interaction, and what and when to communicate?
- Coordination among agents: how to enable individual agents to reason about the actions, plans, strategies and beliefs of other agents and about their coordinated process?
- Identification and reconciliation of disparate viewpoints and conflicts among agents trying to coordinate their actions.
- Balance of local computation and communication: how to avoid computational overload by the means of load balancing strategies?
- Implementation of a MAS: how to engineer and construct practical MAS and what are the technology platforms and development methodologies to support MAS design and implementation?
- Verification and correction of MAS applications using formal and practical approaches.

MAS research must address the following key elements in order to tackle the above mentioned problems (Green, Hurst et al. 1997):

1) *Coordination:* agents have to coordinate their activities in order to determine the organisational structure in a group of agents and to allocate tasks and resources.
2) *Negotiation:* agents must negotiate if a conflict occurs.
3) *Communication:* agents must be able to communicate with each other in order to exchange information and knowledge.

Indeed, "some of the key research issues related to problem-solving activities of agents in a MAS are in the areas of coordination, negotiation and communication" (Park and Sugumaran 2005). Additionally, recent research started to address the problem of trust in MAS (Wong and Sycara 1999; Ramchurn, Huynh et al. 2004; Jiang, Xia et al. 2005). Even if inter-agent coordination, communication and negotiation models are successful, agents in open-network MAS may face some security and trust issues (Wong and Sycara 1999; Ramchurn, Huynh et al. 2004). The remainder of this section offers a more detailed discussion on coordination, negotiation, communication, ontologies and trust in MAS.

3.3.3. Coordination in Multi-Agent Systems

MAS research benefits from the results of various other areas that study coordination e.g. organisation theory, political science, social psychology, anthropology, law and sociology. As regards to the MAS field, interacting agents have to efficiently coordinate their activities towards a common goal.

Coordination has been defined as "a process in which agents engage in order to ensure a community of individual agents acts in a coherent manner" (Nwana 1996). Agents may have to *communicate* in order to achieve the necessary coordination (Nwana 1996). However, an agent can coordinate its activities with those of another agent unaware of its presence meaning that coordination does not imply reciprocation (Durfee 2001).

Coordination is necessary in a MAS because agents have different and limited capabilities and expertise (Nwana, Lee et al. 1996; Green, Hurst et al. 1997). Furthermore, interdependent activities require coordination (the action of one agent might depend on the completion of a task for which another agent is responsible). Enabling efficiency, coordination prevents anarchy or chaos (such a situation is possible because each agent has a partial view over its environment and therefore, its actions might interfere with rather than support the actions of another agent) during conflicts (Nwana, Lee et al. 1996; Green, Hurst et al. 1997).

The foremost techniques to address coordination in MAS proposed by different researchers include organisational structuring, Contract Net Protocol (CNP), multi-agent planning, social laws and computational market-based mechanisms (Nwana, Lee et al. 1996; Green, Hurst et al. 1997; Oliveira, Fischer et al. 1999).

Organisational structuring is the simplest coordination technique and exploits the a priori organisational structure: the system of agents is provided with an agent which has a wider perspective of the system. Hierarchical structuring yields the classic master-slave or client-server coordination technique. Many researchers adopted the blackboard strategy to implement this technique: scheduled by a master agent, agents can read/write to/from the blackboard. Systems that exploit this architecture include the Designer Fabricator Interpreter (DFI) system proposed by Werkman (Werkman 1990), the Sharp Multi-Agent Kernel (SMAK), the Distributed Vehicle Monitoring Testbed (DVMT) system and the DRESUN testbed for research on distributed situation assessment (DSA) which explores the implications of agents with more sophisticated representations and control capabilities than those in DVMT (Carver, Lesser et al. 1993) and the free-market agent architecture MAGMA (Tsvetovatyy, Gini et al. 1997). Although useful where the master-slave relationships are inherent to the modelled MAS, this technique is impractical in many realistic applications (because it presumes that at least one agent has a global view over the entire environment).

The *Contract Net Protocol* is a high-level coordination strategy proposed by Smith and Davis (as cited in (Nwana 1996)) and used in many applications. This approach assumes a decentralised market structure in which agents can have two roles: a manager or a contractor. While monitoring the problem's overall solution, the manager breaks the problem in sub-problems and assigns them to contractors which in turn solve them or may recursively become managers and further decompose the sub-problem. The best applications of this technique include well-defined hierarchical tasks, problems with a coarse-grained decomposition and applications characterised by minimal coupling among subtasks. The contract net strategy presents various advantages (e.g. better agreements due to dynamic task allocation, dynamic introduction/removal of agents, natural load-balancing, reliable mechanism for distributed control and failure recovery) as well as some limitations such as communication-intensity and, more important, the fact that it does not detect or resolve conflicts presuming only passive, benevolent and non-antagonistic agents (which is unrealistic for many real-world problems) (Nwana, Lee et al. 1996; Green, Hurst et al. 1997).

Multi-agent planning employs a detailed plan of agents' future actions and interactions (needed to achieve their goals) to avoid inconsistence and conflicts (Nwana 1996). There are two types of multi-agent planning i.e. centralised and distributed. The centralised multi-agent planning uses a coordinating agent to identify potential inconsistencies and conflicting interactions from the local plans sent by individual agents. Applications of this approach include the air-traffic control domain (implemented by Cammarata, McArthur and Steeb) and the MATPEN model proposed by Jin and Koyama (as cited in (Nwana 1996)). The distributed multi-agent planning allows agents to build and update their individual plans as well as to model other agents' plans until all plan conflicts are resolved. This technique was used by Lesser and Corkill in their functionally accurate, cooperative (FA/C) approach for structuring distributed processing systems (Lesser and Corkill 1981). Durfee implemented a framework for coordinating multiple AI systems cooperating in a distributed environment called partial global planning (Durfee and Lesser 1991). The main disadvantage of the multi-agent planning technique for coordination is that it requires more computation and communication than other approaches because agents have to share and process substantial amounts of information (Green, Hurst et al. 1997).

Social laws is another technique that can be applied for coordination among intelligent agents. Conflicts among agents' actions can be avoided if any agent would have complete knowledge of the goals and intentions of all agents (Green, Hurst et al. 1997). Chaib-draa proposes a framework for designing MAS in which agents "are capable of coordinating their activities in routine, familiar, and unfamiliar situations" (Chaib-draa 1996). The guiding principles of this strategy are that coordination is easier in routine than in unfamiliar situations and that all agents adopt and obey social laws such as social regularities and social collectivities.

Computational market-based mechanisms facilitate distribution of tasks and resource allocation through the use of auction-inspired protocols. This strategy can enhance the adaptivity, robustness and flexibility of MAS. However, Oliveira et al note that this technique "should include some risk assessment and risk management features" and the overall system performance needs to be further studied (Oliveira, Fischer et al. 1999).

3.3.4. Negotiation in Multi-Agent Systems

Representing the focus of many research studies, negotiation is essential within a MAS for conflict resolution and can be regarded as a significant aspect of the coordination process among autonomous agents (Nwana, Lee et al. 1996; Green, Hurst et al. 1997; Jennings,

Sycara et al. 1998; Oliveira, Fischer et al. 1999). Negotiation has been defined as "the communication process of a group of agents in order to reach a mutually accepted agreement on some matter" (as cited in (Green, Hurst et al. 1997)). In accord with this definition, Fatima et al view negotiation as "a means for agents to communicate and compromise to reach mutually beneficial agreements" indicating that "agents can mutually benefit from reaching agreement on an outcome from a set of possible outcomes, but have conflicting interests over the set of outcomes" (Fatima, Wooldridge et al. 2004). The main characteristics of negotiation include the existence of a conflict, the need to resolve the conflict in a decentralised manner by self-interested agents, bounded rationality and incomplete information (Jennings, Sycara et al. 1998).

Research shows that an effective negotiation process may be achieved by having agents reasoning about the beliefs, desires and intentions of other agents (Rao and Georgeff 1995; Nwana, Lee et al. 1996). This approach motivates the interest in other research areas such as logic, case-based reasoning, belief revisions, distributed truth maintenance, model-based reasoning, optimisation and game theory (Nwana and Wooldridge 1996; Nwana 1996; Zlotkin and Rosenschein 1996; Green, Hurst et al. 1997; Oliveira, Fischer et al. 1999; Shintani, Ito et al. 2000).

Game theory-based negotiation involves the application of concepts such as utility functions, space of deals and strategies and negotiation protocols. Agents use payoff matrices to represent common knowledge (each agent knows the *utility value* of the outcome of some interaction). Following a set of rules that govern the negotiation (the negotiation protocol), agents exchange their offers during an interactive process until an acceptable deal is reached (Nwana and Wooldridge 1996; Nwana 1996; Oliveira, Fischer et al. 1999). The key researchers of this area are Zlotkin and Rosenschein who use game theory to achieve coordination among autonomous agents in cooperative domains (Zlotkin and Rosenschein 1989; Zlotkin and Rosenschein 1996). The main critique to this technique stresses the lack of realism due to the fact that agents are presumed to be fully rational and to have full knowledge of other agents' values (Nwana 1996).

In a recently published study, Fatima, Wooldridge and Jennings (Fatima, Wooldridge et al. 2004) propose *an agenda-based model for multi-issue negotiation* under time constraints in an incomplete information setting. A bargaining equilibrium exists even with uncertain and partial information for each agent. They argue that the problems of existing game theoretic models are overcome by treating each agent's information state as its private knowledge and by considering agent deadlines.

Many negotiation techniques are inspired from the human negotiation strategies. (Nwana 1996; Jennings, Sycara et al. 1998) Motivated by theoretical analysis and observations of human interactions, Sycara and her research team adopt the logical model of the mental states of the agents (beliefs, desires, intentions and goals) to enable communication and negotiation among agents. Based on case-based reasoning and multi-attribute utility theory, Sycara proposed a system in which conflicts are resolved in labour relations with the aid of two practising negotiators. More recently, a new persuasive method for multiple-agent negotiation called *multiple negotiations* was proposed by Sycara and her team (Shintani, Ito et al. 2000).

3.3.5. Communication in Multi-Agent Systems

In order to achieve a beneficial agent interoperation, communication in a MAS is a requirement because agents need to exchange information and knowledge or to request the performance of a task since they only have a partial view over their environment (Green, Hurst et al. 1997; Jennings, Sycara et al. 1998; Nwana and Ndumu 1999). Considering the complexity of the information resources exchanged, agents should communicate through an *agent communication language* (ACL) (Genesereth and Ketchpel 1994; Nwana 1996; Green, Hurst et al. 1997; Labrou, Finin et al. 1999; Chaib-draa and Dignum 2002). Chaib-draa indicates that "the main objective of an ACL is to model a suitable framework that allows heterogeneous agents to interact, to communicate with meaningful statements that convey information about their environment or knowledge" (Chaib-draa and Dignum 2002). Nwana et al classify ACLs in two categories i.e. *ad hoc* and *standard* (Nwana 1996) Many agent-based applications contain collaborative agents that communicate using an *ad hoc* set of performatives within ad hoc ACLs or by depositing information in a shared database. However, this approach does not support interoperation between agent applications created by different developers.

Therefore, *standard* ACLs are essential to the cooperation process among various autonomous agents. Designed to support interactions among intelligent software agents, the Knowledge Query and Manipulation Language (KQML) is such a standard ACL proposed by the Knowledge Sharing Effort (KSE) consortium (Finin, Fritzson et al. 1994). KQML is a high-level communication language and set of protocols for identifying, connecting with and exchanging information and knowledge among agents. Agents can specify the information requirements and capabilities using a set of KQML *performatives* that define the allowed "speech acts" (that agents may attempt in communicating with each

other) and support the creation of more complex co-ordination and negotiation strategies. Run-time knowledge sharing is facilitated by a special type of agents called *communication facilitators* that coordinate the actions of other agents. KQML consists of three layers as follows (Finin, Labrou et al. 1997):

1. The *content layer* specifies the content of the message.
2. The *message layer* encodes a message using the set of performatives provided by the language. It determines the kinds of KQML agent interactions and specifies the protocol for delivering the message.
3. The *communication layer* is used for encoding low level communication parameters such as the identity of the sender and recipient and a unique identifier for the communication.

Figure 3.13 shows this layered organization of a KQML message.

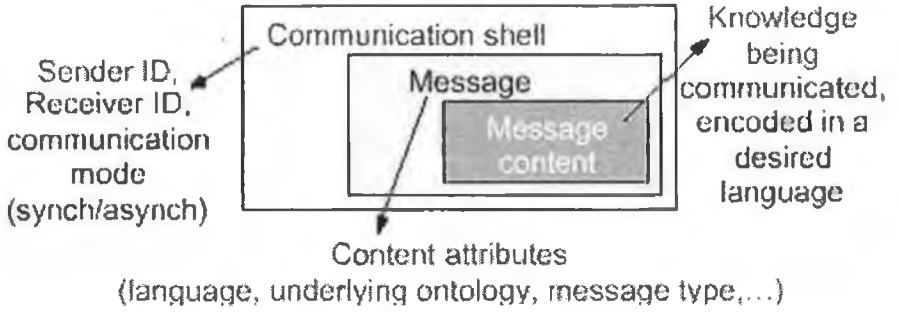


Figure 3.13. KQML layered organization (after (Devedzic 2001))

KSE researchers also designed a representation language for the contents of the messages called Knowledge Interchange Format (KIF) as an extension of first-order logic (Finin, Labrou et al. 1997). However, KQML is independent of the content language (KIF, SQL, etc) and of the ontology assumed by the content (Labrou, Finin et al. 1999).

Although KQML is probably the most used agent communication language/protocol in the agent community, many researchers have identified various limitations of the language (Cohen and Levesque 1995; Nwana 1996; Labrou, Finin et al. 1999). Labrou et al indicate that "different KQML implementations cannot interoperate" and that "there is no fixed specification sanctioned by a consensus-creating body" and "no agreed-upon semantics foundation" (Labrou, Finin et al. 1999).

FIPA ACL is another standard ACL proposed by the Foundation for Intelligent Physical Agents (FIPA). FIPA is a standards organisation (see section 3.4) in the area of software agents whose goal is to develop specifications that maximize interoperability within and across agent-based systems (http://www.fipa.org; Labrou, Finin et al. 1999; Poslad, Buckle

et al. 2000). Similarly to KQML, communication between FIPA agents relies on the speech act theory. FIPA ACL is based on a set of communicative acts (also called performatives) such as *request*, *inform* and *refuse* that are specified by FIPA independently from the overall content of the message (Dale and Mamdani 2001). The FIPA ACL also focuses on the effects on the mental attitudes (such as beliefs, desires, intentions) of the sender and receiver agents (Poslad, Buckle et al. 2000). The FIPA ACL message structure includes the identity of sender and receiver as well as the ontology and interaction protocol of the message. The content of the message supplied with a communicative act is expressed in a content language such as the FIPA semantic language (FIPA SL). To achieve the desired agent interaction, a set of FIPA interaction protocols (including requesting an action, contract net and several kinds of auctions) was created to describe entire conversations between agents (http://www.fipa.org).

In a comparison of KQML and FIPA ACLs, Labrou et al find the two languages almost identical with the primary difference in the details of their semantic frameworks (Labrou, Finin et al. 1999). However, Kumar et al indicate that "most contemporary agent communication languages, notably FIPA and KQML, have either no provision or no well-defined semantics for group communication" (Kumar, Huber et al. 2000). More recently, Chaib-draa and Dignum identified a set of issues in the development of ACLs and agent communication theory using KQML and FIPA ACLs as examples. Potential problems of these ACLs include the following (Chaib-draa and Dignum 2002):

- The linkage between the semantic theory and the theory of agency (which have to be aligned so that the ACL messages to be formally coherent).
- The semantics of KQML and FIPA-ACL (which are based on the mental agency while agents are almost never developed to use mental states).
- The verification of the semantics of an ACL and of an instantiation of a protocol to a protocol specification.
- The use of ontologies to interpret components of an ACL message.
- Limited coverage of communicative acts which are either assertives or directives (both KQML and FIPA ACL are extensible ACLs but the addition of performatives by different developers would lead to different incompatible dialects of these ACLs).
- The gap between individual messages and the extended message sequences (or conversations) that arise between agents.

Furthermore, because agents may have different terms for the same concept or identical terms for different concepts (Odell 2000), a meaningful communication process among agents requires, besides an ACL, a common understanding of all the concepts exchanged by agents. Ontologies (Gruber 1993; Guarino, Carrara et al. 1994; Borst, Akkermans et al. 1997; Studer, Benjamins et al. 1998; Uschold 1998; Noy and McGuinness 2001) represent the technology to support this requirement by semantically managing the knowledge from various application domains (Nwana and Ndumu 1999; Odell 2000; Chaib-draa and Dignum 2002). Supporting agent interoperation, a shared common ontology may contain the terms used in agent communication and the knowledge (e.g. definitions, attributes, relationships between terms and constraints) associated with them (Nwana 1996). Chaib-draa indicates that many ACLs need an ontology that should be characterised by the following (Chaib-draa and Dignum 2002):

- Broad coverage (for allowing multiple agents to share knowledge in several contexts).
- Extensibility (for allowing designers to add new elements).
- Relevance to the domain.

Both KQML and FIPA ACLs are designed to be independent of particular application vocabularies (by identifying the source of the vocabulary used in the message content). However, "the way that an agent would make use of the KQML or FIPA-ACL ontology specification to interpret unfamiliar parts of an ACL message has never been precisely defined" (Chaib-draa and Dignum 2002). Active ongoing research is still focusing on the general ontological problem.

3.3.6. Ontologies

As indicated in the previous subsection, many researchers identified the need to use ontologies for domain knowledge representation in order to meaningfully support agent interoperation (Nwana and Ndumu 1999; Odell 2000; Chaib-draa and Dignum 2002). The study of ontologies has developed gradually from specific needs associated with the problem of knowledge management within a computational environment and particularly from the problem of knowledge sharing and reuse (emerged within AI) (Chira 2004). Ontologies overcome the difficulties raised by "monolithic, isolated knowledge systems" (Gruber 1991), by specifying content specific agreements to facilitate knowledge sharing and reuse among systems that submit to the same ontology/ontologies by the means of ontological commitments (Spyns, Meersman et al. 2002). They describe concepts and relations assumed to be always true independent from a particular domain by a community

of humans and/or agents that commit to that view of the world (Guarino 1997). Being generic and task-independent, ontologies differ from traditional database schemas from the following perspectives (Fensel 2000):

- "A language for defining ontologies is syntactically and semantically richer than common approaches for databases."
- "The information that is described by an ontology consists of semi-structured natural language text and not tabular information."
- "An ontology must be shared and consensual terminology because it is used for information sharing and exchange."
- "An ontology provides domain theory and not the structure of a data container."

Many researchers (Neches, Fikes et al. 1991; Gruber 1993; Guarino, Carrara et al. 1994; Borst, Akkermans et al. 1997; Studer, Benjamins et al. 1998; Uschold 1998; Fikes 1999; Sowa 2000; Noy and McGuinness 2001) have proposed ontology definitions from an AI sense i.e. an ontology as a language dependent formal artefact (Guarino 1998). A merge of Gruber (Gruber 1993) and Borst et al (Borst, Akkermans et al. 1997) definitions is generally accepted by researchers, as follows: "Ontologies are *explicit formal* specification of a *shared conceptualization*" (Studer, Benjamins et al. 1998), where *explicit* means that "the type of concepts used, and the constraints on their use are explicitly defined", *formal* means that "the ontology should be machine readable, which excludes natural language", *shared* "reflects the notion that an ontology captures consensual knowledge, that is, it is not private to some individual, but accepted by a group" and *conceptualization* emphasizes the "abstract model of some phenomenon in the world by having identified the relevant concepts of that phenomenon". Most definitions and interpretations of ontologies use consensus and formality as the key characteristics (Chira 2004). However, only the *consensus* property is generally accepted to support the representation of knowledge from an ontology in a consensual manner. Regarding the *formality* requirement, Uschold (Uschold 1998) allows ontologies to be expressed in a restricted and structured form of natural language, while Gruber (Gruber 1993) enforces a well-defined logical model for ontologies. However, the general vision is that ontologies should be machine-enabled and, if not directly human-readable, they should at least contain plain text notices or explanations of concepts and relations for the human user (Borst, Akkermans et al. 1997; Guarino 1998; Studer, Benjamins et al. 1998; Uschold 1998; Fikes 1999; Sowa 2000; Noy and McGuinness 2001).

Methodologies for building ontologies represent the focus of active ongoing research within the AI community. The pioneer methodologies include the ones developed within the Enterprise Ontology project (Uschold and King 1995) and the TOVE project (Gruninger and Fox 1995). Developed within the Laboratory of Artificial Intelligence at the Polytechnic University of Madrid, the Methontology (Fernandez, Gomez-Perez et al. 1997; Gomez-Perez 1998) approach to building ontologies is recommended by FIPA and seems to be the most appreciated methodology for ontology construction within the AI community (Chira 2004). Based on the IEEE 1074-1995 standard (Fernandez-Lopez 2001), the proposed framework includes three main processes i.e. identification of the ontology development process, ontology life cycle based on evolving prototypes and particular techniques for carrying out each activity (Blazquez, Fernandez et al. 1998). However, the testing and validation of methodologies for building ontologies is still in its infancy as there are no tools available for a testing phase and there are not enough ontology developers to practically test the various methodologies (Chira 2004). Furthermore, none of the methodologies proposed by different ontology research groups are as mature as those from the knowledge engineering and software engineering fields.

Ontologies are currently very popular mainly within fields that require a knowledge-intensive approach to their methodologies and system development, such as knowledge engineering (Gruber 1993; Uschold and Gruninger 1996; Gaines 1997), knowledge representation (Artala, Franconi et al. 1996), qualitative modeling, language engineering, database design (Van de Riet 1998), information modeling (Weber 1997), information integration (Bergamaschi, Castano et al. 1998; Mena 1998), knowledge management and organization and agent-based design (Nwana 1996; Odell 2000; Chaib-draa and Dignum 2002).

3.3.7. Trust in Multi-Agent Systems

As already indicated, interoperation between agents plays a crucial role in MAS. In addition to negotiation, coordination and communication models, large-scale open distributed systems also require MAS architectures to address the issue of *trust* among agents (Wong and Sycara 1999; Ramchurn, Huynh et al. 2004; Jiang, Xia et al. 2005). Because interacting agents in practical contexts will probably never achieve a state of perfect information about the environment and the properties of partner-agents, "agents have to trust each other in order to minimise the uncertainty associated with interactions in open distributed systems" (Ramchurn, Huynh et al. 2004). Moreover, trust needs to be

addressed because the application of MAS in large-scale open distributed systems presents new challenges such as (Ramchurn, Huynh et al. 2004):

- Agents may represent different parties with potentially different aims and objectives.
- Open systems allow agents to come and go at any time.
- Agents with different characteristics (e.g. policies, abilities, roles) may be required to interact with one another.
- Agents can trade products or services, and collaborate in many different ways.

Trust in MAS has been defined as "a belief an agent has that the other party will *do what it says it will* (being honest and reliable), given an *opportunity to defect* to get higher payoffs" (Ramchurn, Huynh et al. 2004). Ramchurn et al conceptualize trust in two levels as follows (see Figure 3.14):

- *Individual-level trust:* having some beliefs about its opponents, an agent can reason about its level of trust in its interaction partners.
- *System-level trust:* agents are forced to trust the actions of their interaction partners.

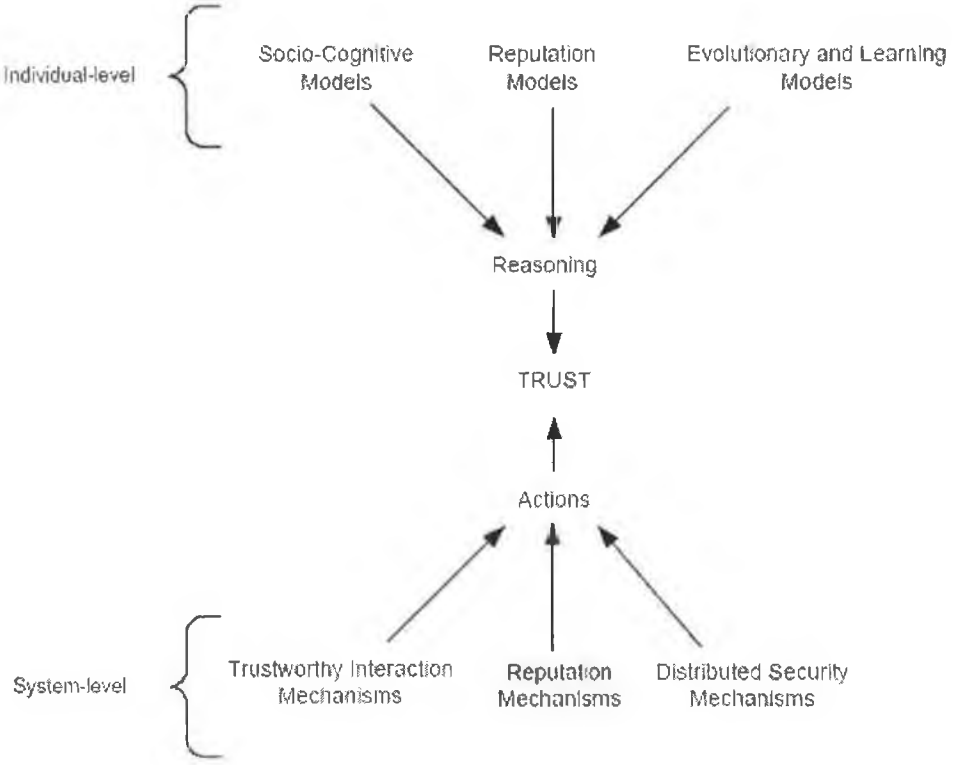


Figure 3.14. Approaches to trust in MAS (Ramchurn, Huynh et al. 2004)

While individual-level trust models enable agents to reason about strategies, motivations, capabilities and other information about potential interaction partners to decide whether to believe in their trustworthiness, system-level trust models require agents to act and interact

truthfully by using agent reputation information and imposing conditions and specified standards to ensure that the actions of other agents can be trusted (Ramchurn, Huynh et al. 2004). Even if advances in the field of MAS security and trust have been made, there is still much theoretical and practical work to be done.

3.4. Agent Standards

With the rapid growth of MAS exploitation in a variety of domains, a number of research groups started to address the issue of standardization of agent technology. Striving to become a "significant and generic computing technology" (Luck, McBurney et al. 2003), agent-based systems should follow common standards especially for the interoperability among different systems or components (Dickinson 1997; Chen and Su 2003). With the potential of attracting investments by industrial corporations, standardization is viewed as a "good mechanism to facilitate the development of both agent research and agent-based software products" (Dickinson 1997).

Two major organizations i.e. Object Management Group (OMG) and Foundation of Intelligent Physical Agents (FIPA) have addressed the issue of agent standardization.

Founded in 1989, OMG (http://www.omg.org) is an international organization that produces and maintains computer industry specifications for interoperable enterprise applications. OMG proposed Mobile Agent System Interoperability Facilities (MASIF) to address interoperability between agents systems created by different vendors but written in the same language (OMG 2000). The goal of MASIF is to standardize agent management, agent transfer, agent and agent system names, agent system types, location syntax and agent tracking. MASIF defines a reference model which includes the following concepts (http://www.omg.org):

- *Agent System:* a platform that can create, interpret, execute, transfer, and terminate agents.
- *Agent System Type:* the profile of an agent e.g. vendor, language, serialization mechanism.
- *Place:* a context within an agent system in which an agent can execute.
- *Region:* set of agent systems with the same authority but not necessarily the same agent system type.

Furthermore, OMG considers the integration of the Common Object Request Broker Architecture (CORBA) services into the proposed MASIF specification.

Formed in 1996, FIPA (http://www.fipa.org) promotes the development of standard specifications for open and interoperable agent infrastructures and offers guidelines for industrial development of agent platforms and agent-based applications. The latest specifications produced by FIPA are the FIPA 2000 standards (the previous versions FIPA 97 and FIPA 98 are already considered obsolete). The FIPA 2000 standard contains specifications on abstract architectures, content languages, agent management, interaction protocols, ACL message transport, quality and service. Playing an essential role, the FIPA agent management reference model addresses the creation, registration, location, communication, migration and retirement of agents. The FIPA reference model consists of the following logical components (FIPA 2004):

- *Agent:* "a computational process that implements the autonomous, communicating functionality of an application". An ACL is used for communication between agents. Each agent is distinguished unambiguously within the Agent Universe through an Agent Identifier (AID).
- *Directory Facilitator (DF):* an optional component that provides yellow pages services to all agents.
- *Agent Management System (AMS):* supervises the access and use of the Agent Platform by maintaining a directory of AIDs for registered agents. It offers white pages services to all agents.
- *Message Transport Service (MTS):* the default communication mechanism between agents from different Agent Platforms.
- *Agent Platform (AP):* "the physical infrastructure in which agents can be deployed". It consists of the machine(s), operating system, agent support software, DF, AMS, MTS and agents.

Considered a major contribution in enhancing visibility, credibility and feasibility of agents in real markets, the FIPA standard specifications are largely employed for academic and industrial development of various agent applications e.g. 15 different FIPA compliant platforms, 70 FIPA related projects, 159 FIPA compliant platforms registered within the Agentcities.NET[9] network (Calisti 2003). Publicly available implementations of agent platforms compliant with the FIPA specifications include FIPA-OS - http://www.emorphia.com, JACK Intelligent Agents - http://www.agent-software.com, JADE - http://jade.cselt.it, JAS - http://www.java-agent.org, LEAP - http://leap.crm-

[9] http://www.agentcities.net/globalapd.jsp

paris.com and Grasshopper - http://www.grasshopper.de (see section 3.5 for a short review).

Other proposals for agent standardization include the ARPA Knowledge Sharing Effort (Neches 1994) and the KAoS architecture (Shintani, Ito et al. 2000). In addition to these *de jure* standards (i.e. created by consensus and debate among concerned research or industrial groups of people), some *de facto* standards may arise from the widespread adoption of certain available agent technologies (e.g. JADE, JatLITE, Aglets, JACK Intelligent Agents). Common features of such technologies may include the programming language, the communication language between agents or the semantic language.

3.5. Agent-Oriented Methodologies

Many researchers in the area of agents and MAS believe that agent-based computing has the potential to improve the conceptualisation, design and implementation of complex distributed software systems (Oliveira, Fischer et al. 1999; Jennings 2000; Odell 2000; Wooldridge and Ciancarini 2001; Zambonelli, Jennings et al. 2003). A systematic methodology for the analysis and design of agent-based applications is a crucial requirement for the success of agent-oriented software engineering (AOSE) (Jennings 2000; Wooldridge and Ciancarini 2001). Moreover, Ndumu and Nwana indicate that "appropriate design methodologies for constructing the different types of agent systems for different application domains" are needed before "generic platforms for engineering agent-based applications" (Ndumu and Nwana 1996).

Although there are many agent theories, languages and architectures available, little work has been done in the area of agent-oriented methodologies to assist the developer in all the phases of the life cycle of an agent-based application. The available methodologies for the analysis and design of agent-based systems can be classified in two groups as follows (Iglesias, Garijo et al. 1999; Wooldridge and Ciancarini 2001):

- Methodologies that extend or adapt object-oriented methodologies e.g. AAII (Kinny, Georgeff et al. 1996), Gaia (Wooldridge, Jennings et al. 2000), MaSE (DeLoach 1999), AUML (Odell, Parunak et al. 2000; Odell, Nodine et al. 2005).
- Methodologies that adapt knowledge engineering models or other techniques e.g. DESIRE (Brazier, Dunin-Keplicz et al. 1997).

The *Agent Modelling Technique for Systems of BDI agents* (also called the *AAII* – Australian Artificial Intelligence Institute – methodology) was developed by Kinny et al

(Kinny, Georgeff et al. 1996) by building upon and adapting existing object-oriented models. Based on the BDI paradigm, this agent-oriented methodology and modelling technique provides both internal and external perspectives of MAS. The internal model deals with the agent's beliefs, desires and intentions. Presenting a system level view, the external model decomposes the system into agents and deals with the relationships between them.

Wooldridge et al (Wooldridge, Jennings et al. 2000) proposed a methodology for agent-oriented analysis and design called *Gaia*. Using the Gaia methodology, the designer of an agent-based application can systematically progress from a set of requirements to a detailed design ready to be implemented. Extending object-oriented analysis and design models, Gaia supports the developer to model complex systems through a process of organisational design. It provides a set of agent-specific concepts, which are of two types: abstract (i.e. roles, permissions, responsibilities, protocols, activities, liveness properties and safety properties) and concrete (i.e. agent types, services, acquaintances). Abstract concepts are used to conceptualise the system during analysis while concrete concepts are used within the design process. More recently, the Gaia methodology has been extended for the analysis and design of MAS (Zambonelli, Jennings et al. 2003).

Proposed by DeLoach (DeLoach 1999; DeLoach, Wood et al. 2001), the *Multiagent Systems Engineering (MaSE)* methodology for formal agent system synthesis is a further abstraction of the object-oriented paradigm. Two languages i.e. Agent Modeling Language (AgML) and Agent Definition Language (AgDL) are used to describe agents and MAS. Graphically based models are used to describe system goals, behaviours, agent types and agent communication interfaces.

Odell et al (Odell, Parunak et al. 2000; Bauer, Müller et al. 2001; Odell, Nodine et al. 2005) explored UML (Unified Modeling Language) idioms and extensions that can be used to model agents and agent-based systems. The result is an UML-based approach to building agent applications called *Agent UML (AUML)*. It should be noted that UML is a widely accepted standard for object-oriented modelling but it is not a methodology (it is rather a language). AUML extends the UML notation by supporting concurrent threads of interaction and allowing an agent to play many roles. Both FIPA (http://www.fipa.org) and OMG (http://www.omg.org) groups are recommending the use of UML extensions for the specification of agent-based systems.

All these extensions of object-oriented methodologies share a number of advantages as follows (Iglesias, Garijo et al. 1999):

- The benefit of the similarities between the object-oriented paradigm and the agent-oriented paradigm.
- The commonly usage of object-oriented languages to implement agent-based applications (as shown in the next section).
- The popularity of object-oriented methodologies.

However, "object-oriented methodologies simply do not allow us to capture many aspects of agent systems" (Wooldridge and Ciancarini 2001). Most disadvantages of these methodologies fall out from the differences between objects and agents, as follows (Iglesias, Garijo et al. 1999):

- Message passing used to communicate (for objects, is just method invocation while agents analyse and model these messages, can decide whether or not to execute the requested action and use complex protocols to negotiate).
- Lack of techniques for modelling the agent's mental state.
- Lack of procedures for modelling the social relationships between agents.

Because the predominant approach to implementing methodologies for agents and MAS is to extend object-oriented paradigms, little work has been done on agent-oriented methodologies that adapt knowledge engineering models. The *DESIRE* (DEsign and Specification of Interacting REasoning components) framework supports the specification and implementation of component-based autonomous interactive agents (Brazier, Dunin-Keplicz et al. 1997). Using DESIRE, the analyst can explicitly model complex reasoning within agents, communication patterns between agents as well as interactions with the external world. This high-level modelling framework supports conceptual design and specification of both dynamic and static aspects of agent behaviour. Brazier et al report the successful application of the compositional multi-agent framework DESIRE to develop a conceptual specification of simple agents and to simulate the behaviour in a dynamic environment (Brazier, Eck et al. 2001).

Other approaches to modelling MAS such as CoMoMAS and MAS-CommonKADS extend the existing CommonKADS methodology for knowledge engineering (Iglesias, Garijo et al. 1999).

The advantages of adapting knowledge engineering methodologies for the design of agent-based systems can be summarised as follows (Iglesias, Garijo et al. 1999):

- They provide techniques for modelling the agent's knowledge (knowledge acquisition process).
- The existing tools, ontology libraries and problem solving method libraries can be reused.

However, these methodologies "do not address the distributed or social aspects of the agents, or their reflective and goal-oriented attitudes" (Iglesias, Garijo et al. 1999) because a knowledge based system is conceived as a centralised one.

3.6. Agent Languages and Environments

The growing interest in the area of software agents and MAS motivated the development of languages that facilitate the design and construction of agent-based applications. Wooldridge and Jennings define an agent language as "a system that allows one to program hardware or software computer systems in terms of some of the concepts developed by agent theorists" (Wooldridge and Jennings 1995).

Although several languages and platforms have been created by different research groups and companies to support the development of agent-based applications, traditional languages are still used to construct agent applications. Nwana and Wooldridge indicate, "typically, object-oriented languages such as Smalltalk, Java or C++ lend themselves more easily for the construction of agent systems" (Nwana and Wooldridge 1996). The reason for this is that agents and objects share some properties such as encapsulation, inheritance and message passing. However, objects may respond to the same message in different ways (polymorphism) while agents must have a common ACL. In object-oriented programming, an object can decide to invoke a method of another object but when an agent wants to do the same thing (to request an action from another agent) the decision lies with the agent that receives the request. Jennings et al summarise this distinction by observing "objects do it for free; agents do it for money" (Jennings, Sycara et al. 1998). The most used programming language for developing agent applications is Java due to its rich library of functions tackling concurrency as well as security (Huget 2002), support for object-oriented programming techniques, code portability, native support for multithreading and introspection of object properties and methods (Bigus, Schlosnagle et al. 2002).

Zambonelli et al indicate that "agent-based computing promotes an abstraction level that is suitable for modern scenarios and that is appropriate for building flexible, highly modular, and robust systems, *whatever the technology adopted to actually build the agents*" (Zambonelli, Jennings et al. 2003). However, it should be noted that there is a very large

number of agent languages, environments, frameworks and toolkits available and only some of them will be reviewed in the following, as it is impossible to be exhaustive.

3.6.1. Agent-Oriented Programming

In the early 90s, Shoham proposed a new programming paradigm called agent-oriented programming (AOP) that "promotes a societal view of computation" (Shoham 1998). A specialization of object-oriented programming (OOP), AOP allows the direct programming of agents in terms of their mental state (consisting of components such as beliefs, decisions, capabilities and obligations). Agent programs control agents and include communication primitives such as informing, requesting and offering (based on the speech act theory). Table 3.6 presents the relation between OOP and AOP (Shoham 1998).

Property	OOP	AOP
Basic unit	object	agent
Parameters defining state of basic unit	unconstrained	beliefs, commitments, capabilities, choices,...
Process of computation	message passing and response methods	message passing and response methods
Types of messages	unconstrained	inform, request, offer, promise, decline,...
Constraints on methods	none	honesty, consistency

Table 3.6. The relation between OOP and AOP (Shoham 1998)

Shoham indicates that a complete AOP system contains a restricted formal language for describing the mental state, an interpreted programming language for defining and programming agents and an 'agentifier' for converting neutral devices into programmable agents. Furthermore, Shoham developed the *AGENT0* programming language as an implementation of the AOP paradigm. AGENT0 allows the specification of an agent using a set of capabilities, a set of initial beliefs, a set of initial commitments and a set of commitment rules. Each commitment rule consists of a message condition, a mental condition and an action. The agent becomes committed to the action if the message condition matches the messages received by the agent and the mental condition matches the beliefs of the agent (Shoham 1998). Figure 3.15 presents the control flow in AGENT0 (Wooldridge 1999). An operation loop of the agent consists of reading all current messages, updating beliefs and executing all necessary commitments.

Although Shoham's work is of significant importance in the research area of agent languages, Wooldridge notes that the AGENT0 AOP language is only a prototype "not intended for building anything like large-scale production systems" and is limited because

"the relationship between the logic and interpreted programming language is only loosely defined" (Wooldridge 1999).

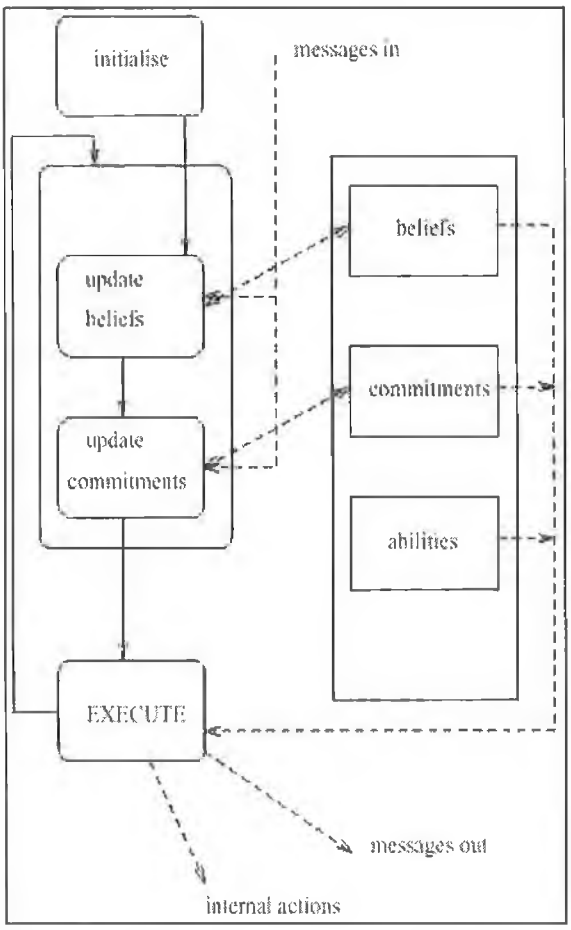


Figure 3.15. The flow of control in the AGENT0 language (after (Wooldridge 1999))

Another logical approach to agent programming is *Concurrent METATEM* (Fisher 1994) a multi-agent programming language based upon the direct execution of linear time temporal logic agent specifications. A Concurrent METATEM system consists of concurrently executing agents whose behaviour is implemented using executable temporal logic and which can communicate via asynchronous broadcast message passing.

Created by Rao, *AgentSpeak(L)* (Rao 1996) is a programming language that allows the formalization of BDI agents. It is based on restricted first-order language with events and actions and consists of a set of base beliefs and a set of context-sensitive plans allowing hierarchical decomposition of goals.

Extending a previous version, *ConGolog (Concurrent Golog)* is a concurrent programming language for process specification and agent programming (Giacomo, Lespérance et al. 2000). It handles concurrent processes with possibly different priorities, high-level interrupts and arbitrary exogenous actions. ConGolog supports the formal specification of

complex MAS but "lacks features for modelling the rationale behind design choices" (Wang and Lespérance 2001).

3.6.2. Agent Toolkits and Frameworks

Over the last years, a large number of toolkits and developing environments have been created to support the agent developer in the task of implementing agent-based systems.

ZEUS (Nwana, Ndumu et al. 1999) is a toolkit for constructing collaborative multi-agent applications developed by BT Laboratories. Implemented in the Java programming language, ZEUS facilitates the creation of agents by specifying the attributes and the tasks of individual agents. A generator tool can then be invoked to create the source code implementation for each agent. The developer can make use of an agent component library, visualisation tools and agent building software components in order to access the application-independent agent-level functionality required of collaborative agents, to observe the agents' behaviour and to interactively create agents by specifying their attributes and strategies. Initially, ZEUS supported only KQML as the agent communication language but it was further developed to support FIPA ACL as well. ZEUS is an open-source software freely available under an general public license.

Fully implemented in Java, *JADE - Java Agent DEvelopment Framework* - (Bellifemine, Poggi et al. 1999) is a software framework that facilitates the development of MAS. The JADE agent platform is compliant with the FIPA specifications and performs all agent communication through message passing (using FIPA ACL to represent messages). JADE adopts the multi-thread solution (offered by Java) and supports scheduling of cooperative behaviours. The graphical user interface facilitates the remote management, monitoring and controlling of the status of agents, the creation and execution of an agent on a remote host as well as control of other FIPA compliant agent platforms. JADE successfully participated in the FIPA interoperability tests and is currently under further development.

Developed by Nortel Networks, *FIPA-OS* (Poslad, Buckle et al. 2000) is an open agent platform designed to comply with the FIPA agent standards (http://www.fipa.org). It supports communication between multiple agents and operates in a heterogeneous open service environment. The key aspect of FIPA-OS is openness, which is accentuated by the fact that the platform software is distributed and managed under an open-source licensing scheme.

Developed by the Agent Oriented Software group (http://www.agent-software.com), *JACK Intelligent Agents* (Howden, Ronnquist et al. 2001) is a Java framework for multi-agent system development. It supports the BDI architecture model but it can be extended to support different agent models and specific application requirements. The JACK agent language extends the Java programming language with agent-oriented concepts such as agents, capabilities, events, plans, agent knowledge bases, and resource and concurrency management. Using a component-based approach, JACK provides the core architecture and infrastructure for building, running and integrating software agents in distributed applications.

The *Open Agent Architecture (OAA)* (Cheyer and Martin 2001) is a domain-independent framework for constructing agent-based systems. The facilitator agent and libraries (in several languages) can be used for creating application agents. Coordination and communication among agents is addressed by one ore more facilitators (specialized server agents) included in each OAA-based system. Supporting flexible, dynamic configurations of autonomous agents, OAA facilitates the declaration of capabilities by service-providing agents, the construction of goals by users and service-requesting agents, the creation and maintenance of shared repositories of data and the use of triggers to instantiate commitments within and between agents.

Developed at Stanford University, *JATLite (Java Agent Template, Lite)* (Jeon, Petrie et al. 2000) is a collection of Java objects and class libraries that facilitates the creation of software agents communicating robustly over the Internet in order to perform a distributed computation. Agent messages used in the communication process are primarily based on the KQML language and protocol. JATLite features include multi-threaded operation, a message router for agent registration, connection, name and password services, storage and queuing of messages for mobile and sporadic agents, and support for both stand-alone agents in Java and C++ and applet agents.

Other available agent toolkits and multi-agent platforms include the Java-based system AgentBuilder (http://www.agentbuilder.com/), the IBM Agent Building and Learning Environment (ABLE) (Bigus, Schlosnagle et al. 2002), the Java Intelligent Agent Component Ware (JIAC) (Ballmann and Wieczorek 1998), the Java-based Grasshopper (IKV++GmbH 2001), the International Knowledge System's AgentX (http://www.iks.com/agentx.htm), Mitsubishi's Java-based mobile agent system Concordia (Kiniry and Zimmerman 1997), the Direct Intelligent Adaptation (DirectIA) system for

creating adaptive agents (http://www.directia.com/), the Agent Development Toolkit (ADK) for mobile agents (http://www.tryllian.com), the iGEN toolkit for building cognitive agents (http://www.cognitiveagent.com), IBM Japan's Java-based autonomous software agent technology Aglets Software Development Kit (http://www.trl.ibm.com/aglets), the SodaBot system developed at MIT Artificial Intelligence Lab (http://www.ai.mit.edu/people/sodabot/sodabot.html). Offering varied functionality, these systems do not necessarily adhere to any standards: an agent created using one system will not work in another. Also, "there is no uniform support for communication protocols across these tools either" (Odell 2000). However, the majority of these agent systems adopted the following strategies (Odell 2000):

- The programming language used for implementation is Java or C++.
- The communication language for agent interoperability is KQML or FIPA ACL.
- The content language used for representing knowledge is KIF or FIPA SL.

Some of the commercial agent-building systems are purchasable and some are freely available under a general public license.

3.7. Applications of Agents and Multi-Agent Systems

Several application areas are currently focused on the employment of agents and MAS in complex problem solving processes. Jennings and Wooldridge indicate that the agent-based solution is appropriate for open (or at least, highly dynamic, uncertain or complex) environments in which flexible and autonomous agents may be the only solution (Jennings and Wooldridge 1998; Wooldridge 1998). Domains in which data, control, expertise or resources are inherently distributed can be addressed using agent technology. Finally, the agent-based approach is suitable for environments that are naturally modelled as societies of autonomous cooperating components (the agent is a natural metaphor) as well as for systems that contain legacy components (see section 3.2) (Jennings and Wooldridge 1998). Also, Oliveira et al identify elements such as distribution, complexity, flexible interaction, highly dynamic environments and openness as the typical properties of the application domains in which multi-agent technologies are most appropriate (Oliveira, Fischer et al. 1999):

Jennings and Wooldridge present a classification of agent applications by the domain to which they are applied (Jennings and Wooldridge 1998):

- *Industrial applications*: process control, manufacturing, and air traffic control.

- *Commercial applications*: information management, electronic commerce, and business process management.
- *Medical applications*: patient monitoring, and health care.
- *Entertainment*: games, and interactive theatre and cinema.

To these, Wooldridge adds the following domains (Wooldridge 1998): industrial systems management, distributed sensing, space shuttle fault diagnosis, factory process control (distributed applications of agent technology). Also, in the area of agent applications for the Internet, there is special attention accorded to mobile agents that can migrate around the Internet working on the user's behalf. Other than the already mentioned electronic commerce domain, work has been done in the information gathering area and Personal Digital Assistant (PDA) systems. Another application area for agents is represented by interfaces: the agent assists the user during tasks by anticipating requirements (Wooldridge 1998).

Jennings and Wooldridge (Jennings and Wooldridge 1998) identify three dimensions along which these agent applications can be analysed: sophistication of the agents, role of the agents and granularity of view. There are three levels of sophistication based on the type of the agent's behaviour as follows (Jennings and Wooldridge 1998):

1. The *gopher* agent – executes simple tasks based on well-defined and pre-specified information.
2. The *service performing* agent – executes high-level tasks based on well-defined information.
3. The *predictive/proactive* agent – is capable of executing tasks in a flexible and autonomous manner on behalf of its user.

Secondly, agent based systems can be analysed by considering the role of the agents e.g. in industrial and commercial applications, the role is to provide decision support functionality (the final decision belongs to the user) while in the entertainment domain, the role is to completely solve the problem. Finally, granularity of view refers to the use of the individual agent as opposed to a society of agents for specific domains. Some applications adopt the single-agent approach, others use MAS (these are probably more complicated since issues such as communication, coordination and negotiation have to be considered by the developer) (Jennings and Wooldridge 1998).

Other classification schemes for agent applications are available from different authors or groups of researchers. For example, the OMG group (http://www.omg.org) classify the applications that use agents as follows (Odell 2000):

- Enterprise applications e.g. smart documents, goal-oriented enterprise, role and personnel management.
- Business-to-business applications e.g. market making for goods and services, team management.
- Process control e.g. intelligent buildings, plant management, robots.
- Personal agents e.g. email and news filters, personal schedule management, personal automatic secretary.
- Information management tasks e.g. searching for information, information filtering, information monitoring, data source mediation, interface agents/personal assistants.
- Nomadic computing applications (agents for mobile computing).

Although agents and MAS are increasingly employed in various application domains, there are several problems associated with the agent-based approach to building computer-based systems as follows (Jennings and Wooldridge 1998):

- *No overall system controller:* agents may not be the answer for domains with global constraints or that require a guaranteed real-time response.
- *No global perspective:* a basic issue in MAS research is that of integrating the agent's decisions based on local knowledge with the desire to achieve globally optimal performance.
- *Trust and delegation:* in order to delegate tasks to agents, users must trust agents that they work indeed in their behalf. Furthermore, "the agent must strike a balance between continually seeking guidance (and needlessly distracting the user) and never seeking guidance (and exceeding its authority). Put crudely, an agent must know its limitations." (Jennings and Wooldridge 1998).

While there are still many problems associated with the design and implementation of agent-based applications, MAS "provide a powerful model for computing in the 21st century, in which networks of interacting, real-time, intelligent agents seamlessly integrate the work of people and machines, and dynamically adapt their problem solving to effectively deal with changing usage patterns, resource configurations and available sources of expertise and information" (Lesser 1999).

3.8. Conclusions

Composed of interacting autonomous software agents, multi-agent systems offer a promising software engineering solution for developing robust and scalable software systems (Jennings 2000; Wooldridge and Ciancarini 2001; Luck, McBurney et al. 2003;

Zambonelli, Jennings et al. 2003). Enjoying certain special properties (e.g. autonomy, pro-activeness, communication, learning, temporal continuity, mobility) that distinguish them from standard programs (Nwana 1996; Bradshow 1997; Wooldridge 1999), agents have the potential to manage the complexity inherent in distributed software systems and therefore forming an important new agent-oriented software engineering paradigm (Jennings 2000; Wooldridge and Ciancarini 2001).

However, software agent research still lacks in universally accepted concepts from definitions, architectures, methodologies and languages to protocols for coordination, negotiation and communication. Ongoing research focuses on the development of agent-oriented methodologies and languages, the study of interoperation and trust models as well as the establishment of agent standards.

References

Anumba, C. J., O. O. Ugwu, L.Newnham and A.Thorpe (2002). "Collaborative design of structures using intelligent agents." Automation in Construction 11: 89-103.

Artala, A., E. Franconi, N. Guarino and L. Pazzi (1996). "Part-Whole Relations in Object-Centered Systems: an Overview." Data and Knowledge Engineering 20(3): 347-383.

Ballmann, S. and D. Wieczorek (1998). Java Intelligent Agent Component Ware (JIAC) - technical documentation. Berlin, DAI Laboratory Technical University of Berlin.

Bauer, B., J. P. Müller and J. Odell (2001). Agent UML: A Formalism for Specifying Multiagent Interaction. Agent-Oriented Software Engineering, Springer-Verlag, Berlin.

Bellifemine, F., A. Poggi and G. Rimassa (1999). JADE - A FIPA-compliant agent framework. Proceedings of PAAM'99, London.

Bergamaschi, S., S. Castano, S. D. C. d. Vimercati and M. Vincini (1998). An Intelligent Approach to Information Integration. Formal Ontology in Information System. N. Guarino. Amsterdam, IOS Press.

Bigus, J. P., D. A. Schlosnagle, J. R. Pilgrim, W. N. M. III and Y. Diao (2002). "ABLE: A toolkit for building multiagent autonomic systems." IBM Systems Journal 41(3).

Blazquez, M., M. Fernandez, J. M. Garcia-Pinar and A. Gomez-Perez (1998). Building Ontologies at the Knowledge Level using the Ontology Design Environment. 11th Knowledge Aquisition Workshop, KAW98, Bamff, Canada.

Borst, P., H. Akkermans and J. Top (1997). "Engineering Ontologies." International Journal of Human-Computer Studies 46(Special Issue on Using Explicit Ontologies in KBS Development): 365-406.

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

Brazier, F. M. T., B. M. Dunin-Keplicz, N. R. Jennings and J. Treur (1997). "DESIRE: Modelling Multi-Agent Systems in a Compositional Formal Framework." International Journal of Cooperative Information Systems 6(Special Issue on Formal Methods in Cooperative Information Systems: Multi-Agent Systems): 67-94.

Brazier, F. M. T., P. A. T. v. Eck and J. Treur (2001). "Modelling a Society of Simple Agents: From Conceptual Specification to Experimentation." Journal of Applied Intelligence 14: 161-178.

Brooks, R. A. (1986). "A robust layered control system for a mobile robot." IEEE Journal of Robotics and Automation 2: 14-23.

Calisti, M. (2003). FIPA standards for promoting interoperability of industrial agent systems. Agencities Info Days, Helsinki.

Carver, N., V. Lesser and Q. Long (1993). Distributed sensor Interpretation: Modelling Agent Interpretations in DRESUN, UMass Technical Report, UMCS 93-75.

Chaib-draa, B. (1996). "Interaction Between Agents in Routine, Familiar and Unfamiliar Situations." International Journal of Intelligent & Cooperative Information Systems 5(1): 1-25.

Chaib-draa, B. and F. Dignum (2002). "Trends in Agent Communication Language." Computational Intelligence 18(2).

Chen, J. J.-Y. and S.-W. Su (2003). "AgentGateway: A communication tool for multi-agent systems." Information Sciences 150: 153-164.

Cheyer, A. and D. Martin (2001). "The Open Agent Architecture." Journal of Autonomous Agents and Multi-Agent Systems 4(1): 143-148.

Chira, V. O. (2004). Towards a Machine Enabled Semantic Framework for Distributed Engineering Design. Department of Mechanical & Industrial Engineering. Galway, Galway-Mayo Institute of Technology.

Chu, E., K. Srihari and C. R. Emerson (1996). "Distributed Artificial Intelligence in Process Control." 19th International Conference on Computers and Industrial Engineering.

Cohen, P. R. and H. J. Levesque (1995). Communicative actions for artificial agents. Proceedings of the International Conference on Multi-Agent Systems, San Francisco, AAAI Press.

Dale, J. and E. Mamdani (2001). "Open Standards for Interoperating Agent-Based Systems." Software Focus, Wiley.

DeLoach, S. A. (1999). Multiagent Systems Engineering: A Methodology And Language for Designing Agent Systems. Agent-Oriented Information Systems (AOIS) '99.

DeLoach, S. A., M. F. Wood and C. H. Sparkman (2001). "Multiagent Systems Engineering." The International Journal of Software Engineering and Knowledge Engineering 11(3).

Devedzic, V. (2001). "Knowledge Modeling - State of the Art." Integrated Computer-Aided Engineering 8(3): 257-281.

Dickinson, I. (1997). "Agents Standards." Hewlett-Packard Company.

Durfee, E. H. (2001). "Scaling Up Agent Coordination Strategies." IEEE Computer 34(7): 39-46.

Durfee, E. H. and V. R. Lesser (1991). "Partial Global Planning: A Coordination Framework for Distributed Hypothesis Formation." IEEE Transactions on Systems, Man, and Cybernetics, Special Issue on Distributed Sensor Networks SMC-21(5): 1167-1183.

Fatima, S. S., M. Wooldridge and N. R. Jennings (2004). "An agenda-based framework for multi-issue negotiation." Artificial Intelligence 152: 1-45.

Fensel, D. (2000). Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce. Berlin, Springer.

Ferguson, I. A. (1992). "TouringMachines: Autonomous Agents with Attitudes." IEEE Computer 25(5).

Fernandez, M., A. Gomez-Perez and N. Juristo (1997). METHONTOLOGY: From Ontological Art Towards Ontological Engineering Workshop on Ontological Engineering. Symposium on ONtological Engineering of AAAI, Standford, California.

Fernandez-Lopez, M. (2001). "Overview Of Methodologies for Building Ontologies." Intelligent Systems 16(1): 26-34.

Fikes, R., Farquhar, A. (1999). "Distributed Repositories of Highly Expressive Reusable Ontologies." IEEE Intelligent Systems 14(2): 73-79.

Finin, T., R. Fritzson, D. McKay and R. McEntire (1994). KQML as an Agent Communication Language. Proceedings of the Third International Conference on Information and Knowledge Management.

Finin, T., Y. Labrou and J. Mayfield (1997). KQML as an agent communication language. Software Agents. B. M. Jeffrey, MIT Press.

FIPA (2004). FIPA Agent Management Specification.

Fisher, M. (1994). A Survey of Concurrent METATEM - The Language and its Applications. Proceedings of First International Conference on Temporal Logic (ICTL), Bonn, Germany, Springer-Verlag.

Foner, L. N. (1993). What's An Agent, Anyway? A Sociological Case Study, Media Laboratory, Massachusetts Institute of Technology.

Franklin, S. and A. Graesser (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, Springer-Verlag, 1996, Berlin, Germany.

Gaines, B. (1997). "Editorial: Using Explicit Ontologies in Knowledge-based System Development." International Journal of Human-Computer Systems 46: 181.

Gasser, L. (1998). Social conceptions of knowledge and action: DAI foundations and open systems dynamics. Readings in Agents. M. N. Huhns and M. P. Singh, Morgan Kaufmann Publishers.

Genesereth, M. R. and S. P. Ketchpel (1994). "Software Agents." Communications of the ACM, ACM Press.

Georgeff, M., B. Pell, M. Pollack, M. Tambe and M. Wooldridge (1999). The Belief-Desire-Intention Model of Agency. Intelligent Agents. J. P. Muller, M. Singh and A. Rao, Springer-Verlag. 1365.

Giacomo, G. D., Y. Lespérance and H. J. Levesque (2000). "ConGolog, a concurrent programming language based on the situation calculus." Artificial Intelligence 121: 109-169.

Gomez-Perez, A. (1998). Knowledge Sharing and Reuse. The Handbook on Expert Systems. Liebowitz, CRC Press.

Greaves, M., V. Stavridou-Coleman and R. Laddaga (2004). "Dependable Agent Systems." IEEE Intelligent Systems.

Green, S., L. Hurst, B. Nangle, P. Cunningham, F. Somers and R. Evans (1997). Software Agents: A review. Dublin, Intelligent Agents Group, Trinity College Dublin, Broadcom Eireann Research Ltd.

Gruber, T. R. (1991). The Role of Common Ontology in Achieving Shareable, Reusable Knowledge Bases. Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference, San Mateo, Morgan Kaufmann, 1991.

Gruber, T. R. (1993). "A Translation Approach to Portable Ontology Specification." Knowledge Acquisition 5(2): 199-220.

Gruninger, M. and M. S. Fox (1995). Methodology for the Design and Evaluation of Ontologies. IJCAI Workshop on Basic Ontological Issues in Knowledge Sharing, Montreal, Quebec, Canada.

Guarino, N. (1997). Semantic Matching: Formal Ontological Distinctions for Information Organization, Extraction, and Integration. Summer School on Information Extraction, Frascati, Italy, July 14-19.

Guarino, N. (1998). Formal Ontology and Information Systems. Formal Ontology in Information Systems. FOIS'98, 6-8 June 1998., Trento, IOS Press,.

Guarino, N., M. Carrara and P. Giaretta (1994). Formalizing Ontological Commitments. National Conference on Artificial Intelligence, AAAI 94, Seatle, Morgan Kaufmann.

Hayes-Roth, B. (1995). Agents on Stage: Advancing the State of the Art of AI. Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95).

Helin, H. (2003). Agent Architectures & Languages, http://www.cs.helsinki.fi/u/hhelin/opetus/oat/. 2003.

Howden, N., R. Ronnquist, A. Hodgson and A. Lucas (2001). JACK Intelligent Agents - Summary of an Agent Infrastructure. 5th International Conference on Autonomous Agents.

http://www.agentbuilder.com/, Last Accessed August 2005.

http://www.agent-software.com, Last Accessed August 2005.

http://www.ai.mit.edu/people/sodabot/sodabot.html, The SodaBot System, Last Accessed October 2004.

http://www.cognitiveagent.com, iGEN Overview, Last Accessed August 2005.

http://www.directia.com/, Last Accessed August 2005.

http://www.fipa.org, Foundation for Intelligent Physical Agents, Last Accessed August 2005.

http://www.iks.com/agentx.htm, Last Accessed September 2004.

http://www.omg.org, Object Management Group, Last Accessed August 2005.

http://www.trl.ibm.com/aglets, Aglets, Last Accessed August 2005.

http://www.tryllian.com, The Agent Development Kit (ADK), Last Accessed August 2005.

Huget, M.-P. (2002). Desiderata for Agent Oriented Programming Languages, University of Liverpool.

Iglesias, C. A., M. Garijo and J. C. Gonzalez (1999). A Survey of Agent-Oriented Methodologies. Proceedings of the 5th International Workshop on Intelligent Agents V : Agent Theories, Architectures, and Languages.

IKV++GmbH (2001). Grasshopper Basics And Concepts. http://www.grasshopper.de/.

Ingrand, F. F., M. P. Georgeff and A. S. Rao (1992). "An Architecture for Real-Time Reasoning and System Control." IEEE Expert 7(6): 33-44.

Jennings, N. R. (2000). "On agent-based software engineering." Artificial Intelligence.

Jennings, N. R., K. P. Sycara and M. Wooldridge (1998). "A Roadmap of Agent Research and Development." Journal of Autonomous Agents and Multi-Agent Systems 1(1): 7-36.

Jennings, N. R. and M. Wooldridge (1998). Applications of Agent Technology. Agent Technology: Foundations, Applications, and Markets. N. R. Jennings and M. Wooldridge, Springer-Verlag.

Jeon, H., C. Petrie and M. R. Cutkosky (2000). "JATLite: A Java Agent Infrastructure with Message Routing." IEEE Internet Computing.

Jiang, Y. C., Z. Y. Xia, Y. P. Zhong and S. Y. Zhang (2005). "Autonomous trust construction in multi-agent systems—a graph theory methodology." Advances in Engineering Software 36: 59-66.

Kiniry, J. and D. Zimmerman (1997). "A Look at Mitsubishi's Concordia." IEEE Internet Computing online.

Kinny, D., M. Georgeff and A. Rao (1996). A Methodology and Modelling Technique for Systems of BDI Agents. Agents Breaking Away, 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, Springer.

Kumar, S., M. J. Huber, D. R. McGee, P. R. Cohen and H. J. Levesque (2000). Semantics of Agent Communication Languages for Group Interaction. The Seventeenth National Conference on Artificial Intelligence (AAAI 2000), Austin, Texas, AAIT Press/The MIT Press.

Labrou, Y., T. Finin and Y. Peng (1999). "Agent Communication Languages: The Current Landscape." IEEE Intelligent Systems.

Lazansky, J., O. Stepankova, V. Marik and M. Pechoucek (2001). "Application of the multi-agent approach in production planning and modelling." Engineering Applications of Artificial Intelligence 14(3): 369-376.

Lesser, V. and D. Corkill (1981). "Functionally Accurate, Cooperative Distributed Systems." IEEE Transactions on Systems, Man, and Cybernetics SMC-11(1): 81-96.

Lesser, V. R. (1995). "Multiagent Systems: An Emerging Subdiscipline of AI." ACM Computing Surveys 27(3).

Lesser, V. R. (1999). "Cooperative Multiagent Systems: A Personal View of the State of the Art." IEEE Transactions on Knowledge and Data Engineering 11(1).

Luck, M., P. McBurney and C. Preist (2003). "Agent Technology: Enabling Next Generation Computing." AgentLink(ISBN 0854 327886).

Maes, P. (1995). "Artificial Life meets Entertainment: Lifelike Autonomous Agents." Communications of the ACM, ACM Press 38(11): 108-114.

Martin, F. J., E. Plaza, J. A. Rodriguez-Aguilar and J. Sabater (1998). Java Interagents for Multi-Agent Systems. Software Tools for Developing Agents.

Mena, E., Kashyap, V., Illarramendi, A., Sheth, A. (1998). Domain Specific Ontologies for Semantic Information Brokering on the Global Information Infrastructure. Formal Ontology in Information Systems. N. Guarino. Amsterdam, IOS Press.

Muller, J. P. and M. Pischel (1993). The Agent Architecture InteRRaP: Concept and Application, DFKI Saarbrucken.

Ndumu, D. and H. Nwana (1996). "Research and Development Challenges for Agent-Based Systems." IEE/BCS Software Engineering Journal.

Neches, R. (1994). The Knowledge Sharing Effort, http://www-ksl.stanford.edu/knowledge-sharing/papers/kse-overview.html.

Neches, R., R. Fikes, T. Finin, T. Gruber, R. Patil, T. Senator and W. R. Swartout (1991). Enabling Technology For Knowledge Sharing. AI Magazine. 12: 36-56.

Noy, N. F. and D. L. McGuinness (2001). Ontology Development 101: A Guide to Creating Your First Ontology. Stanford, CA, 94305, Stanford University.

Nwana, H., L. Lee and N. Jennings (1996). "Coordination in Software Agent Systems." BT Technology Journal 14(4): 79-88.

Nwana, H. and M. Wooldridge (1996). "Software Agent Technologies." BT Technology Journal 14(4): 68-78.

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

Nwana, H. S. and D. T. Ndumu (1999). A Perspective on Software Agents Research. Ipswich, British Telecommunications Laboratories.

Nwana, H. S., D. T. Ndumu, L. C. Lee and J. C. Collis (1999). "ZEUS: A Tool-Kit for Building Distributed Multi-Agent Systems." Applied Artificial Intelligence Journal 13(1): 129-186.

Odell, J. (2000). Agent Technology - Green Paper, OMG - Agent Platform Special Interest Group.

Odell, J., M. Nodine and R. Levy (2005). A Metamodel for Agents, Roles, and Groups. Lecture Notes on Computer Science. J. Odell, P. Giorgini and J. Müller. Berlin, Springer. Agent-Oriented Software Engineering (AOSE) V.

Odell, J., H. V. D. Parunak and B. Bauer (2000). Extending UML for Agents. Proceedings of the Agent-Oriented Information Systems Workshop at the 17th National conference on Artificial Intelligence.

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

OMG (2000). Mobile Agent Facility Formal Specification.

Park, S. and V. Sugumaran (2005). "Designing multi-agent systems: a framework and application." Expert Systems with Applications 28: 259-271.

Poslad, S., P. Buckle and R. Hadingham (2000). The FIPA-OS Agent Platform: Open Source for Open Standards. Proceedings of the 5th International Conference and Exhibition on the Practical Application of Intelligent Agents and Multi-Agents, UK.

Ramchurn, S. D., D. Huynh and N. R. Jennings (2004). "Trust in multi-agent systems." The Knowledge Engineering Review 19(1).

Rao, A. S. (1996). AgentSpeak(L): BDI Agents speak out in a logical computable language. Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World.

Rao, A. S. and M. P. Georgeff (1995). BDI Agents: From Theory to Practice. Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95), San Francisco, USA.

Russell, S. and P. Norvig (2003). Artificial Intelligence: A Modern Approach, 2/E, Prentice Hall.

Sen, S. (1997). "Multiagent systems: milestones and new horizons." Trends in Cognitive Sciences 1(9).

Shintani, T., T. Ito and K. Sycara (2000). Multiple negotiations among agents for a distributed meeting scheduler. Proceedings of the Fourth International Conference on MultiAgent Systems.

Shoham, Y. (1998). Agent-oriented programming. Readings in Agents, Elsevier Science. Artificial Intelligence 60 (1993).

Sowa, J. F. (2000). Knowledge Representation: Logical, Philosophical, and Computational Foundations. Pacific Grove, CA, Brooks Cole Publishing Co.

Spyns, P., R. Meersman and M. Jarrar (2002). Data Modelling versus Ontology Engineering, ACM SIGMOD Record. 31.

Studer, R., V. R. Benjamins and D. Fensel (1998). "Knowledge Engineering: Principles and Methods." Data and Knowledge Engineering 25(1-2): 161-197.

Sycara, K. P. (1998). "Multiagent Systems." American Association for Artificial Intelligence.

Tsvetovatyy, M., M. Gini, B. Mobasher and Z. Wieckowski (1997). "MAGMA: An agent-based virtual market for electronic commerce." Journal of Applied Artificial Intelligence.

Uschold, M. (1998). "Knowledge level modelling : concepts and terminology." The Knowledge Engineering Review 13(1): 5-29.

Uschold, M. and M. Gruninger (1996). "Ontologies:Principles, Methods and Applications." The Knowledge Engineering Review 11(2): 93-136.

Uschold, M. and M. King (1995). Towards a Methodology for Building Ontologies. Workshop on Basic Ontological Issues in Knowledge Sharing" IJCAI-95.

Van de Riet, R., Burg, H., Dehne, F. (1998). Linguistic Issues in Information System Design. Formal Ontology in Information System. G. Nicola. Amsterdam, IOS Press.

Wang, X. and Y. Lespérance (2001). Agent-Oriented Requirements Engineering Using ConGolog and i*. Proceedings of the 3rd International Bi-Conference Workshop AOIS-2001, Berlin, iCue Publishing.

Weber, R. (1997). Ontological Foundations of Information Systems. Melbourne, Coopers and Lybrand.

Werkman, K. J. (1990). Multiagent Cooperative Problem-Solving through Negotiation and Sharing of Perspectives. DAI-List, http://www-2.cs.cmu.edu/afs/cs.cmu.edu/project/ai-repository/ai/pubs/lists/dai-list/dailist/006.10may90.

Wong, H. C. and K. Sycara (1999). Adding Security and Trust to Multi-Agent Systems. Autonomous Agents '99 Workshop on Deception, Fraud, and Trust in Agent Societies.

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

Wooldridge, M. (1999). Intelligent Agents, The MIT Press.

Wooldridge, M. and P. Ciancarini (2001). Agent-Oriented Software Engineering: The State of the Art. Agent-Oriented Software Engineering. P. Ciancarini and M. Wooldridge, Springer-Verlag. AI Volume 1957.

Wooldridge, M. and N. R. Jennings (1995). "Intelligent Agents: Theory and Practice." Knowledge Engineering Review 10(2).

Wooldridge, M., N. R. Jennings and D. Kinny (2000). "The Gaia Methodology for Agent-Oriented Analysis and Design." Autonomous Agents and Multi-Agent Systems Kluwer Academic Publishers(3): 285-312.

Wooldridge, M. J. and N. R. Jennings (1995). "Agent Theories, Architectures, and Languages: A Survey." Lecture Notes in Artificial Intelligence, Springer-Verlag 890.

Zambonelli, F., N. R. Jennings and M. Wooldridge (2003). "Developing multiagent systems: the Gaia Methodology." ACM Transactions on Software Engineering and Methodology 12(3): 317-370.

Zlotkin, G. and J. S. Rosenschein (1989). Negotiation and Task Sharing Among Autonomous Agents in Cooperative Domains. The Eleventh International Joint Conference on Artificial Intelligence, Detroit, Michigan.

Zlotkin, G. and J. S. Rosenschein (1996). "Mechanism Design for Automated Negotiation, and its Application to Task Oriented Domains." Journal of Artificial Intelligence 86(2): 195-244.

Chapter 4

Multi-Agent Design Information Management and Support

4.1. Introduction

Subsequent to the literature review in the areas of distributed collaborative engineering design and multi-agent systems, the development of the Multi-Agent Design Information Management and Support System (MADIS) concentrates on requirements specification and analysis, architectural system specification and prototype implementation. The starting point for this design and implementation stage of the proposed system is formed by the high-level specification of an intelligent architecture presented in Chapter two. From a technological perspective, the extensive study of software agents available in Chapter three will inform the design stage of MADIS particularly regarding agent architectures, standards, languages and environments.

Intended to support the distributed engineering design organization, MADIS should facilitate interoperation among distributed resources as well as knowledge sharing and reuse. In order to address the current needs of distributed collaborative engineering design (see Chapter two), the following steps will be pursued:

- The requirements list for a system intended to support distributed engineering design (that concluded Chapter two) will be detailed and analysed.
- The architectural model of MADIS will be designed based on the requirements specification and the identified technological supporting elements.
- The main components of the proposed architectural model will be addressed in the implementation phase by engaging emerging software technologies.

These steps closely reflect the structure of the current chapter. After the requirements definition phase, the design and implementation stage of MADIS is described followed by a set of final remarks that conclude the chapter.

4.2. Distributed Collaborative Engineering Design Requirements

The requirements gathering phase for the development of the proposed MADIS system started at the end of Chapter two. The aim of this section is to further detail and analyse each requirement already identified.

The overall objective of MADIS is to help and support multidisciplinary designers to achieve the optimal design solution in a distributed virtual environment. Table 4.1 details the main MADIS requirements analysing them and specifying some enabling technological elements.

No	Identified Requirement (in Chapter 2)	Details	Analysis	Enabling Technological Elements
R1	*The system should efficiently manage the design data, information and knowledge circulated in a distributed environment in order to support the designer in finding, accessing and retrieving the information needed in the various design stages.*	✓ Design information needs to be stored and managed by the various components supporting the system. ✓ Design information has to be readily accessible to the user through various components able to provide browse and search services.	• User Agent Systems should support the designer in finding, accessing and retrieving the information • Information (Ontology) Agent Systems should manage the information	• Software Agents • Multi-Agent Communication
R2	*The system should aid distributed and multidisciplinary design teams in establishing and maintaining cooperation through an effective communication, co-location, coordination and collaboration.*	✓ Coordination and co-location between distributed designers has to be supported by efficient sharing of the design information managed by the system. ✓ Communication and collaboration should be supported by integrated tools but also by stored information shared by distributed users.	• The information shared in the distributed environment needs to be effectively managed within a semantically enabled knowledge base • User Agent Systems should aid the collaboration process among distributed designers	• Software Agents • Ontologies • Communication Technologies
R3	*The system should offer content related support for the exchange of data, information and knowledge in order to enable knowledge sharing and reuse in a distributed design environment.*	✓ Design information databases managed by the system should provide content-related support. ✓ Sharing and reuse of design information needs to be supported by various components of the system enabling semantic approach to browse and search services.	• Information should be stored using ontologies in order to enable semantics • Communication among agents should be supported by the ontology library	• Ontologies • Multi-Agent System
R4	*The system should address the integration of heterogeneous software tools used by designers to support the flow of information within the distributed environment.*	✓ The system should have different components integrated in the various applications used by the designer in order to extract and store information. ✓ The system should enable the flow of information by making the application design information readily available in the distributed environment.	• Application Agent Systems should autonomously capture design information from the software applications regularly used by the designer • The information captured should be stored in the ontological instances readily available to distributed designers	• Software Agents • Ontologies

Table 4.1. MADIS Requirements

It should be noted that the list of requirements for MADIS was generated based on the information needs of the distributed collaborative engineering design domain and its problematic aspects. Moreover, the current trends in software support for distributed design (see Chapter two) certainly influenced the development of the proposed MADIS system, hence the initial list of requirements.

MADIS aims to support the distributed design process by managing information, integrating resources dispersed over a computer network and aiding collaboration processes. It is intended to design a multi-agent system composed of several interacting agent sub-systems in order to deliver these aims. Furthermore, the information circulated within the distributed design environment will be stored in an ontology library to enable content-related support for information management. These technical choices were mainly motivated by the following factors:

- The engineering design domain is inherently distributed and heterogeneous making autonomous software agents a promising solution for computational support.
- The information resources involved in the engineering design process are heterogeneous and distributed.
- The human designers involved in the engineering design process are multidisciplinary and dispersed over computer networks.

The multi-agent approach to distributed engineering design coupled with the use of ontologies promises to tackle important distributed design issues such as interdisciplinary cooperation among distributed designers, exchange of design data, information and knowledge and integration of heterogeneous software tools.

4.3. MADIS Architectural Design

The goals of the architectural design phase of MADIS are (i) to identify the kinds of agents needed to deliver the requirements of the system, (ii) to specify the interconnections between them and (iii) to design the ontology library supporting the overall system. Based on the literature review presented in Chapter three and the research to date (Odell, Parunak et al. 2000; Bauer, Müller et al. 2001; Odell, Nodine et al. 2005), Agent UML (AUML) has been selected as the methodology to support the MADIS design phase. Promoted by both FIPA[1] and OMG[2] agent standard organizations, AUML uses a set of UML[3] idioms and extensions to support the modelling of agents and agent-based systems (Odell, Parunak et

[1] Foundation for Intelligent Physical Agents (http://www.fipa.org)
[2] Object Management Group (http://www.omg.org)
[3] Unified Modelling Language (http://www.uml.org)

al. 2000; Bauer, Müller et al. 2001). Furthermore, FIPA (http://www.fipa.org) has been selected as the agent standard to support the MADIS design and development (see section 3.4 from Chapter three for a literature review on agent standards).

MADIS employs agents for information storage and retrieval, for enhancing collaboration within a distributed design environment and for providing a suitable interface to the user. The efficient performance of these tasks is ensured by the cooperation process among the different kinds of agents that form MADIS. These agents can be divided in four societies as follows (see Figure 4.1):

1. User management
2. Application management
3. Ontology management
4. Agent interconnection and management

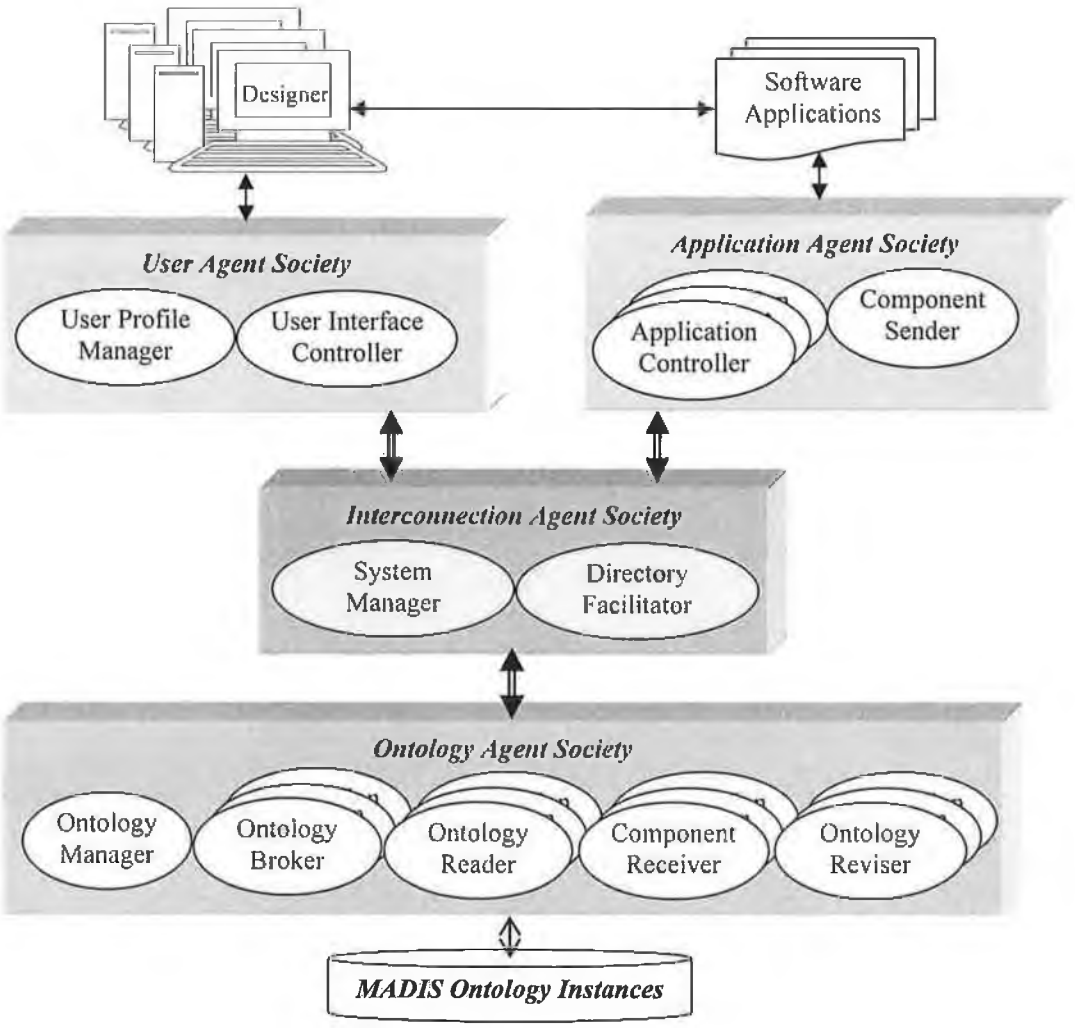


Figure 4.1. MADIS agent society

The agents from the *user society* form the interface between MADIS and the designer. They provide different services to the user and respond to queries and events initiated by the user (or on behalf of the user) with the help of the ontological agents. The agents from the *application society* are in charge of retrieving information from the applications used by the designer and forward it for storage to the ontological agents. They should be integrated in the software tools regularly used in the distributed engineering design domain and act autonomously pursuing their objective (i.e. information retrieval). The agents from the *ontology society* provide ontology management services. They are able to access, retrieve, add, modify and delete information from the ontology library. The agents from the *interconnection society* manage the cooperation process among other agents based on the needs and the services advertised by them.

The FIPA agent management ontology is part of each MADIS agent expertise so as to support agent interoperation (see section 4.3.5). The agent communication language used in MADIS is FIPA ACL, based on which MADIS agents are able to exchange messages (of types such as request, query, and inform) in order to achieve different objectives. Furthermore, the MADIS ontology (see section 4.3.3) completes the expertise of the basic MADIS agents.

The next step of the architectural design phase is to identify the organizational structures for each society presented above. The agent(s) specific to each society will be described using the AUML concept of Agent Class Diagram (Bauer 2001). The agent class diagram specifies role(s), state description, actions, methods, capabilities, service description and supported protocols (Bauer 2001). Furthermore, agent interoperation within MADIS will be described using AUML protocol diagrams, which are diagrams able to model protocols for multi-agent interaction (Bauer, Müller et al. 2001).

4.3.1. The User Agent Society

Each human user of the MADIS system will have a personal agent responding to specific needs and providing required services according to the user profile. This is achieved internally by employing two agents that will collaborate closely:

- The *User Profile Manager* agent stores and manages the user description and preferences based on which various MADIS services are offered. This agent should learn from the user as to improve the functionality of the system over time.
- The *User Interface Controller* agent directly assists the user in his/her tasks through a graphical interface. This agent enables user access to different MADIS services

mainly based on the cooperation with the User Profile Manager agent and the ontology agent society.

The *User Interface Controller* agent is responsible for providing the interface between the user and the MADIS system. Through this agent, the user should have access to the design information needed for the task at hand. Figure 4.2 presents the AUML class diagram of the User Interface Controller agent.

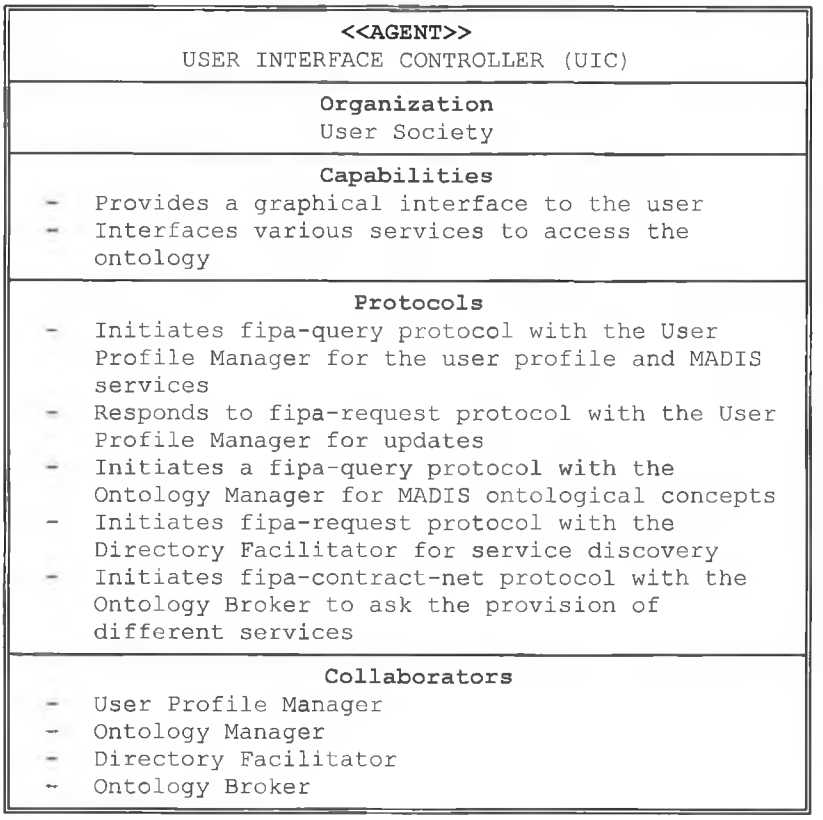


Figure 4.2. The User Interface Controller agent class diagram

The following cooperation processes ensure the optimum functionality of the User Interface Controller agent for achieving its objective of continuously supporting the user:

- *User Profile Manager*: this cooperation process provides and updates the user-preferred services in the desired format.
- *Ontology Manager*: facilitates the discovery of the conceptual schema of MADIS. The services available to the user can be applied for each ontological concept received from the Ontology Manager.
- *Directory Facilitator*: helps the User Interface Controller to find and invoke the agents advertising the services requested by the user.
- *Ontology Broker*: the User Interface Controller communicates with the Ontology Broker to actually request the service.

These agent interactions are performed using ACL messages within FIPA request, query and contract-net protocols.

4.3.2. The Application Agent Society

The information circulated in the computer-defined distributed design environment is usually available in the various proprietary applications used by designers to support their tasks. In order to make this heterogeneous information readily available to distributed users and semantically integrate dispersed resources, each application is controlled by MADIS through an Application Controller agent. This agent has to be integrated[4] in the application served and forward all the information that can be extracted using the API to the ontology agent society for storage purposes. The Application Controller agents can act autonomously to achieve their objectives or can be controlled by the user (through the User Profile Manager) who can set different functional parameters if desired. Depending on the flexibility of the specific API, each Application Controller agent can extract information about a number of different conceptual structures from the current application (e.g. assembly structural information from a Catia system) and transform it into MADIS internal format. For each component, an autonomous Component Sender agent will actually manage the process of forwarding the extracted information to the ontology agent society (the exact agent from this society will be identified through a cooperation process with the Directory Facilitator) who will store the given component in the pre-defined ontological format. Therefore, the application agent society is mainly composed of the following agents:

- The *Application Controller* agent is integrated in a software application and is responsible for extracting available information (there should be as many Application Agents as software applications engaged in the distributed design process).
- The *Component Sender* agent acts without any user interaction to achieve its objective of intercepting and sending information from an Application Controller agent to a specific agent from the ontology society that knows how to save the received information.

The ontology agent society (that will be presented in section 4.3.3) has to include specific agents that know how and where to save the different design components in order to support the MADIS application agents.

[4] This integration process should be facilitated by the Application Programming Interface (API) of the design application.

Figure 4.3 presents the AUML class diagram of the Application Controller agent. Using FIPA ACL performatives, the Application Controller agent cooperates with the following agents to achieve its objectives:

- *Component Sender*: this cooperation process is necessary to forward a specific component for storage purposes.
- *User Profile Manager*: the user-desired functioning parameters (which can be optionally set by the user) are obtained from the User Profile Manager agent.

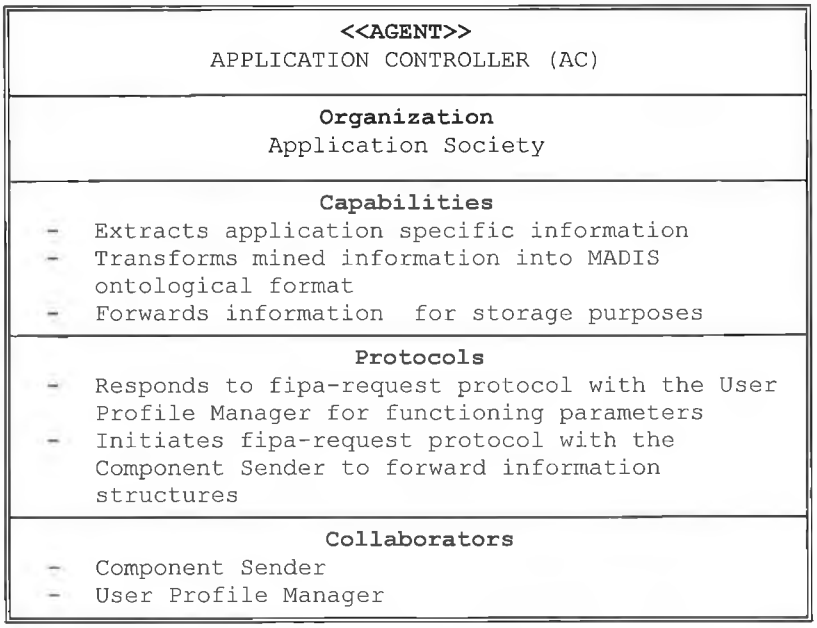


Figure 4.3. The Application Controller agent class diagram

When requested to send a component for ontology storage, the Component Sender identifies the agent who can provide the service of saving information in the ontology through the Directory Facilitator. A Component Receiver agent from the ontology society is located and asked by the Component Sender to perform the required task. The AUML diagram of the Component Sender agent is presented in Figure 4.4. Using FIPA ACL and FIPA request, inform and contract-net protocols, the Component Sender cooperates with the following agents:

- *Application Controller*: this cooperation process sets the objective of the Component Sender.
- *Directory Facilitator*: facilitates the discovery of the required service.
- *Component Receiver:* the objective is achieved by sending information to the correct Component Receiver agent that can add it to the MADIS ontology.

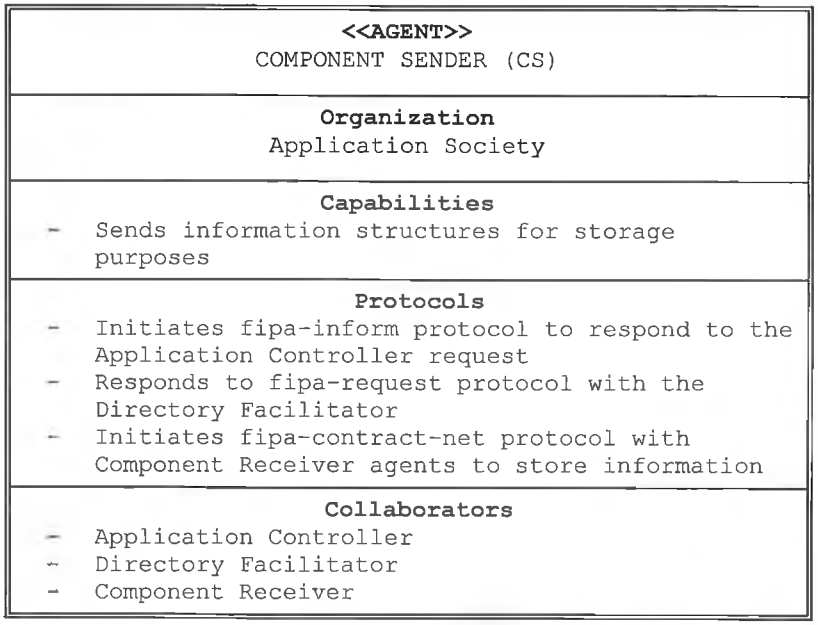


Figure 4.4. The Component Sender agent class diagram

Extracting and saving information structures from various design software tools to MADIS, the application agent society addresses the need for heterogeneous application integration within a distributed engineering design environment.

4.3.3. The Ontology Agent Society

Creating semantic link among the MADIS architectural components, the MADIS ontology describes concepts, relationships and inference rules of the engineering design domain. The development of this engineering design ontology started during the design phase of the multi-agent architecture and was a continuous process throughout the implementation phase of the multi-agent system.

The methodology used to develop the MADIS ontology for the engineering design domain was inspired by the Methontology approach (Fernandez, Gomez-Perez et al. 1997; Fernandez-Lopez, Gomez-Perez et al. 1999; Gomez-Perez 1999) proposed by the Polytechnic University of Madrid. Based on the IEEE 1074-1995 standards (Fernandez-Lopez 2001) for Developing Software Life Cycle Processes (IEEE96 1996), the Methontology framework includes a development-oriented process referring to specification, conceptualization, formalization, implementation and maintenance activities (Fernandez, Gomez-Perez et al. 1997). This process is closely followed in the MADIS ontology design and implementation.

The scope of the MADIS engineering design ontology is to create a common shared understanding of the application domain so that information and knowledge can be shared among the members of the distributed design environment. These members can be humans

or software agents. The ontology aims to establish a joint terminology between these members. A more detailed view of the MADIS ontology scope is as follows:

- To support agent interoperations as well as human-to-agent and agent-to-human interactions.
- To support knowledge management activities (see Table 4.2).
- The main *intended users* of the ontology are engineering designers (probably through their personal User Interface Controller agent). Because the engineering design terminology will probably have common parts with the ontologies corresponding to other product life cycle phases, other possible users include the manufacturers, the suppliers, the users of the product and the EOL people.
- The inputs consist of the formal and formalizable concepts of the engineering design domain and data structures from the design software tools.
- The ontology will not perform any processing but it will describe the processing environment e.g. concepts (environment objects), relationships between concepts, axioms, inference rules.

To summarize, the MADIS ontology aims to formally conceptualise the engineering design domain in order to allow knowledge sharing, reuse and integration in a distributed design environment.

Knowledge Management Activity	**MADIS Ontology Scope**
Gathering (capturing)	The ontology is intended to provide a common framework for all distributed design participants (humans and agents).
Organization (structuring, storing)	The ontology will define engineering design concepts and describe their meanings so that a common understanding of the domain is achieved.
Refinement	Software and/or human agents will use the ontology (understanding) to correct, update, add, and delete knowledge, i.e. to maintain knowledge.
Distribution (sharing, dissemination)	One of the main purposes of the ontology is to share concepts and meanings to all distributed design participants that commit to this ontology.
Using	The ontology keeps the domain knowledge in a standard uniform format to enable distributed design participants to use data, information and knowledge in the same way traditional software tools are using data from databases.

Table 4.2. MADIS Ontology Scope relative to various knowledge management activities

Intended to incorporate as many concepts of the engineering design domain as possible, the MADIS ontology actually refers to a library of specific information ontologies focusing on various aspects of the targeted domain (see Figure 4.5). The ontological instances contained within this library can be distributed in different locations across the enterprise. A *Generic Design Ontology* resides at the top-level of the MADIS ontology library. It introduces and defines the main concepts of the distributed design domain e.g. space, time,

activity, process and artefact. The other ontologies are specializations of the Generic Design Ontology. Figure 4.5 depicts some of the ontologies used in MADIS.

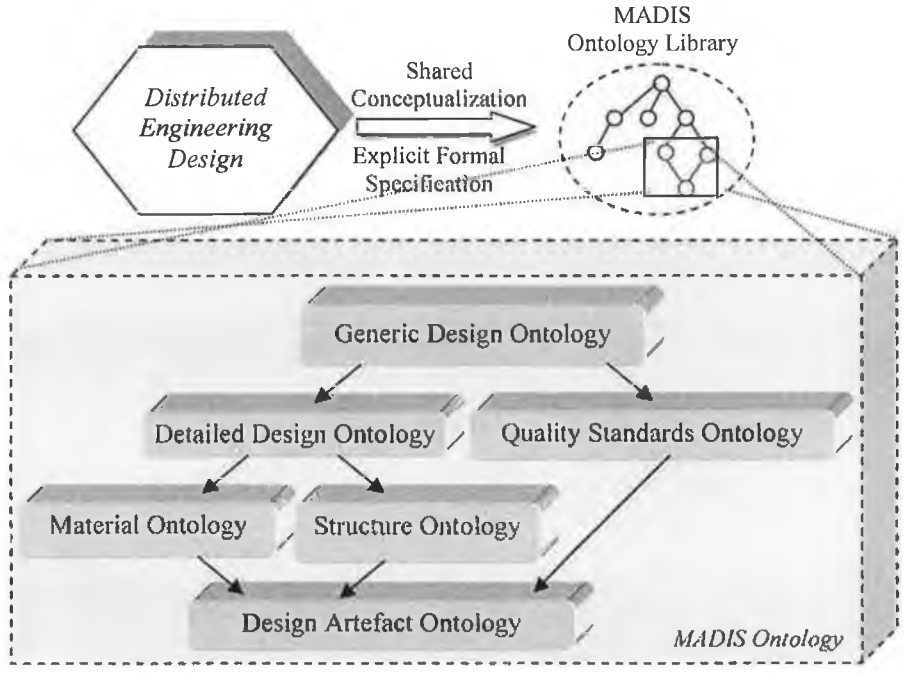


Figure 4.5. MADIS Ontology

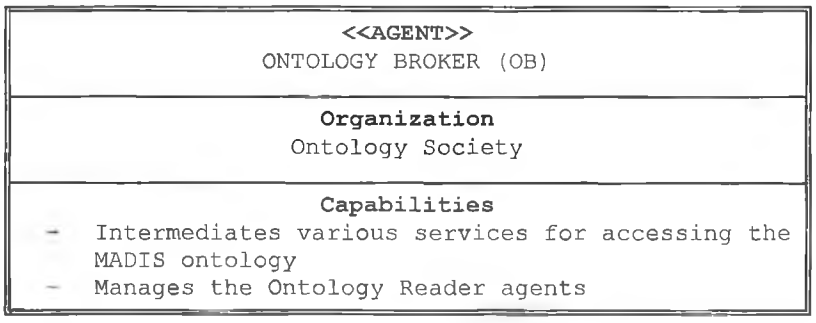
The *Detailed Design Ontology* formalizes general concepts specific to the detailed design phase. The *Quality Standards Ontology* defines the various quality standards and techniques (e.g. ISO9000, FMEA – Failure Mode Effect Analysis, TQM – Total Quality Management) that might be used throughout the design process, so that the artefact will adhere to certain quality standards. The *Material Ontology* defines concepts and relations about the material properties relating to an artefact (e.g. material type, material ID, ductility, malleability, thermal conductivity and density). The *Structure Ontology* describes the relationships between the components of the artefact (e.g. fasteners, assembly/disassembly times, routes and tools). Both the Material and Structure Ontologies are specializations of the Detailed Design Ontology. The *Design Artefact Ontology* is a further specialization of ontologies such as the Material Ontology and the Structure Ontology and will describe the various design parameters of the artefact. The implementation of these ontologies using specific models and ontology engineering languages will be presented in section 4.4.3.

The *ontology agent society* contains different kinds of agents able to maintain (e.g. add, delete, modify) the information structures stored in the MADIS ontology. These agents can be classified as follows:

- The *Ontology Manager* agent supervises the ontology management process ensuring the consistency of the ontology is accurate and that requested ontology-related services are delivered.
- The *Ontology Broker* agent manages the agents that can read the ontology (i.e. the Ontology Reader agents) and the services provided by them.
- The *Ontology Reader* agents extract and forward information under certain conditions from the MADIS ontology to the requester agent.
- The *Component Receiver* agents have the capability of adding new instances of specific concepts (or components) to the corresponding ontology from the MADIS ontology library.
- The *Ontology Reviser* agents can update the ontology by deleting or modifying existing information.

The *Ontology Manager* agent represents the core of this MADIS agent society controlling the behaviour of the other ontology dedicated agents. The consistency of the MADIS ontology is periodically checked to assure the absence of any contradictory information and to verify that all inference rules are satisfied. Furthermore, the Ontology Manager has knowledge of all the services that the other agents of the society can provide. This means that the Ontology Manager is able to respond to different ontology-related requests by invoking the particular agent that knows how to address the specific request. Finally, the Ontology Manager ensures that each agent has a complete copy of the latest version of the MADIS ontology so that users are provided with up-to-date design information.

The *Ontology Broker* agent manages the Ontology Reader agents and responds to requests related to information retrieval from the ontology. Knowing the capabilities of the agents that can read the ontology, the Ontology Broker registers with the Directory Facilitator all the services provided by the Ontology Reader agents. When an agent requests a service advertised, the Ontology Broker can instantiate and activate the correct Ontology Reader agent that knows how to deliver the requested service. Figure 4.6 presents the AUML class diagram of the Ontology Broker.



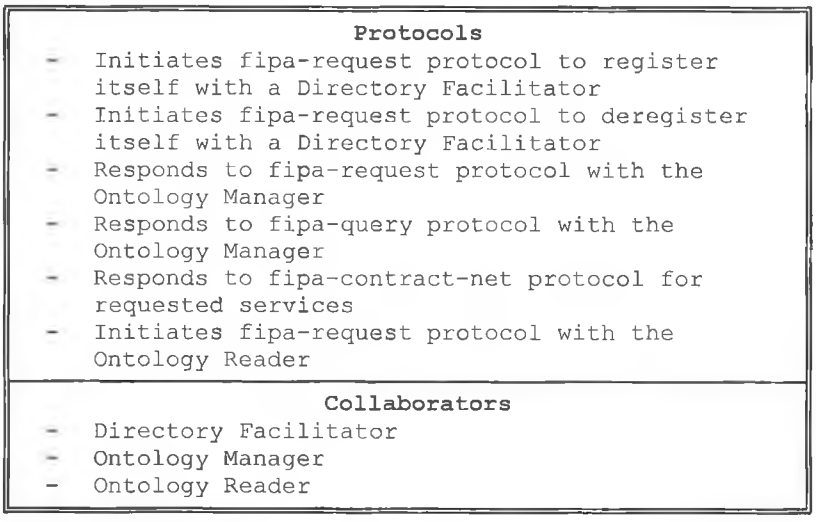
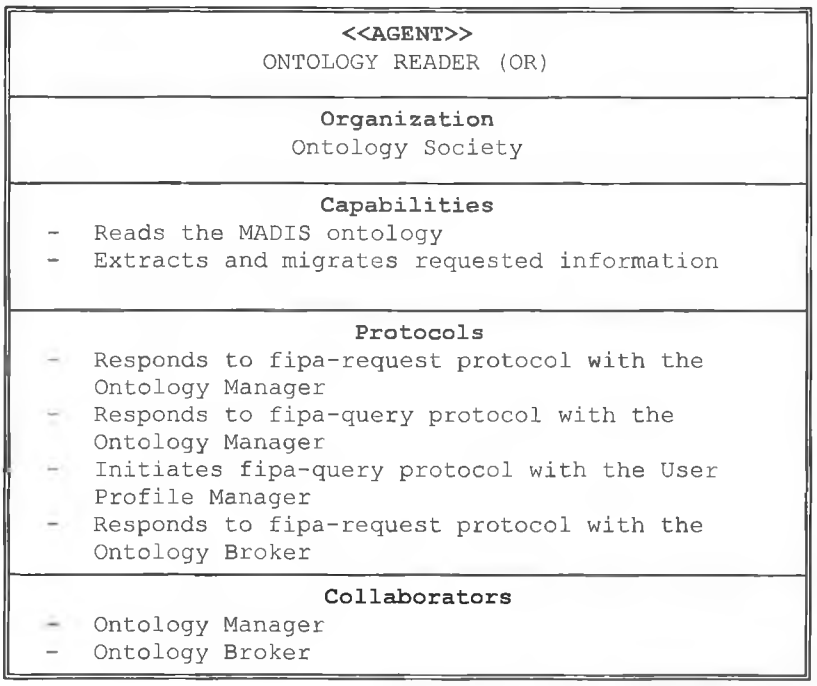


Figure 4.6. The Ontology Broker agent class diagram

The *Ontology Reader* agents are mobile agents able to read the ontology and to arrange the mined information in graphical format. After the user interface containing the requested information has been created, the Ontology Reader migrates to the initial requester agent (e.g. a User Interface Controller agent). There are two major examples of Ontology Reader agents as follows:

- An Ontology Reader that extracts from the specific MADIS ontology all instances of a given concept and forwards them to the requester (e.g. response to a browse service requested).
- An Ontology Reader that extracts from the MADIS ontology only the instances that satisfy a given query (e.g. response to a search service requested).

Figure 4.7 presents the AUML diagram of the Ontology Reader agent.



- User Profile Manager

Figure 4.7. The Ontology Reader agent class diagram

The *Component Receiver* agent can update the MADIS ontology by adding new instances of various existing ontological concepts. This action is performed as a response to update requests made by the other MADIS agents particularly the Component Sender agents from the application agent society. The AUML class diagram of the Component Receiver agent is presented in Figure 4.8.

<<AGENT>> COMPONENT RECEIVER (CR)
Organization Ontology Society
Capabilities - Adds new ontological instances in MADIS
Protocols - Responds to fipa-request protocol with the Ontology Manager - Responds to fipa-query protocol with the Ontology Manager - Initiates fipa-request protocol to register itself with a Directory Facilitator - Initiates fipa-request protocol to deregister itself with a Directory Facilitator - Responds to fipa-contract-net protocol for requested services (e.g. by Component Sender)
Collaborators - Ontology Manager - Directory Facilitator - Component Sender

Figure 4.8. The Component Receiver agent class diagram

The number of active Component Receiver agents may increase as the size of the MADIS ontology library (and therefore, the number of defined concepts) grows. Furthermore, the Component Receiver collaborates with the Ontology Manager to ensure proper functionality and with the Directory Facilitator to register the provided services once active.

The *Ontology Reviser* agents form another class of agents in the ontology agent society. They are able to update the MADIS ontology through delete and modify types of actions. These services are registered through the Directory Facilitator and performed upon request. A collaboration process with the Ontology Manager will establish exactly which ontology is the one where the alterations should be applied. This class of agents exists mainly because of maintainability issues as it is probably better (and recommended in some

environments) to keep saved design information and manage it through a revision control system rather then remove it completely (or update existing information).

4.3.4. The Interconnection Agent Society

The interconnection society contains agents that supervise and support the interoperation process among the other MADIS agents. The main objective of this agent society is to ensure that MADIS agents are meaningfully interconnected. This is achieved through the following types of agents:

- The *System Manager* agent supervises the overall functionality of the multi-agent system.
- The *Directory Facilitator* agent helps MADIS agents to find the agent(s) that provides a requested service.

Central to the MADIS agent management, the System Manager[5] has the capability of supervising the MADIS multi-agent environment by controlling the state of each MADIS agent. All MADIS agents must register with the System Manager in order to allow efficient operation management of the multi-agent system. Based on the FIPA specifications (http://www.fipa.org), the System Manager must be able to perform functions such as register, deregister, modify, search and get-description. Furthermore, the System Manager has the capability to execute the actions such as suspending an agent, terminating an agent, creating an agent, resuming agent execution, invoking an agent, executing an agent and managing resources (http://www.fipa.org). The existence of a System Manager agent underpins the optimum functionality of MADIS and consequently provides the desired system robustness.

Compliant with the FIPA specifications, the Directory Facilitator provides a Yellow Pages service to the MADIS agent community. Any MADIS agent can use the Directory Facilitator to find other agents providing the services he requires in order to achieve his goals. Hence, there are two main facilities supported by Directory Facilitator as follows (see Figure 4.9):

1. Agents can register (and deregister) their services with the Directory Facilitator.
2. Agents can query the Directory Facilitator to find out which agent or agents (if any) offer a requested service.

[5] The System Manager corresponds to the Agent Management System (AMS) described in the FIPA specifications. AMS is a mandatory component of a FIPA compliant agent platform.

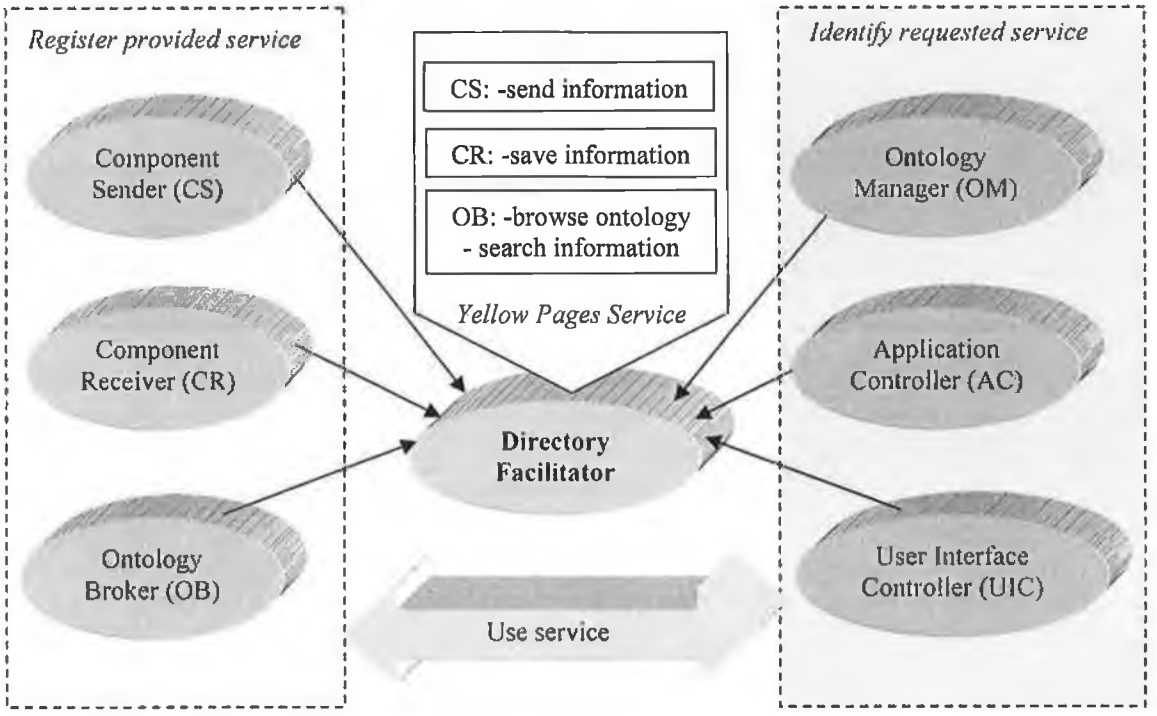


Figure 4.9. The Directory Facilitator within MADIS

Figure 4.9 exemplifies how the Directory Facilitator supports the performance of other agents. Based on the published services (the CS, CR and OR registered their services with the Directory Facilitator), the User Interface Controller is able to identify the Ontology Broker as the agent that is able to provide the service of searching the ontology and has the option of establishing a direct cooperation process in order to achieve the current objective.

4.3.5. Agent Interoperation

The agent interactions within MADIS are vital for a successful and constructive support provided to the distributed designers. As already indicated, MADIS agents are FIPA compliant and communicate by exchanging ACL messages. According to the FIPA international standard for agent interoperability (http://www.fipa.org), the structure of any ACL message exchanged contains the following parameters:

- The FIPA ACL performative of the message (type of communicative act) e.g. REQUEST, INFORM, QUERY.
- The sender of the message.
- The receiver of the message.
- The actual content of the message.

- The content language used to express the content of the message (communication will be effective if both the sender and the receiver are able to encode and parse expressions using the syntax of the content language).
- The ontology used to define the concepts present in the content of the message (communication will be effective if both the sender and the receiver commit to this ontology).

The content language used in the ACL message is FIPA SL and the FIPA-agent-management ontology supports agent interoperation. Furthermore, all MADIS agents commit to the MADIS ontology (described in section 4.3.3).

Supported by FIPA ACL, the foremost MADIS agent interactions can be summarized as follows:

- The *User-Request-Information* scenario implies agent interoperation involving the User Interface Controller, the User Profile Manager, the Ontology Broker, the Ontology Reader and the Directory Facilitator agents.
- The *Application-Save-Information* scenario implies agent interoperation among the Application Controller, the Component Sender, the Component Receiver and the Directory Facilitator agents.

Each of these scenarios will be described next using AUML protocol diagrams (Bauer, Müller et al. 2001). Extending UML state and sequence diagrams, AUML protocol diagrams include elements such as agent roles, agent lifelines, threads of interaction, connectors, conditions, cardinality, nested and interleaved protocols (Bauer, Müller et al. 2001; Huget 2002).

The *User-Request-Information* scenario occurs each time the user wants to browse or to search the MADIS ontological instance base. Having the MADIS environment set up on his/her computer, the user can request information through a personal agent managed by the User Interface Controller. The User-Request-Information scenario involves the following main steps (see Figure 4.10 for the AUML protocol diagram):

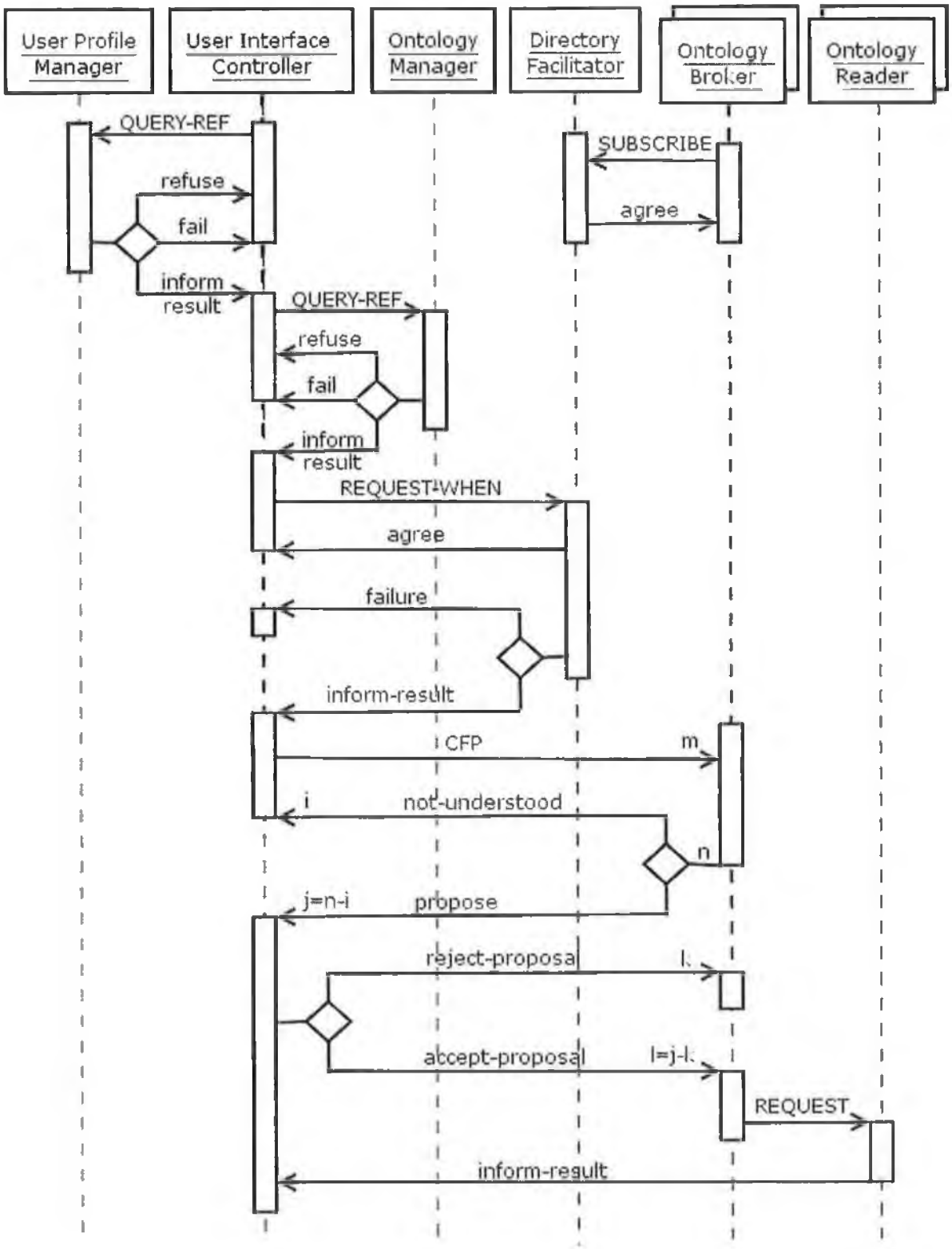


Figure 4.10. The User-Request-Information AUML interaction protocol diagram

1. The User Interface Controller queries the User Profile Manager for the services provided to the user through a FIPA-QUERY protocol.

2. The User Interface Controller queries the Ontology Manager for the concept categories available in the ontology that can be accessed by the user. The FIPA-QUERY protocol rules the agent interoperation.
3. The User Interface Controller requests the Directory Facilitator the identification of the agent that can provide the service requested by the user (i.e. Ontology Broker). The FIPA-REQUEST protocol rules the agent interoperation.
4. The User Interface Controller requests the Ontology Broker (identified in the previous step) for the service (e.g. browse, search) needed by the user. The FIPA-CONTRACT-NET protocol rules the agent interoperation.
5. The Ontology Broker instantiates the appropriate Ontology Reader mobile agent that will fulfil the requested service and will migrate back to the User Interface Controller location with the result. The FIPA-REQUEST protocol rules the agent interoperation.

The *Application-Save-Information* scenario occurs when information extracted from a design software application (that is being served by a MADIS Application Agent) is saved to the MADIS ontology. An Application Controller agent has to be integrated in the design software application and must communicate with a Component Sender agent to forward the extracted information for storage purposes. This process can occur autonomously or can be triggered by the user based on the settings saved by the User Profile Manager (the cooperation process between the Application Controller and the User Profile Manager will be omitted from this scenario for simplicity reasons). Figure 4.11 presents the AUML protocol diagram for the Application-Save-Information scenario. The main steps are as follows:

1. The Application Controller transforms extracted information into MADIS format and requests the Component Sender to transmit it for ontology storage. The FIPA-REQUEST protocol rules the agent cooperation.
2. The Component Sender requests the Directory Facilitator the identification of the agent that can provide the service of saving information in the ontology. The FIPA-REQUEST protocol rules the agent interoperation.
3. The Component Sender requests the identified Component Receiver to store the information in the ontology. The FIPA-CONRACT-NET protocol rules the agent cooperation.

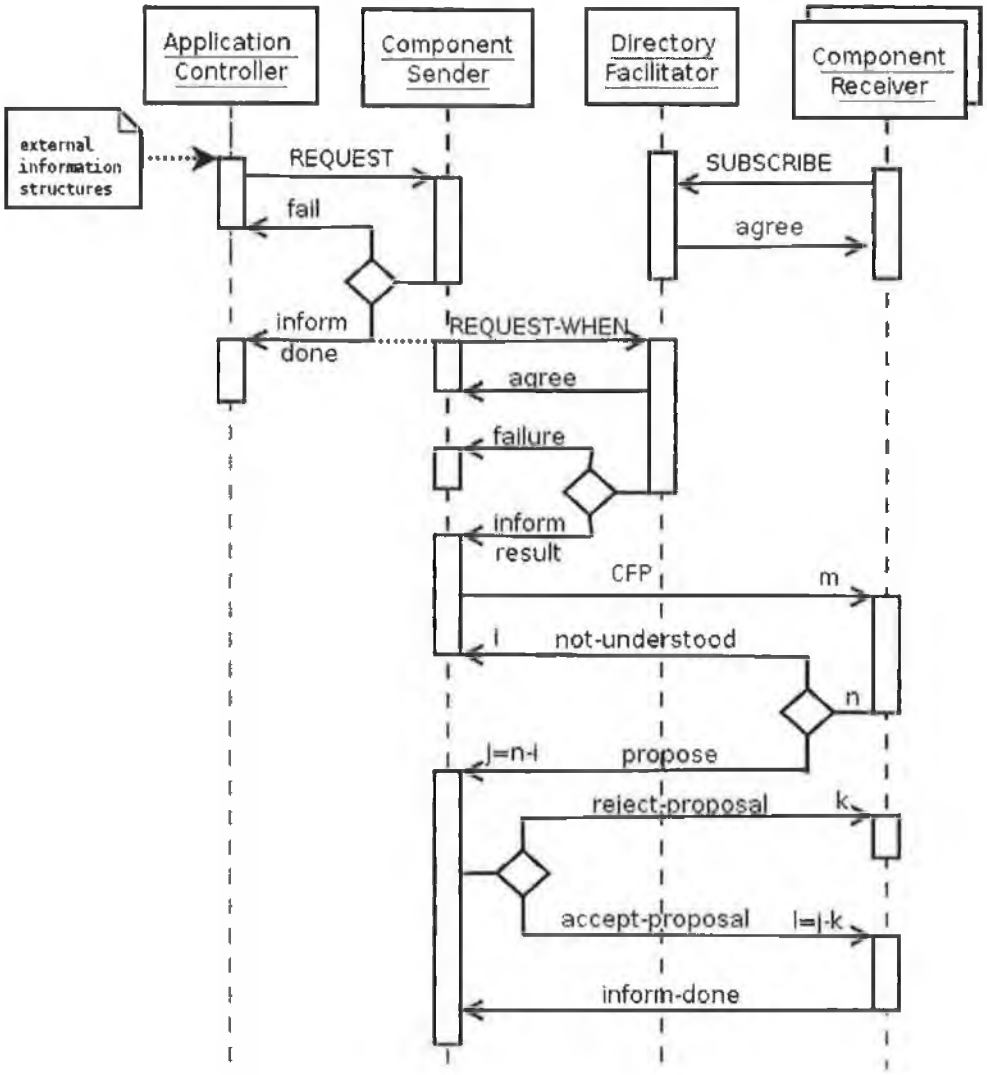


Figure 4.11. The Application-Save-Information AUML protocol diagram

The design phase of the MADIS multi-agent system also includes AUML protocol diagrams for other secondary MADIS scenarios (e.g. user profile updated, ontology revision) similar at a conceptual level with the ones presented above.

4.3.6. Summary

Supported by the AUML and FIPA specifications, the MADIS design phase identified the multi-agent infrastructure necessary to accomplish the requirements of a system intended to support distributed engineering design. Table 4.3 summarizes the MADIS agents described in this section.

Agent society	Agent	Objectives	Properties
User agent society	User Profile Manager (UPM)	• Organize user profile • Learn user preferences	Learning Autonomy Cooperation Pro-activeness
	User Interface Controller (UIC)	• Provide services to the user • Create GUI based on	Cooperation Semi-autonomy

		cooperation with UPM	Pro-activeness
Application agent society	Application Controller (AC)	• Extract information from a design software application • Transforms information into MADIS format • Forward information fro storage purposes	Autonomy Cooperation Pro-activeness
	Component Sender (CS)	• Send information for ontology storage	Autonomy Cooperation Pro-activeness
Ontology agent society	Ontology Manager (OM)	• Supervise ontology management • Check ontology consistency • Cooperate with the other agents to ensure they have a complete copy of the ontology	Autonomy Cooperation Pro-activeness
	Ontology Broker (OB)	• Provide information retrieval services • Supervise the OR agents	Cooperation Pro-activeness
	Ontology Reader (OR)	• Read the ontology in a specific way • Create a GUI containing the information read • Migrate through the network	Mobility Cooperation Pro-activeness
	Component Receiver (CR)	• Receive information that needs to be stored • Save information in the ontology	Autonomy Cooperation Pro-activeness
	Ontology Reviser (OV)	• Update ontology	Cooperation Pro-activeness
Interconnection agent society	System Manager (SM)	• Supervise MADIS agents • Manage the agent platform	Autonomy Cooperation Pro-activeness
	Directory Facilitator (DF)	• Provide Yellow Pages service to MADIS agents	Autonomy Cooperation Pro-activeness

Table 4.3. MADIS agents

All MADIS agents should be able to take the initiative (i.e. pro-activeness) and interoperate (i.e. cooperation) with other agents in order to achieve their objectives. Moreover, autonomy is a desired MADIS property as some of the MADIS agents should be able to operate on their own without the intervention of users or other agents e.g. User Profile Manager, Application Controller, Ontology Manager, System Manager.

Figure 4.12 presents a possible deployment of the MADIS agents in a distributed engineering design environment.

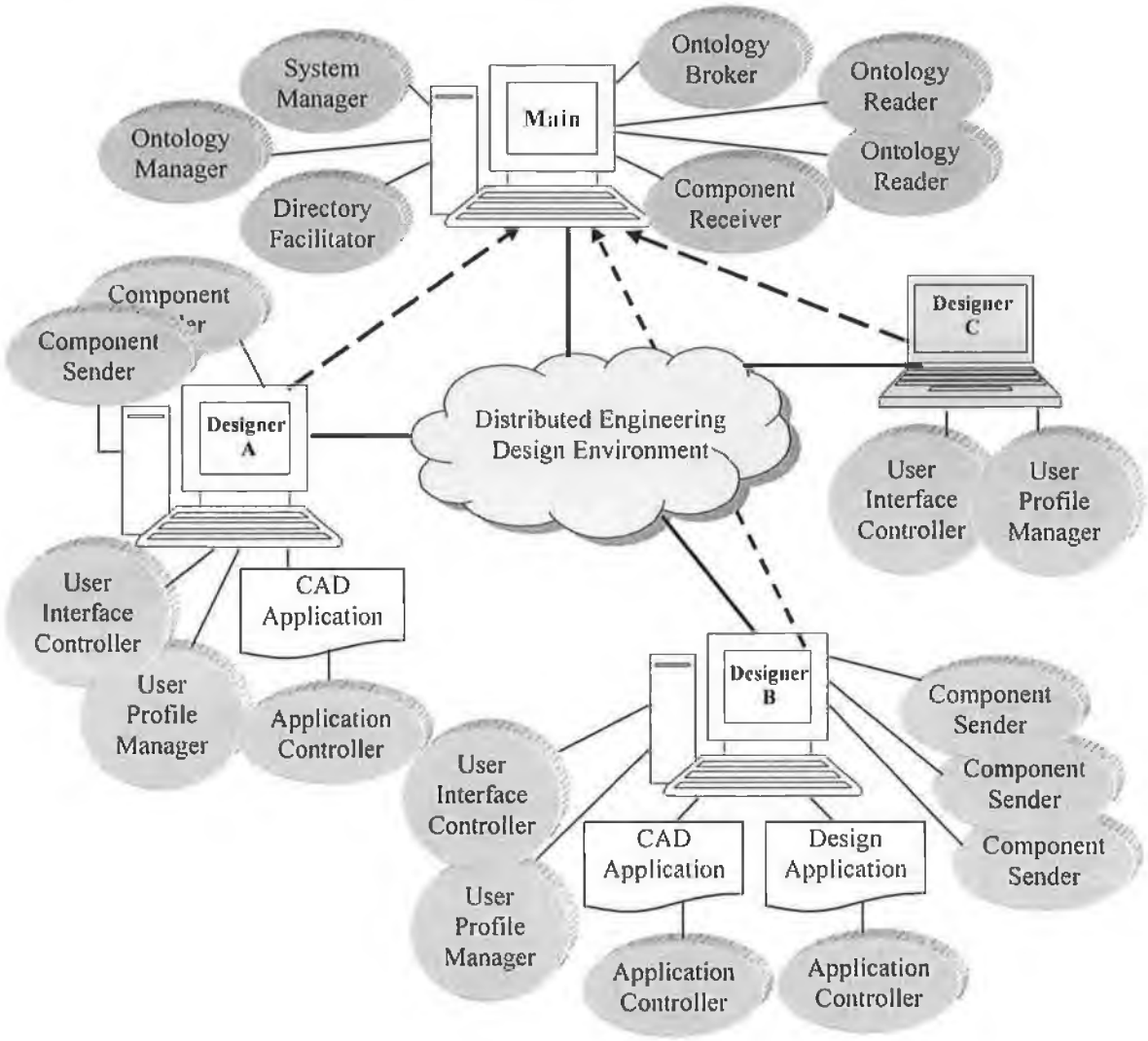


Figure 4.12. MADIS operation

The computer labelled 'Main' in Figure 4.12 represents the main platform containing the MADIS manager agents e.g. System Manager, Ontology Manager that supervise the entire agent interoperation process. The other computers in the network are used by different designers each served by a User Interface Controller agent and a User Profile Manager agent (that will have to register with the System Manager and optionally with the Directory Facilitator). Furthermore, some designers have one or more Application Controller and Component Sender agents active depending on the number of software applications integrated in MADIS (e.g. the information handled by Designer A using a CAD application is also organized by an Application Controller agent).

4.4. MADIS Implementation

The aim of the implementation phase is to provide a working prototype model of MADIS that can exemplify and demonstrate the purpose and validity of the system and that can be

analysed and evaluated in the testing and validation phase. Therefore, MADIS implementation will address most (but not necessarily all) components identified in the design phase. Nevertheless, the implementation phase will not exclude any components without which the notion of agency would become useless for distributed engineering design (e.g. the implementation of agent properties such as autonomy, cooperation and pro-activeness).

Using the architectural design elements described in the previous section as inputs, the implementation phase of MADIS commenced by establishing the agent language and environment that (in the author's opinion) can best support the development of the proposed multi-agent system.

The programming language selected for implementation is Java (http://java.sun.com) due to its rich library of functions tackling concurrency as well as security (Huget 2002), support for object-oriented programming techniques, code portability, exception and event handling capabilities, native support for multithreading and introspection of object properties and methods (Bigus, Schlosnagle et al. 2002; Zambonelli, Jennings et al. 2003). Being a portable language, Java enables MADIS agents to run on any Java-enabled platform (e.g. Microsoft Windows, Linux, Unix, Solaris). Furthermore, the Java Agent DEvelopment Framework (JADE)[6] (http://jade.cselt.it) enables the implementation of agent interoperation within MADIS. Compliant with the FIPA specifications, JADE is a software framework fully implemented in Java that facilitates the development of multi-agent systems. JADE supports scheduling of cooperative behaviours and implements the full FIPA communication model integrating all its components e.g. interaction protocols, envelope, ACL, content languages, encoding schemes, ontologies and transport protocols. All agent communication is performed through asynchronous message passing (using FIPA ACL to represent messages). JADE supports the development, debugging and deployment phases for multi-agent systems by graphically facilitating the remote management, monitoring and controlling of the status of agents, the creation and execution of an agent on a remote host as well as control of other FIPA compliant agent platforms. Implementing an agent as a Java thread, JADE exploits Java features such as Object Serialization, Reflection API and Remote Method Invocation (RMI).

This section presents the implemented MADIS agents grouped under the four agent societies (i.e. Interconnection, User, Application, Ontology) identified during the design phase. Furthermore, a fifth component (i.e. MADIS Web Portal) that makes the MADIS

[6] An extensive literature review of agent environments, toolkits and frameworks is available in Chapter three.

information structures available in a web format has been added to the implemented MADIS system. The ontology-related implementation strategies and technologies will be presented in the sub-section referring to the ontology agent society. The MADIS implementation phase also requested some secondary technical decisions which will be discussed during this section whenever necessary.

4.4.1. Interconnection Agents

In the current implemented prototype, MADIS relies on the JADE environment to support agent interconnection. The agent platform conforms to the standard model defined by FIPA (see Figure 4.13).

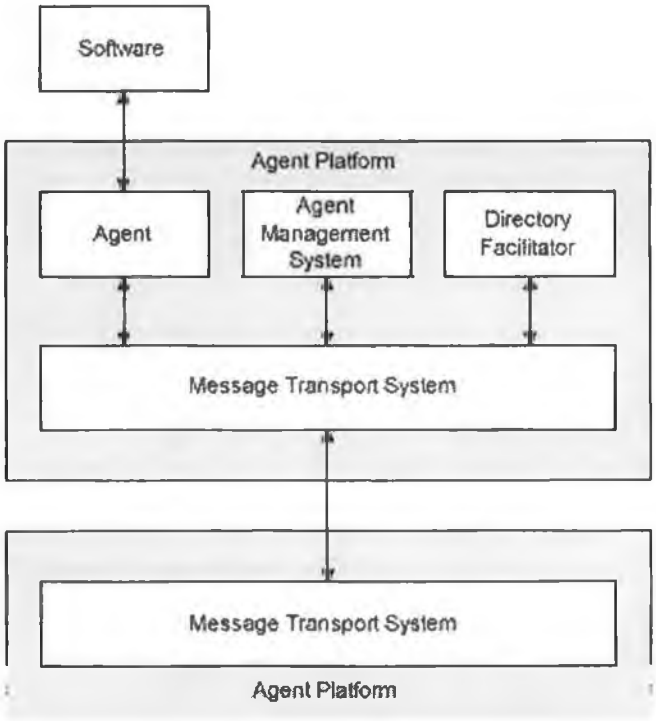


Figure 4.13. The FIPA agent platform (http://www.fipa.org)

The System Manager agent described in the MADIS design stage corresponds to the Agent Management System (AMS) shown in Figure 4.13. The AMS manages the agent platform maintaining a directory of agent identifiers and agent state (http://jade.cselt.it). Each MADIS agent must register with the AMS upon activation.

The Directory Facilitator (DF) agent provides the yellow pages service to all MADIS agents. This is a FIPA-defined agent implemented by JADE that will be employed in MADIS to provide all the services for which the MADIS Directory Facilitator was designed.

The Message Transport System (shown in Figure 4.13) controls the exchange of messages within the agent platform. Agent communication is performed via asynchronous message passing through FIPA ACL. Messages are sent in the agent message queue being up to the receiver agent to decide when to read a waiting ACL message (see Figure 4.14).

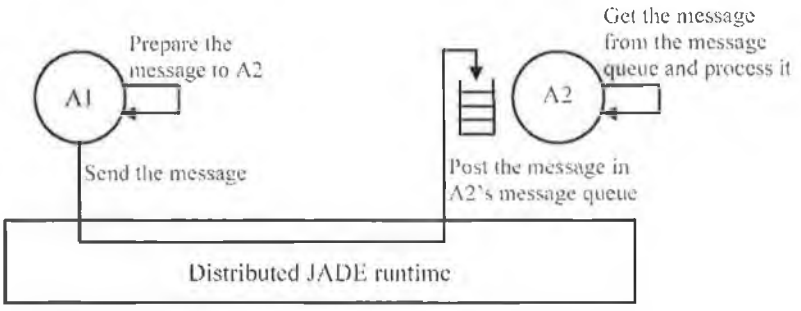


Figure 4.14. JADE message passing (http://jade.cselt.it)

Complying with the FIPA reference model presented in Figure 4.13, the JADE agent platform can be dispersed in a number of hosts but only one of them (i.e. the main container) will contain the AMS and DF (see Figure 4.15).

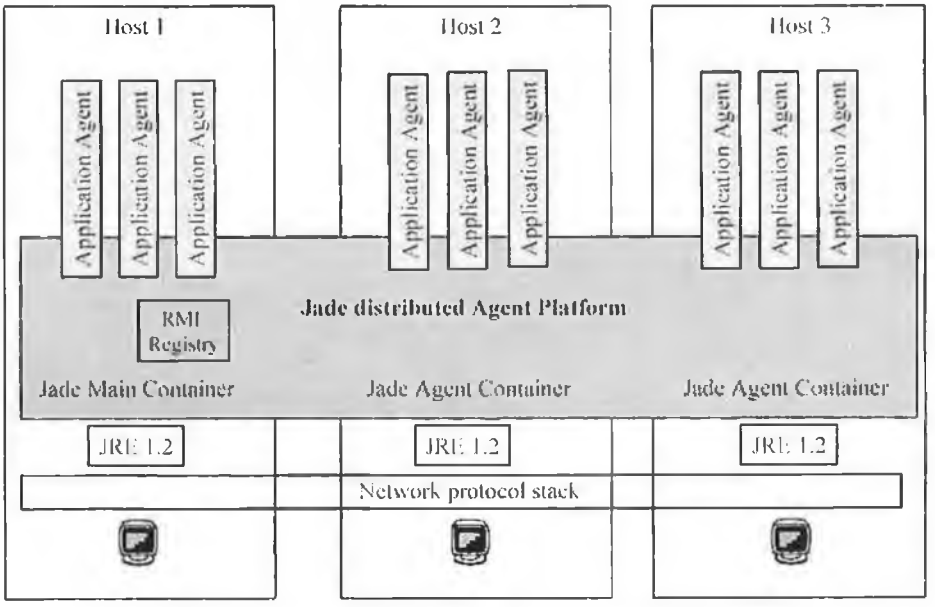


Figure 4.15. JADE agent platform (http://jade.cselt.it)

Referring to the MADIS prototype, the main container also includes agents from the ontology society (e.g. Ontology Broker, Component Receiver) besides the compulsory System Manager and the Directory Facilitator (implemented by JADE). A new container connected to the JADE agent platform will be created for each distributed user connected to MADIS through a User Interface Controller agent. This container will also include all the Application Agents serving the software applications used by the designer and the Component Sender agent. Furthermore, the user container will host any mobile agents that

migrated as a response to a user request (e.g. Ontology Reader). Figure 4.16 gives an example of a working MADIS platform viewed through the Remote Management GUI offered by JADE.

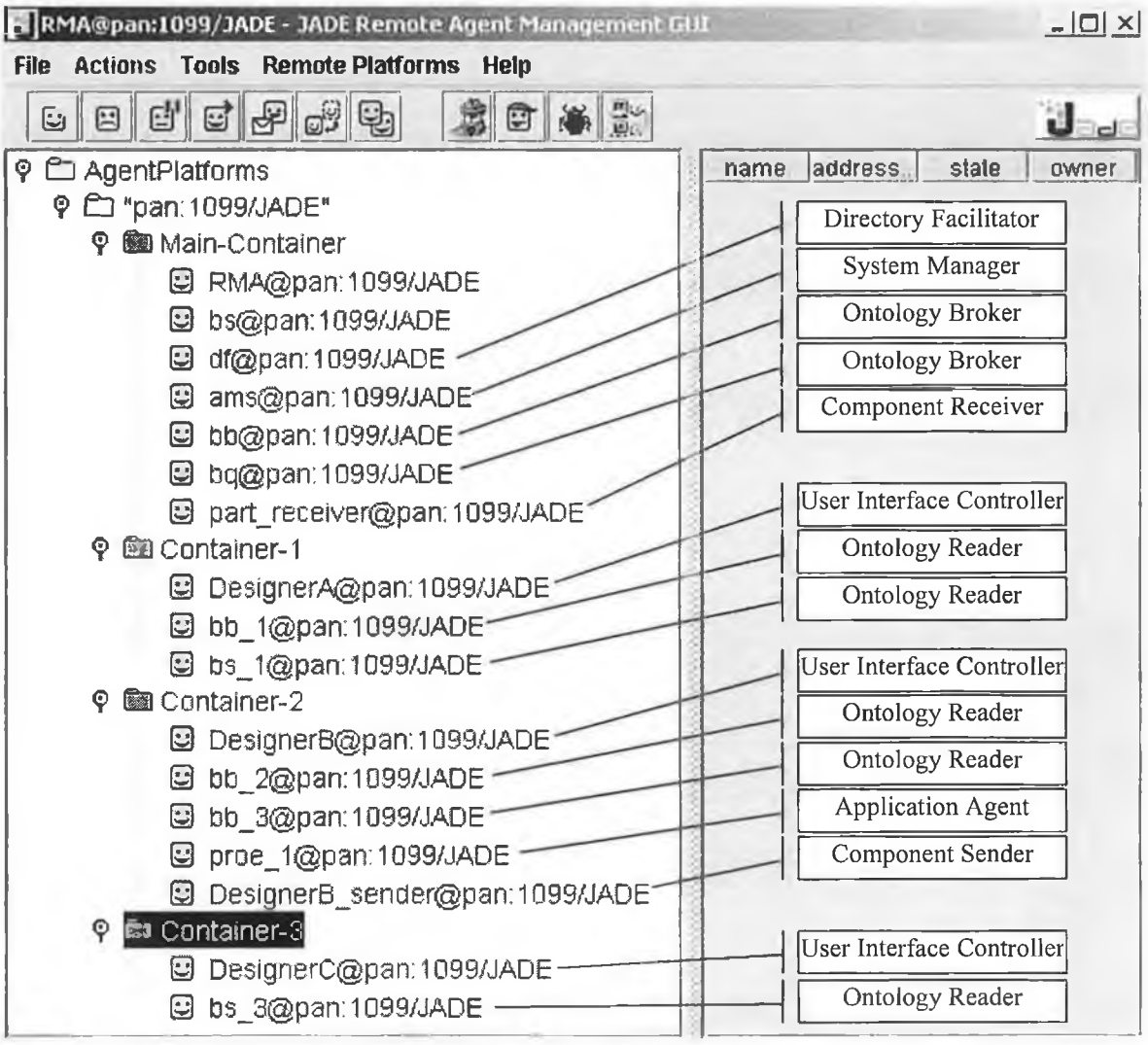


Figure 4.16. MADIS agents in JADE environment

The example provided in Figure 4.16 shows a MADIS agent platform with a main container including the AMS (i.e. System Manager in MADIS), the DF (Directory Facilitator in MADIS), the Ontology Broker and the Component Receiver agents. Three designers are connected to MADIS through their personal User Interface Controller agent (i.e. DesignerA, DesignerB, DesignerC) in three distributed hosts (i.e. Container-1, Container-2, Container-3). Furthermore, Container-1 includes two Ontology Reader agents (i.e. bs_3, bq_3) responding to information retrieval requests made by the user through the User Interface Controller.

4.4.2. User Agents

The User Interface Controller is the agent from the user agent society that received most attention during the implementation phase. This agent controls a graphical user interface (GUI) through which different MADIS services are provided to the user. The browse and search services are the only two implemented in this MADIS prototype as they were considered the most valuable for the purposes of this research as well as sufficient for demonstrating the MADIS objectives. A desired feature not implemented in this MADIS version refers to the creation of an intelligent GUI for the User Interface Controller that can dynamically change based on the user preferences managed by the User Profile Manager. More research into human-computer interaction, designer profile and intelligent user interfaces is necessary to achieve this goal (see Chapter six).

Being characterized by a GUI, the User Interface Controller agent class (i.e. name=MyAgent) extends the jade.gui.GuiAgent class provided by JADE (see Figure 4.17). The creation and management of the User Interface Controller GUI is performed in a separate class (i.e. name=MyAgentGUI) that extends javax.swing.JFrame.

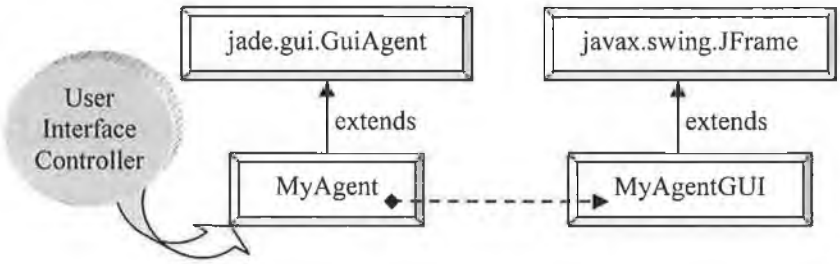


Figure 4.17. User Interface Controller implementation

Upon activation, the User Interface Controller gathers all the information required to create the GUI by cooperating mainly with the User Profile Manager. The next step involves the design of the user interface (performed using Java Swing in MyAgentGUI) and the GUI activation for the designer. Figure 4.18 presents an example of a User Interface Controller GUI.

Figure 4.18. User Interface Controller GUI

The User Interface Controller shown in Figure 4.18 enables user access to the services of browse and search the Material, Product, Fastener and Resource concepts. After activating its GUI, the User Interface Controller accepts requests made by the user. When a request is received, an ACL message will be created containing the name of the requested service (e.g. Browse), the name of the selected concept (e.g. Product) and the agent container (the computer where the User Interface Controller resides). In order to know where to send this message, the User Interface Controller enquires the Directory Facilitator for agents that have registered the requested service. Based on the template provided by the User Interface Controller, the Directory Facilitator identifies one or more Ontology Broker agents that are able to handle the requested service and sends back to the requester the Agent Identifier (AID) of these agents. The User Interface Controller is then able to complete the ACL message by adding each of these agents in the receiver list of the ACL message. The initial request made by the user is served when the User Interface Controller actually sends the constructed ACL message based on FIPA ACL. An Ontology Broker agent will make sure that the service is provided to the user in a timely manner by one or more of the Ontology Reader agents it manages (see section 4.4.4 for the implementation of the ontology dedicated agents).

4.4.3. Application Agents

The current MADIS prototype contains one Application Controller agent integrated in a CAD tool called ProEngineer 2001 (http://www.ptc.com). This integration was realized using a Java toolkit for ProEngineer called J-Link, which allows access to the internal components of a ProEngineer session. Each designer who uses this CAD system to model products will have an Application Controller (able to extract information from ProEngineer) and a Component Sender (able to forward information structures for storage) active. While the Component Sender works without any user interaction, the Application Controller can be managed (if desired) by the user through a ProEngineer menu (see Figure 4.19).

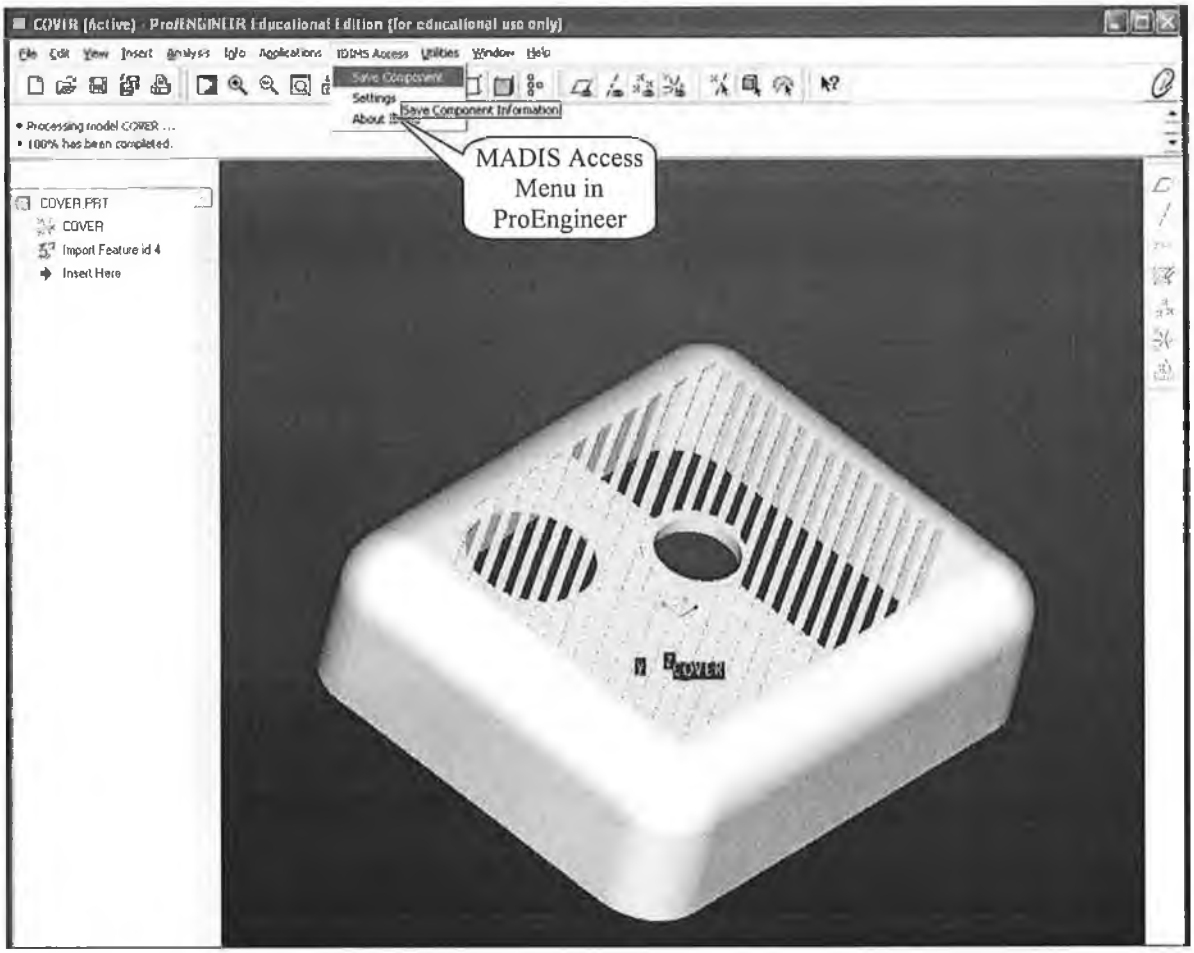


Figure 4.19. The ProEngineer Application Agent[7]

The information extracted by the Application Controller from ProEngineer refers to part name, part mass and other parameters that have a corresponding slot defined in the MADIS ontology. From a technical point of view, this information is mapped into an object that reflects exactly the definition of the class Part defined by MADIS ontology[8]. This object is sent by the Application Controller to the Component Sender agent that is active on the same machine waiting for requests. The first step took by the Component Sender in order to achieve the dynamically defined objective is to use the Directory Facilitator for finding the agent that can provide the service of saving a part to the correct ontology. Next, the Component Sender creates an ACL message containing the object received from the Application Controller agent and sends it to the Component Receiver AID returned by the Directory Facilitator.

[7] The name of project supporting the current research is IDIMS hence the name of the menu bar in the ProEngineer Application Agent.

[8] The MADIS implementation phase addressed the construction of Java classes corresponding to each concept (or class) defined in the MADIS ontology.

Both application agent classes extend the jade.core.Agent class provided by JADE and have no graphical user interface. However, the Application Controller agent can optionally inform the user of the success/failure of a save action through a simple Java Swing message.

4.4.4. Ontology Agents

The MADIS ontology[9] was implemented using the Resource Description Framework (RDF) and RDF Schema (RDFS) infrastructure (http://www.w3.org). Promoted by the World Wide Web Consortium (W3C), the RDF/RDFS model facilitates the encoding of *machine-processable* statements that can be easily exchanged and reused (Lassila and Swick 1999; Fensel 2000; Swartz and Hendler 2001). Furthermore, the development of the MADIS ontology was supported by the Protégé editor tool, which has been developed at Stanford University for the purpose of building domain ontologies (http://protege.stanford.edu).

The implementation phase focused on the Material and Structure Ontologies described in the previous section (i.e. section 4.3.3). Figure 4.20 presents the structure of the Material Ontology in Protégé.

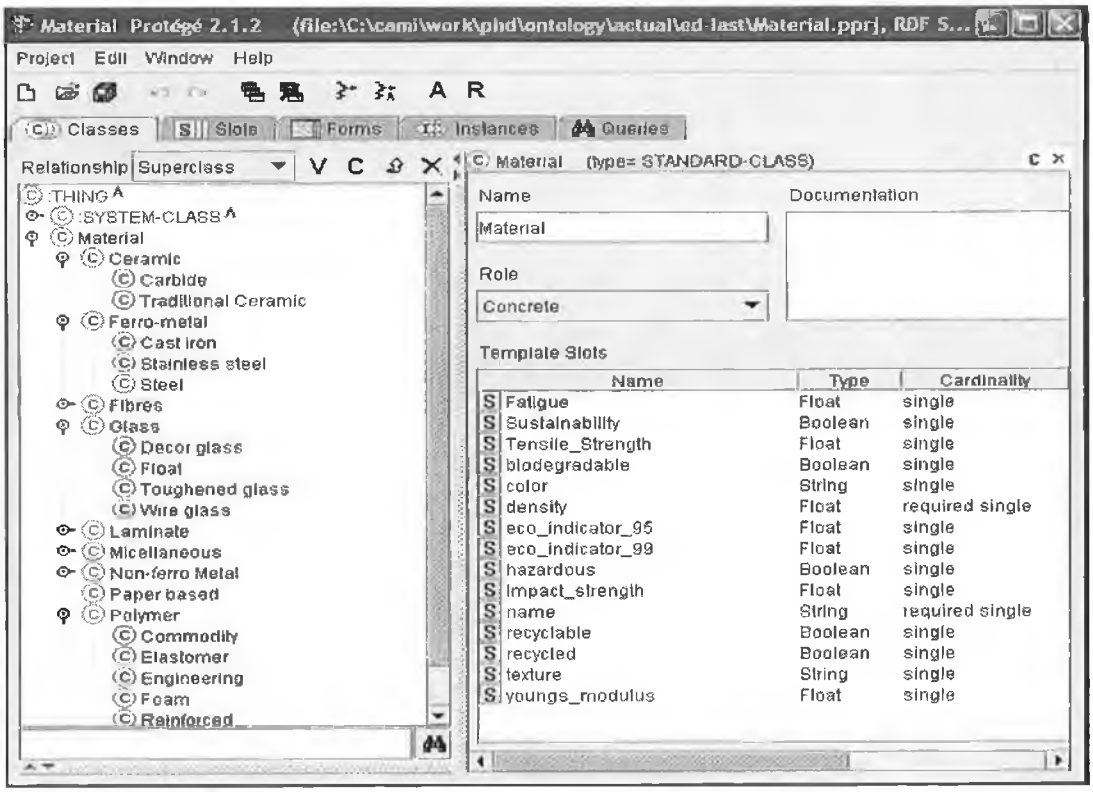


Figure 4.20. The Material Ontology: Protégé view

[9] The ontology developed to support MADIS is only a proof-of-the-concept version since the focus of the current research was not to design and implement an ontology library for engineering design.

All Material subclasses inherit the slots defined for the Material top level class e.g. name, density, sustainability, texture. In addition, the MADIS ontology was populated with some instances for the most commonly used materials (see Figure 4.21).

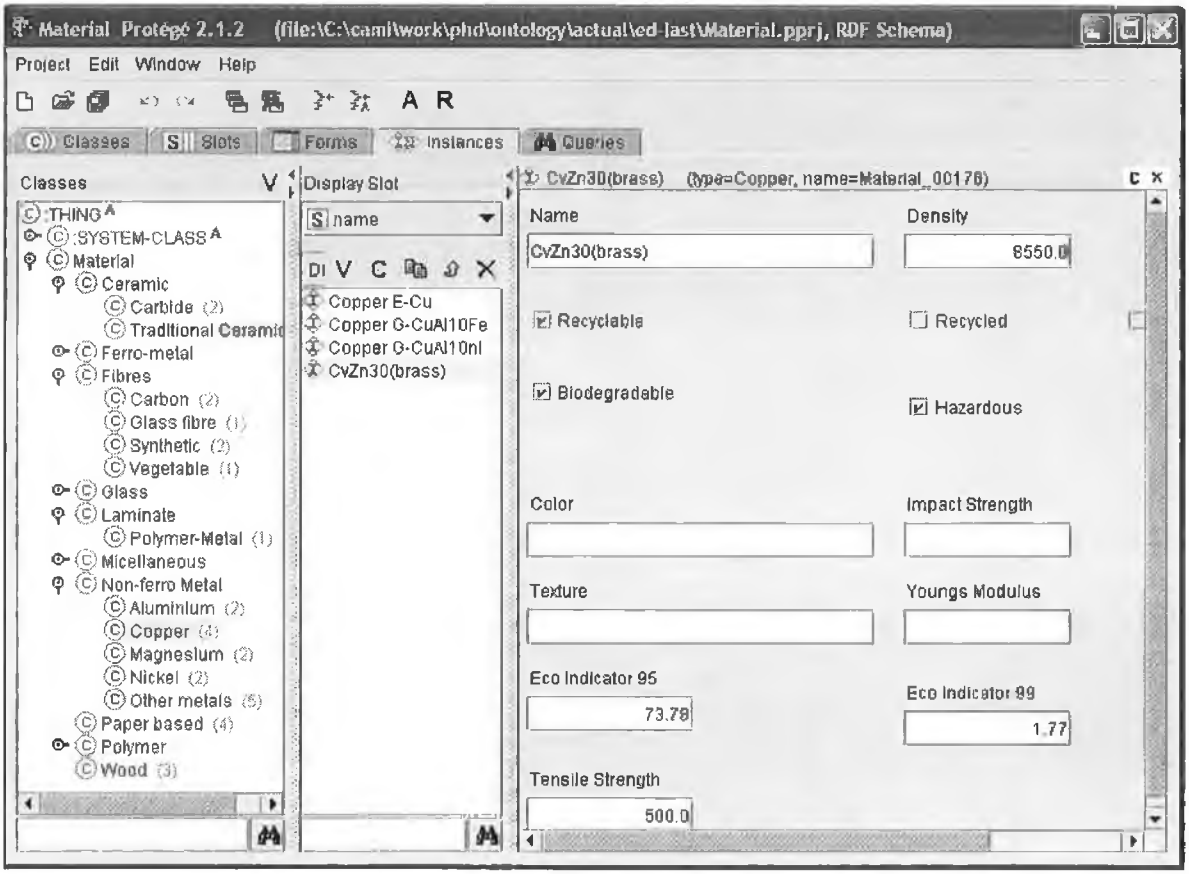


Figure 4.21. Material Ontology Instances

The Structure Ontology defines the MADIS representation and understanding of a product (considered an important domain entity as the product is the final outcome of the distributed design process). Each product is viewed as a hierarchy of *assemblies* and *parts*, with each assembly being made-up of further assemblies (also called subassemblies) and parts. The main constraint defined is that, while a part can be component of an assembly, an assembly cannot be a component of a part. An assembly is considered to be a product if it is not a component of any other assembly (it is not a subassembly). The assemblies and the parts are defined in terms of their characteristics (e.g. name, mass, version) and relations (has_author, has_manager, has_feature, has_material) that can link them to instances from other ontologies. Figure 4.22 shows the UML-based ontology diagram describing the most important subset of the Structure Ontology. The Material Ontology is used to represent the information regarding the material associated with a part while a Resource Ontology is employed to define the distributed design resources (e.g. human designers, design tools and applications).

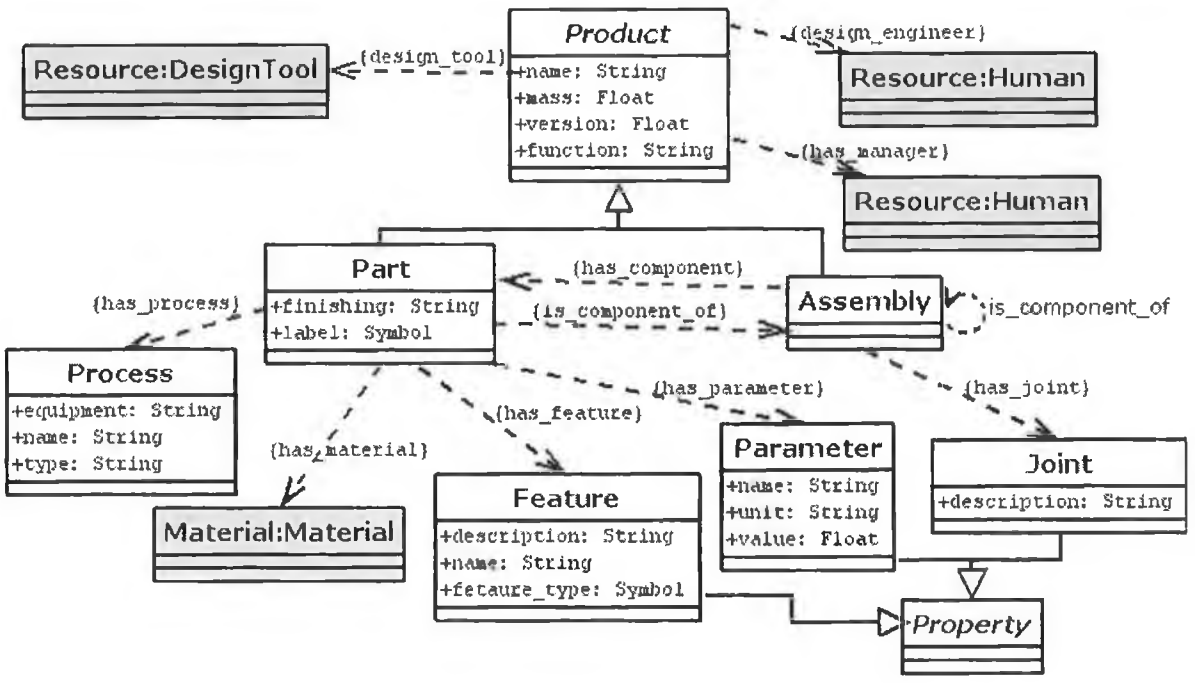


Figure 4.22. UML view over the Structure Ontology

The main classes of the Structure Ontology (represented in Protégé) are shown in Figure 4.23.

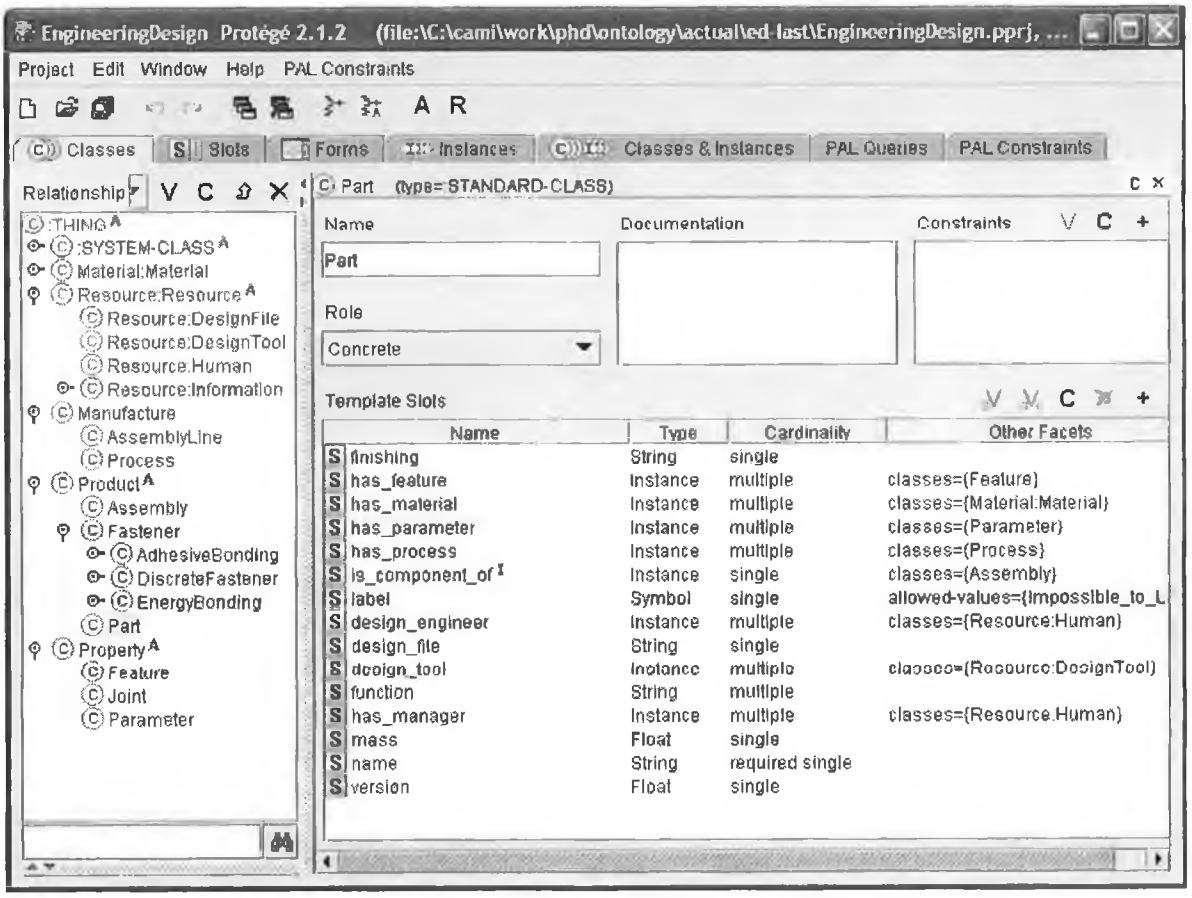


Figure 4.23. The Structure Ontology in Protégé

Finally, some simple product instances (e.g. a smoke alarm product, a chair, a car door mirror, a media server) have been created and added to the MADIS ontology for evaluation purposes. Figure 4.24 presents an example of ontological map using the MADIS model for a smoke alarm product.

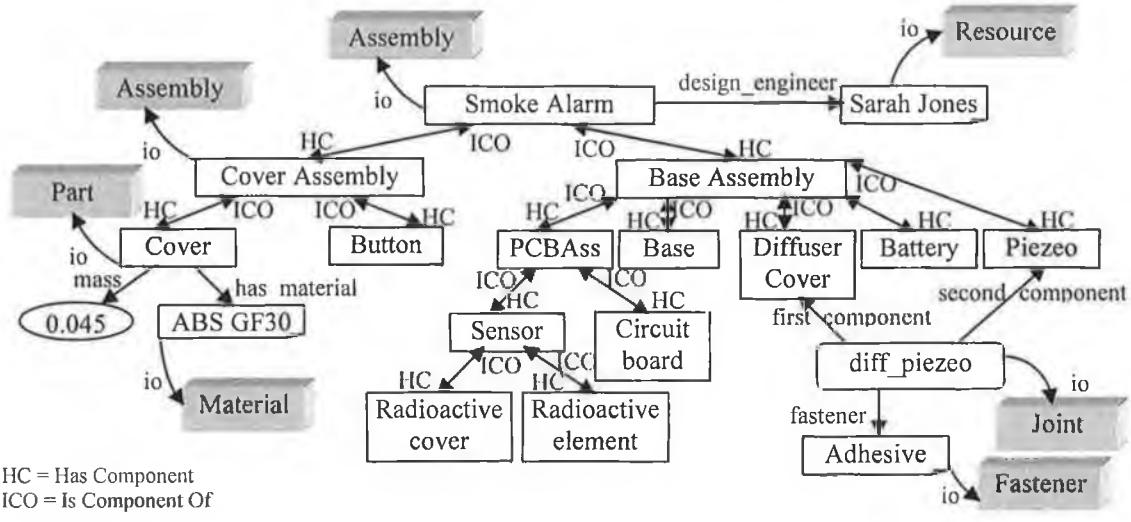


Figure 4.24. MADIS ontological model for a Smoke Alarm product

In the current implemented prototype, all MADIS ontologies and instance bases reside on the same machine simplifying the job of the software programmer significantly (without minimising the effects of MADIS in a distributed design environment). Therefore, the implementation phase focused on the Ontology Broker, Ontology Reader and Component Receiver, which were considered critical to the successful functionality of the MADIS infrastructure.

The *Ontology Broker* agents implemented can supply the services of browse, search and query (perform a specified query) the various concepts defined by the MADIS ontology (e.g. Material, Part, Assembly). This objective is achieved by activating the correct mobile agent (one of the Ontology Reader agents) that can provide the requested service. From a technical perspective, the Ontology Broker achieves its objective by extending the jade.core.Agent class and implementing agent behaviours through the jade.core.behaviours package (see Figure 4.25).

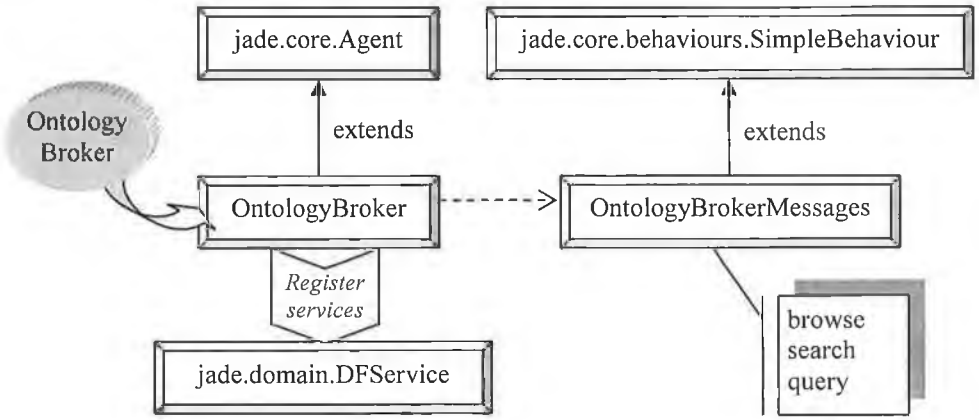


Figure 4.25. The Ontology Broker implementation

The Ontology Broker registers with the Directory Facilitator all the services it can provide based on the Ontology Reader agents and responds to ACL message requests made by other MADIS agents.

The *Ontology Reader* agents are mobile agents with the capability of reading the ontology and porting the information in a GUI through the MADIS network. Information is extracted from the MADIS ontology in two main ways: (i) through a browse service (*Ontology Reader Browse*) and (ii) through a search service that gives the user the option of specifying the criteria for extracting information using a GUI (*Ontology Reader Search*). In both cases, the Ontology Reader agents use the RDF/RDFS representation of the MADIS ontology to extract information. This process is facilitated by the Jena Semantic Web Toolkit (http://www.hpl.hp.com/semweb/jena). Developed at HP Labs, Jena is a Java API that features statement and resource centric methods for manipulating a RDF model as a set of RDF triples or as a set of resources with properties respectively. An important aspect of Jena employed in MADIS is the RDF Data Query Language (RDQL) support. Intended as model-level access mechanism that is higher level than RDF API, RDQL features an SQL-like language for retrieving sets of values providing query with triple patterns and constraints over a RDF model (http://www.hpl.hp.com/semweb/jena).

Once the information is extracted and formatted in a GUI (according to user preferences where available), the Ontology Reader agent migrates to the container of the User Interface Controller that initially made the information retrieval request. This task is implemented by adding specific behaviours to each Ontology Reader agent that allow him to move, clone and exit the agent platform. Figure 4.26 summarizes the implementation strategies incorporated in the Ontology Reader mobile agents.

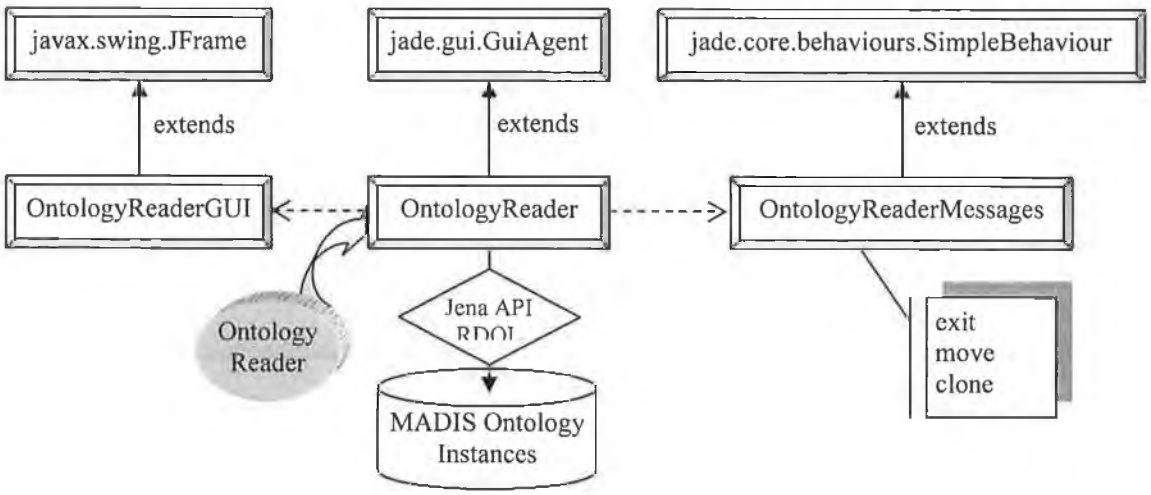


Figure 4.26. The Ontology Reader implementation

The Ontology Reader Browse provides the service of browsing the main concepts of the MADIS ontology. Depending on the concept selected by the user through the User Interface Controller (e.g. Material, Product, Fastener, Resource), the GUI of the Ontology Reader Browse contains all the instances of the selected entity (see Figure 4.27).

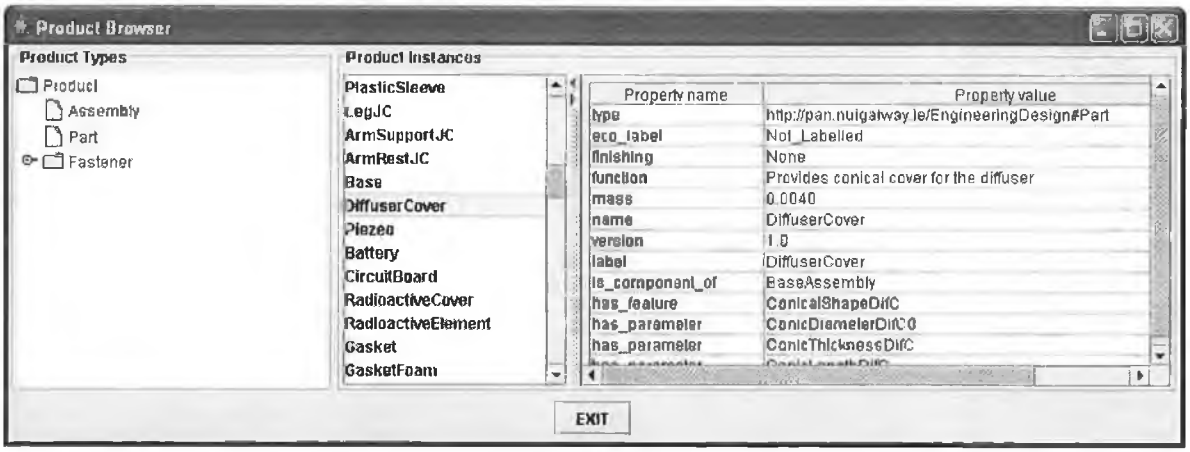


Figure 4.27. The Ontology Reader Browse GUI: browse product parts

The same information might be presented to the user in a different graphical format by the Ontology Reader for a different user that has other preferences sent by his/her User Interface Controller (see Figure 4.28).

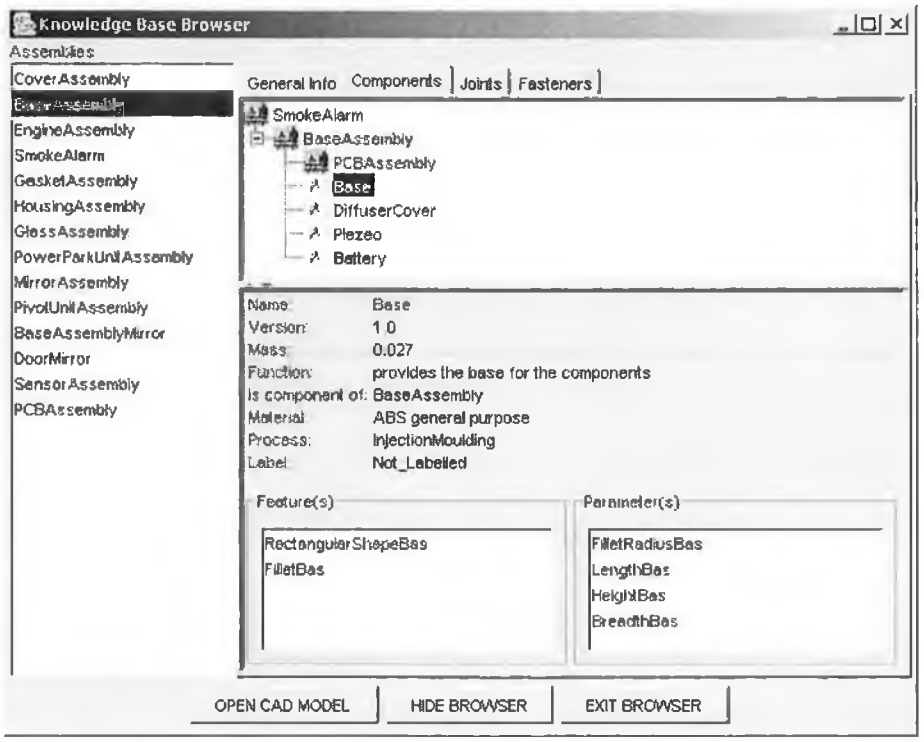


Figure 4.28. The Ontology Reader Browse: browse assemblies

The Ontology Reader Search provides a GUI to the requester user agent through which the user can set a search criteria for the selected concept (see Figure 4.29).

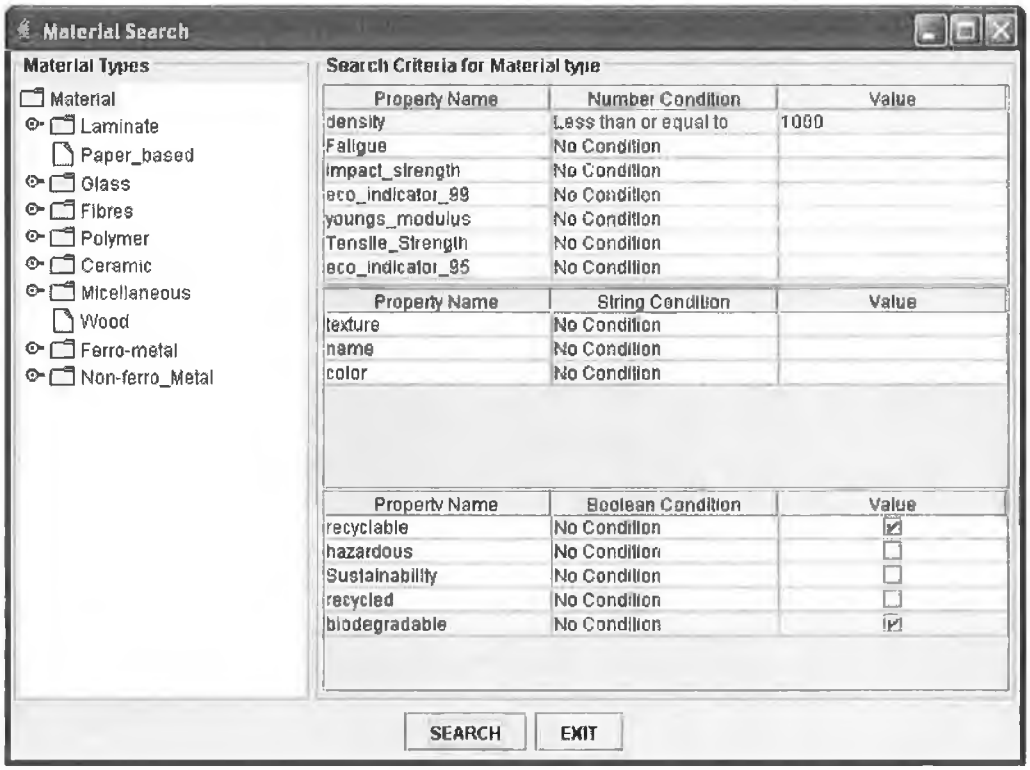


Figure 4.29. The Ontology Reader Search GUI: search material

After the user specified the search criteria (e.g. in Figure 4.29, the user searches for a material with the density less than or equal to 1000 that is recyclable and biodegradable), the Ontology Reader Search agent builds the corresponding Jena RDQL statement and sends an ACL Request message to an Ontology Reader Query agent that knows how to execute the given query and has access to the required ontological instances. The Ontology Reader Query agent extracts the information from the MADIS ontology that matches the given query using JENA RDQL and builds a GUI containing this information. For the example given in Figure 4.29, the RDQL statement looks like

```
SELECT ?material
WHERE (?material, <rdf#type>, <pan#Material>),
      (?material, <pan#density>, ?density),
      (?material, <pan#recyclable>, ?recyclable),
      (?material, <pan#biodegradable>, ? biodegradable)
AND ?density <= 1000,
      recyclable is true
      biodegradable is true
USING rdf FOR <http://www.w3.org/1999/02/22-rdf-syntax-ns>
      pan FOR http://pan.nuigalway.ie/EngineeringDesign
```

and the results of the query might be presented to the user in the GUI shown in Figure 4.28. (see Figure 4.30 for the results of the query specified in Figure 4.29).

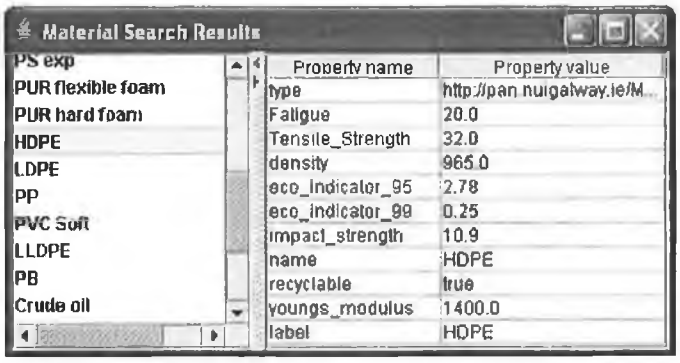


Figure 4.30. The Ontology Reader Query GUI: search material results

The migration to the requester container (the user computer) completes the activities of the Ontology Reader Query agent.

Finally, the *Component Receiver* agents have the ability to update the MADIS ontology by adding new instances of the various concepts defined in MADIS. This task is mainly performed as a response to requests made by the application agent society. The Component Receiver agents write the ontology using the Protégé API to ensure unique identification keys for each ontological instance. All changes made in the Protégé project file of the MADIS ontology will be propagated in the RDF/RDFS representation of the ontology ensuring designer access to up-to-date information. Upon task completion, the Component

Receiver sends an ACL inform message back to the requester agent to notify a successful/unsuccessful result.

4.4.5. Web Portal

The MADIS Web Portal offers the functionality of the user dedicated agents (see section 4.4.2) in a web environment. In the same way that the Ontology Reader agents extract information from the MADIS ontology, the Web Portal uses the Jena toolkit to read the RDF/RDFS model of the ontology. The web pages supplied through the Web Portal are dynamically generated using Java Servlets (http://java.sun.com/products/servlet/) supported by the Jakarta Tomcat servlet container (http://jakarta.apache.org/tomcat/) and the Apache Web Server (http://www.apache.org/).

The Web Portal protects the access to MADIS information through a username and password authentication login (see Figure 4.31).

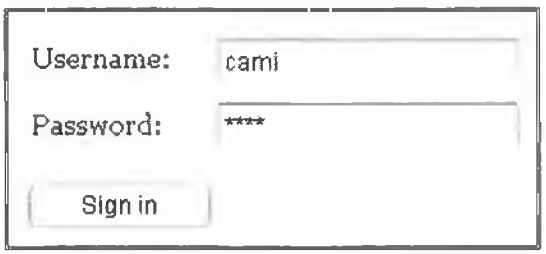


Figure 4.31. MADIS Web Portal access

The main services provided by the Web Portal are the same ones available through the User Interface Controller agent i.e. browse, search. Figure 4.32 presents the browse page of the MADIS Web Portal. The user can select the concept to be displayed through a tree-like structure of the main concepts defined in the MADIS ontology. This tree component is created at run-time by reading the ontology using the Jena API. After the user selected a concept (e.g. Assembly), a new page (displayed in the central frame within the same web page) is dynamically generated to contain a list of all instances for the Assembly class. More information particular to each Assembly instance will be displayed in a table format when selected from the list (see Figure 4.32).

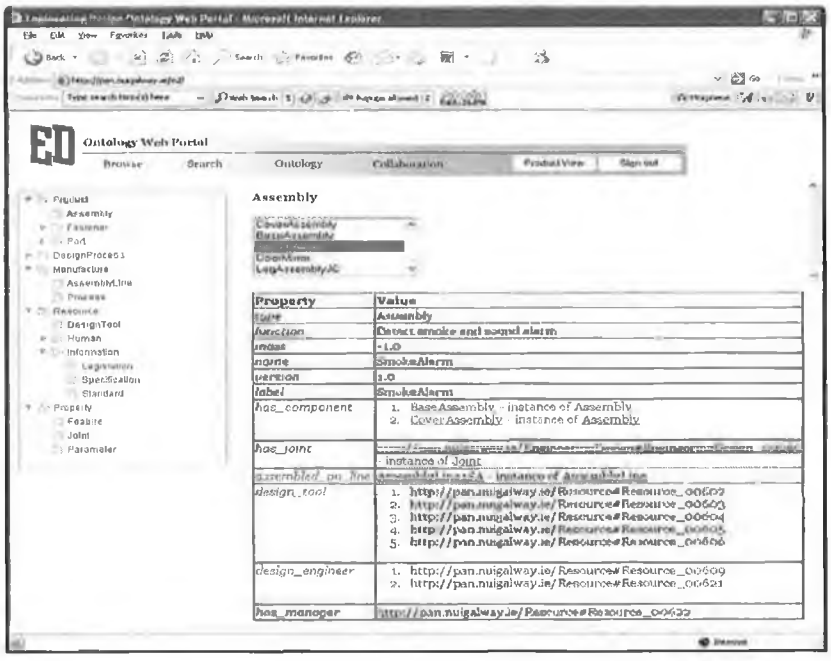


Figure 4.32. MADIS Web Portal: browse page

The search page of the MADIS Web Portal allows the user to set the criteria for information retrieval e.g. look for a part with the name like 'Cover' and the mass less than or equal to 0.5 kg (see Figure 4.33). This criteria is transformed into a RDQL statement and used in conjunction with Jena API to identify requested information, that will populate a new dynamic web page (see Figure 4.34).

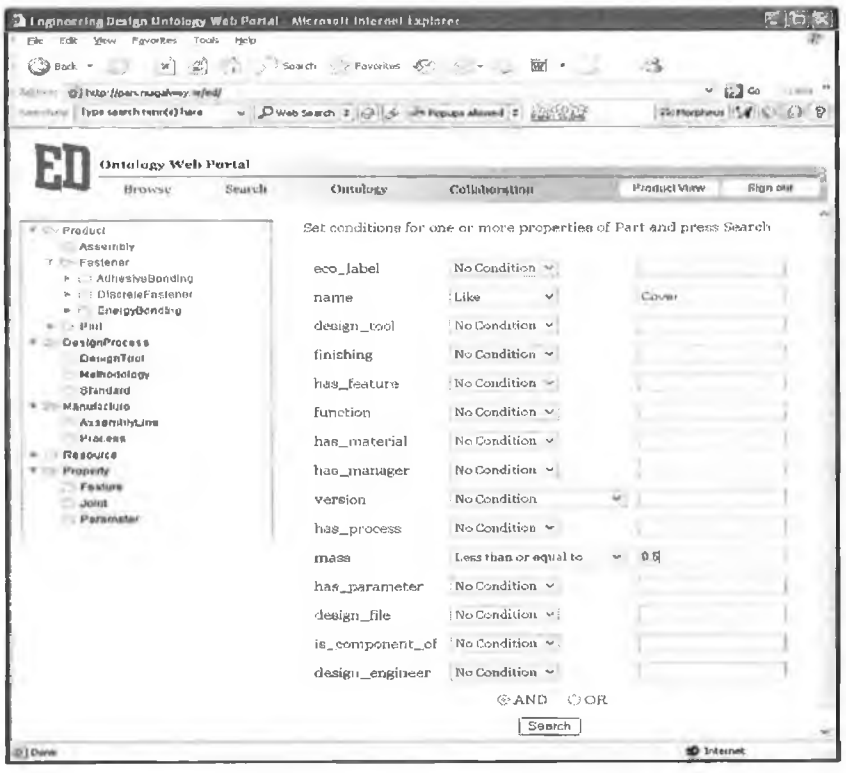


Figure 4.33. MADIS Web Portal: search page

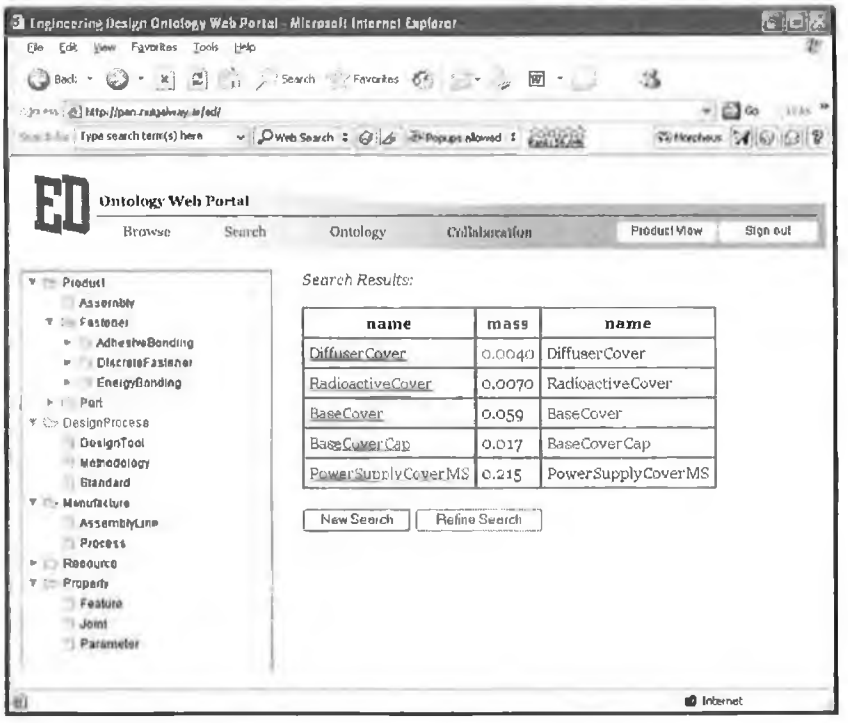


Figure 4.34. MADIS Web Portal: search results

Furthermore, users logged in to the MADIS Web Portal can use the *Collaboration Page* to cooperate in the distributed design environment through instant messaging and participation to virtual meetings enabled with text/audio/video information exchange, whiteboard capabilities and application sharing. These collaboration technologies are supported by an IBM software product called Lotus Sametime (http://www.lotus.com), which is a platform for real-time collaboration promoting presence awareness, instant messaging and web conferencing.

The MADIS Web Portal intention and vision is to prepare MADIS for Semantic Web (http://www.semanticweb.org) integration in order to semantically explore wide area networks such as the Internet besides the internal information currently available within MADIS (see Chapter six for further details).

4.5. Conclusions

Enabled by a multi-agent system and an ontological information base, the proposed MADIS system aims to efficiently support the distributed designer mainly by enabling the access to meaningful information, by integrating dispersed resources and by facilitating the sharing and exchange of information in a distributed environment. The information specific to the engineering design domain is mapped into an ontology library understood and processed by a multi-agent system. The MADIS design and implementation phases focused on the main agent societies (i.e. user, application, ontology and interconnection)

that cooperate to deliver the MADIS objectives. The development of MADIS was facilitated by the following strategies adopted during the implementation phase:

- The MADIS agents are implemented in the Java programming language with the support of the FIPA-compliant JADE platform.
- Agent interoperation is facilitated by the FIPA ACL. The content language that supports the definition of the ACL messages exchanged is FIPA SL.
- The agents commit to a common shared engineering design ontology stored in the RDF/RDFS model (with the support of the Protégé editor tool). The Jena Semantic Web toolkit is used to manipulate the RDF models from the Java code of the agents.

Compliant with the FIPA agent specifications, MADIS exploits agent properties such as autonomy, cooperation, learning and pro-activeness in a semantic approach to support a design process that involves dispersed heterogeneous resources and multidisciplinary people. The MADIS ontological and multi-agent based system aims to ultimately optimise engineering design operation and management by efficiently facilitating the management of the data-information-knowledge value chain.

References

Bauer, B. (2001). UML Class Diagrams: Revisited in the Context of Agent-Based Systems. Agent-Oriented Software Engineering, Montreal.

Bauer, B., J. P. Müller and J. Odell (2001). Agent UML: A Formalism for Specifying Multiagent Interaction. Agent-Oriented Software Engineering, Springer-Verlag, Berlin.

Bigus, J. P., D. A. Schlosnagle, J. R. Pilgrim, W. N. M. III and Y. Diao (2002). "ABLE: A toolkit for building multiagent autonomic systems." IBM Systems Journal 41(3).

Fensel, D. (2000). Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce. Berlin, Springer.

Fernandez, M., A. Gomez-Perez and N. Juristo (1997). METHONTOLOGY: From Ontological Art Towards Ontological Engineering Workshop on Ontological Engineering. Symposium on ONtological Engineering of AAAI, Standford, California.

Fernandez-Lopez, M. (2001). "Overview Of Methodologies for Building Ontologies." Intelligent Systems 16(1): 26-34.

Fernandez-Lopez, M., A. Gomez-Perez, A. Pazos-Sierra and J. Pazos-Sierra (1999). "Building a Chemical Ontology Using Methontology and the Ontology Design Environment." IEEE Intelligent Systems and their applications January/February: 37-46.

Gomez-Perez, A. (1999). "Ontological Engineering: A State Of The Art." Expert Update. Ontono 2(3): 38-43.

http://jade.cselt.it, JADE, Last Accessed August 2005.

http://jakarta.apache.org/tomcat/, Tomcat, , Last Accessed August 2005.

http://java.sun.com, Java, Last Accessed August 2005.

http://java.sun.com/products/servlet/, Java Servlets, Last Accessed August 2005.

http://protege.stanford.edu, Protege 2000, Last Accessed May 2005.

http://www.apache.org/, Apache, Last Accessed August 2005.

http://www.fipa.org, Foundation for Intelligent Physical Agents, Last Accessed August 2005.

http://www.hpl.hp.com/semweb/jena, JENA, Last Accessed August 2005.

http://www.lotus.com, Lotus Sametime, Last Accessed August 2005.

http://www.ptc.com, ProE, Last Accessed August 2005.

http://www.semanticweb.org, Semantic Web, Last Accessed August 2005.

http://www.w3.org, RDF, Last Accessed August 2005.

Huget, M.-P. (2002). Desiderata for Agent Oriented Programming Languages, University of Liverpool.

Huget, M.-P. (2002). Extending Agent UML Protocol Diagrams, University of Liverpool Department of Computer Science.

IEEE96 (1996). IEEE Standard for Developing Software Life Cycle Processes. New York (USA), IEEE Computer Society.

Lassila, O. and R. R. Swick (1999). Resource Description Framework (RDF) Model and Syntax Specification, W3C Recommendation 22 February 1999. 2003.

Odell, J., M. Nodine and R. Levy (2005). A Metamodel for Agents, Roles, and Groups. Lecture Notes on Computer Science. J. Odell, P. Giorgini and J. Müller. Berlin, Springer. Agent-Oriented Software Engineering (AOSE) V.

Odell, J., H. V. D. Parunak and B. Bauer (2000). Extending UML for Agents. Proceedings of the Agent-Oriented Information Systems Workshop at the 17th National conference on Artificial Intelligence.

Swartz, A. and J. Hendler (2001). The Semantic Web: A Network of Content for the Digital City. Proceedings Second Annual Digital Cities Workshop, Kyoto, Japan.

Zambonelli, F., N. R. Jennings and M. Wooldridge (2003). "Developing multiagent systems: the Gaia Methodology." ACM Transactions on Software Engineering and Methodology 12(3): 317-370.

Chapter 5

MADIS Evaluation

5.1. Introduction

Employing multi-agent systems and ontologies, the MADIS framework addresses the need for resource interoperation and integration as well as knowledge sharing and reuse in a distributed design environment. Having completed the design and implementation processes, the MADIS evaluation phase intends to examine thoroughly the proposed system in order to demonstrate the system capabilities, to validate the functionality of the system and to detect any potential errors.

The evaluation of MADIS includes a comprehensive comparison with other existing systems proposed by the research community to offer computational support for the distributed design process (see section 2.3 for the review). The actual testing and validation phase of MADIS uses the protocol analysis technique to evaluate the proposed system when used by a single designer or by a team of designers in a distributed environment to perform a given set of tasks. This chapter describes these protocol analysis tests presenting the data analysis process and the results obtained.

5.2. System Comparison

Referring to current trends in software support for distributed design, the second chapter of this thesis offers an extensive literature review of existing AI approaches to distributed collaborative engineering design support (see section 2.3). The focus of this section is to compare and contrast MADIS with the major AI-based software systems already reviewed. As described in the previous chapter, the MADIS architecture mainly addresses the problems of interoperation among dispersed resources and knowledge sharing, reuse and integration in a distributed design environment. These objectives are achieved by designing a multi-agent system and an ontology library to support knowledge management activities, distributed interoperation, resource integration and cooperation processes. Generally, the main objectives of a system intended to support the process of distributed engineering design can be classified as follows:

- Design data, information and knowledge management e.g. gathering, organization, refinement and distribution of information
- Interoperation among distributed resources
- Integration of distributed tools used by different designers
- Knowledge sharing, reuse and integration
- Content related support for information exchange

- Support of the cooperation process among distributed designers (e.g. collaboration, communication, coordination)

While most of the systems proposed by other researchers focus on the knowledge sharing element and on the cooperation aspect of distributed design, only a few also address other important issues such as distributed interoperability, resource integration or semantic knowledge management. Table 5.1 compares MADIS with other proposed systems in terms of the design objectives incorporated in the specification of the system.

System	Information and knowledge management	Distributed resource inter-operation	Integration of distributed tools	Knowledge sharing, reuse and integration	Content related support	Cooperation support
PACT		√	√	√		
SHARE	√	√		√		√
SHADE	√			√	√	√
DIDE			√	√	√	
ICM				√		√
CAIRO		√		√		√
CLOVER	√	√			√	√
WebBlow		√		√	√	√
ADLIB	√			√	√	√
MADIS	√	√	√	√	√	√

Table 5.1. System comparison at the specification level

Software agents or multi-agent systems are widely adopted by the research community to support the delivery of the proposed design objectives. Furthermore, some systems define an ontology to support knowledge management activities, others engage web technologies to extend system functionality and only a few adopt an existing and generally acknowledged standard. Table 5.2 compares MADIS with other proposed systems in terms of the implementation strategies adopted.

System	Agents	Agent standard	Ontologies	Internet / Web
PACT	√	√	√	
SHARE	√			√
SHADE	√	√	√	
DIDE	√		√	
ICM	√		√	√
CAIRO	√			√
CLOVER	√	√	√	√
WebBlow	√			√
ADLIB	√		√	
MADIS	√	√	√	√

Table 5.2. System comparison at the architectural level

Implementing and testing a prototype of the proposed system ensures that the design objectives and theoretical architectural model are viable proposals. However, a small number of the relevant studies report some implementation and testing results.

Compared with other existing distributed design management systems, MADIS remains a strong proposal addressing the major distributed engineering design issues by engaging emerging AI technologies. The delivery of a robust functional system is ensured by entirely covering the phases of system design, implementation, testing and validation.

5.3. Testing and Validation

The MADIS testing phase was a team effort involving two other PhD researchers working within the IDIMS project. David Tormey had an important role in creating the design scenarios for the test and he had an active contribution to the analysis phase. Valerie Butler was in charge with the setup of the communication environment used for the tests (i.e. Lotus Sametime).

The aim of the testing and validation phase is to evaluate the MADIS system in a distributed design environment in order to provide the developer with valuable information regarding the robustness and functionality of the proposed system infrastructure. The testing method selected is Protocol Analysis (PA). Consisting of collecting verbal data reports and systematically analysing them, PA is a qualitative evaluation method for human cognitive processes (Cross and Cross 1995; Ericsson and Simon 1999; Chan 2000; Benbunan-Fich 2001; Gero and Tang 2001). In a protocol analysis session, the subject is asked to complete a set of predetermined tasks and is observed by the evaluator who typically records users' actions using video and audio techniques. The users are asked to think aloud during or after performing the tasks describing what they believe it is happening, what they are attempting to do, why they take a specific action and other task-related thoughts. The process of verbalization reveals the assumptions, misconceptions, inferences and problems that users face while performing tasks or solving problems (Ericsson and Simon 1999; Roche 1999; Chan 2000; Benbunan-Fich 2001; Gero and Tang 2001). Considered an excellent choice for qualitative researchers interested in a reach source of data, the PA method is suitable for the testing phase of MADIS because of the following advantages (Henderson, Podd et al. 1995; Ericsson and Simon 1999; Roche 1999; Benbunan-Fich 2001; Gero and Tang 2001; Chira 2002):

- Efficient identification of the problems that occur when users interact with a computer-based system.
- Location of the negative aspects concerning the user acceptance of the system.
- Genuine capture of the user attitude towards the computer-based system offering an understanding of how users form their cognitive model of the system.

- Robust and efficient method for investigating causes of errors, mistakes and misinterpretations.
- Even a small number of subjects can trigger important results.

A lot of studies in the human-computer interaction field proved the efficiency of the PA method in revealing important usability problems associated with computer-based systems (Henderson, Podd et al. 1995; Greenberg 1996; Roche 1999; Branch 2000; Benbunan-Fich 2001). In the field of engineering, the PA technique has been used as the main method to study the cognitive activity of the designer whilst in the design process (Cross and Cross 1995; Goldschmidt 1995; Gero and McNeil 1997; Roche 1999; Chan 2000; Gero and Tang 2001; Chira 2002).

Regarding the MADIS testing using PA, a number of four[1] design engineers were asked to complete a set of design related tasks and verbalize their thoughts and actions in the same time. The subjects selected were not familiar with the MADIS system or with the emerging AI technologies employed by MADIS. The testing procedure was divided in three parts as follows:

- *User Introduction* – the context of the test was explained to participants and the environment of the test was described (see Appendix 1). The MADIS system was briefly introduced and a live demonstration of MADIS Agents and Web Portal was carried out. Furthermore, subjects were allowed 15 minutes to familiarize themselves with the testing environment.
- *The MADIS test* – two major PA tests consisting of the actual performing of the tasks were carried out.
- *Feedback* – a short review was held at the end of the test in which participants were asked to provide any comments, opinions or suggestions they have about MADIS (see Appendix 2).

As part of the actual MADIS test, the subject designers were asked to use traditional distributed technologies and the MADIS system in order to complete the given tasks. The intention was to evaluate the MADIS system itself using the PA approach and furthermore to compare it with traditional groupware technologies currently used by designers (in a best-case real scenario) to share information in a distributed design environment. The groupware technology selected for this reason is Lotus Sametime Document Repository

[1] The PA technique has the great advantage that it doesn't require large sample sizes (Ericsson and Simon 1999; Roche 1999; Benbunan-Fich 2001). Due to the richness of data obtained via protocol analyses, a small number of subjects representative of the target population can yield important results (Benbunan-Fich 2001).

(the communication tools provided by Lotus Sametime are currently used within the MADIS Web Portal to support audio/video communication and instant messaging). Sametime Document Repository allows logged users to access and upload documents through a web-based interface. These documents can be organized in different folders and a number of attributes can be set for each. Users with the required permission access can retrieve the documents by browsing or searching them based on different arguments.

Two main experiments have been conducted as part of the MADIS test as follows:

1. *Time-Metric Test* looked at a single designer using the system to complete a set of tasks.
2. *Collaboration Test* used a team of two designers geographically distributed who had to use the system and collaborate in order to complete a set of tasks.

PA was applied in both tests with the difference that subjects were not asked to talk aloud during the Collaboration Test but their verbal communication (using audio technologies) required for collaboration was used as a verbalization protocol. Two cameras were used for each test conducted: one captured the face and body posture of the subject and the second one recorded the screen of the computer (mouse tracking protocol).

5.3.1. The Time-Metric Test

The Time-Metric test required subjects to undertake a task using both the Lotus Sametime Document Repository and the MADIS system while verbalizing their thoughts. The task consisted of getting specific information (i.e. mass, function, eco-label, finishing, process and parent assembly) about the 15 parts of a given product called the Media Server (see Figure 5.1).

Figure 5.1. The media server product used in the Time-Metric Test

The subjects were asked to extract the required information using Sametime Document Repository for the first 5 parts, MADIS Agents for the next 5 parts and MADIS Web

Portal for the last 5 parts (see Appendix 3 for the full task description). An observer was present in the same room with the user to monitor the subject's actions and behaviour and to remind him/her to talk aloud when necessary.

Data Analysis

The transcripts of the Time-Metric PA session were designed to support the capture and analysis of the subject's exact verbalization, the observer's notes and the records of user's actions. Two protocols were recorded as follows:

1. *Think aloud protocol* – consists of the subject's verbalizations.
2. *Mouse tracking protocol* – consists of the exact screens that had to be used by the subject in order to complete the given tasks.

Figure 5.2 shows the process flow of the protocol recording activity.

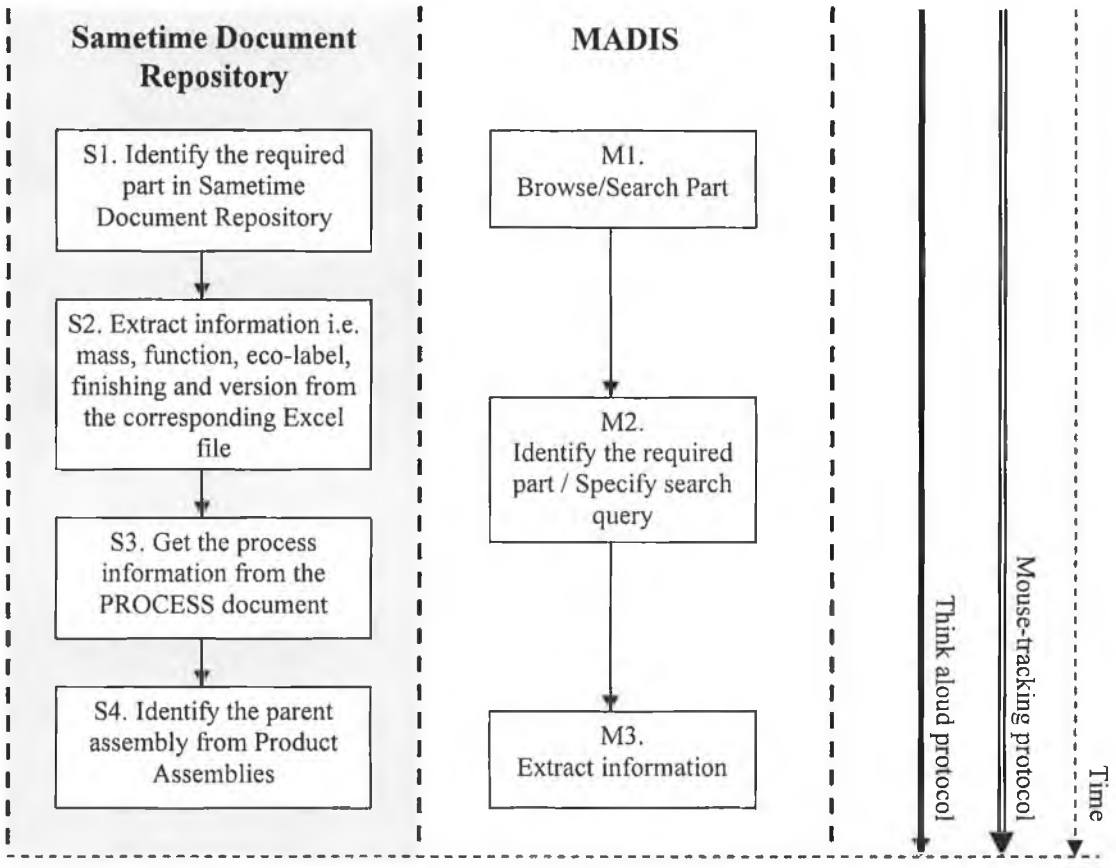


Figure 5.2. Time-Metric Test: tasks and protocols

Using Sametime Document Repository, the user had to identify the required part in the repository, extract information from the corresponding spreadsheet and get the process and parent assembly information from other documents contained within the repository. Using the MADIS Agents or Web Portal, the user had the option of using one of two services as follows:

- Using the *Browse* service, the user had to browse through the parts contained in the MADIS ontology, identify the required part and extract the information requested by the task.
- Using the *Search* service, the user had to specify the search query for a part (e.g. name like 'RearInterface') and extract the required information from the search results.

Three process flow models were created as follows:

(1) One model for the screens necessary to complete the Time-Metric tasks using the Sametime Document Repository (see Table 5.3)

(2) One model for the screens necessary to complete the Time-Metric tasks using the MADIS Agents (see Table 5.4)

(3) One model the screens necessary to complete the Time-Metric tasks using the MADIS Web Portal (see Table 5.5)

Each screen was assigned a screen code containing a letter and the task code from Figure 5.3 where that screen was necessary. For example, screen J(M2,M3) represents screen 'Product Browser Agent' from MADIS system required to support tasks M2 and M3 shown in Figure 5.2.

Screen Code	Screen Name	Steps
A(S1)	Sametime Document Repository – Product Parts	1. Open Product Parts folder 2. Locate the required part 3. Open the part file by double-clicking
B(S2)	Main Document View – Part file	1. Follow the part link 2. Extract necessary information from excel sheet i.e. mass, function, finishing, eco-label and version
C(S3)	Sametime Document Repository – Processes	1. Open Processes folder 2. Open the processes file by double-clicking
D(S3)	Main Document View – Processes file	1. Follow the processes link 2. Locate the part name and extract the corresponding process
E(S4)	Sametime Document Repository – Product Assemblies	1. Locate the MediaServer assembly. 2. Open the file by double-clicking
F(S4)	Main Document View – Assembly file	1. Follow the assembly link 2. Note the children components
G(S4)	Sametime Document Repository – Product Assemblies – Identified Subassembly	1. Locate each noted children component (subassembly). 2. Open the file by double-clicking
H(S4)	Main Document View – Subassembly file	1. Follow the subassembly link 2. Identify if the required part is among the children

Table 5.3. Screens necessary to complete the Time-Metric tasks using the Sametime Document Repository

Screen Code	Screen Name	Steps
I(M1)	Designer Personal Agent	1. Open Browse/Search tab 2. Click Product Button
J(M2, M3)	Product Browser Agent	1. Select Part from concepts tree 2. Select the required part in the list 3. Extract part information
K(M2)	Product Search Agent	1. Select Part from concepts tree 2. Set search criteria e.g. name like 'Bracket' 3. Click Search button
L(M3)	Query Results	1. Select required part from the result list 2. Extract information

Table 5.4. Screens necessary to complete the Time-Metric tasks using the MADIS Agents

Screen Code	Screen Name	Steps
W(M1)	Web Portal – Main Page	1. Select Browse/Search link
X(M1,M2,M3)	Browse Page	1. Select Part from the concepts tree 2. Select required part from the list 3. Extract information
Y(M1,M2)	Product Search Page	1. Select Part from the concepts tree 2. Set search criteria e.g. name like 'Bracket' 3. Click the Search button
Z(M3)	Search Results Page	1. Click on required part from the result list 2. Extract information

Table 5.5. Screens necessary to complete the Time-Metric tasks using the MADIS Web Portal

Using MADIS, the user had the option of choosing between the *Search* and the *Browse* service to complete the Time-Metric tasks. Therefore, the screens presented in tables 5.4 and 5.5 contain both possible paths (e.g. using MADIS Agents, either screen B(M2, M3) *or* both C(M2) and D(M3) screens will be used by subjects to perform the tasks).

Supported by the mouse tracking protocol recorded, the segmentation process for the transcripts of the think aloud protocol was performed according to the screens and steps the subjects used in order to complete the given tasks. For each subject, three transcripts of the Time-Metric PA session have been created as follows:

(1) A transcript of the Sametime Document Repository PA session (see Table 5.6)

(2) A transcript of the MADIS Agents PA session (see Table 5.7)

(3) A transcript of the MADIS Web Portal PA session (see Table 5.8)

Appendix 4 contains the complete list of PA transcripts for the Time-Metric Test.

Time Metric Test - *Sametime Document Repository*
Observer's notes: • No problem with verbalization. • Users generally found the Sametime Document Repository difficult to use • Frustration was observed (e.g. "too many clicks") • Parent Assembly information was very difficult to obtain (few users needed some suggestions) • As time progresses users worked quicker learning from experience to some extent

NO	Time Start	Time End	Notes
1	00:25	00:40	Identify part file for 'BracketMS'
2	00:40	01:30	Extract part information i.e. mass, function, finishing, eco-label and version
3	01:30	02:40	Get the process used for the part; *"How am I supposed to find this part"*
4	02:40	05:54	Get parent; Confusion; *"That's not a proper way to look for information"*
Time Duration = 5:29			
1	06:10	06:50	Identify part file for 'RearInterface'
2	06:50	07:40	Extract part information i.e. mass, function, finishing, eco-label and version
3	07:40	09:56	Get the process used for the part; *"This is annoying"*
4	09:56	12:30	Get the parent assembly; Confusion; *"Very cumbersome"; "This is very bad"; Sighs; Observer at 11:01*
Time Duration = 6:20			
1	13:00	13:30	Get the parent assembly;
2	13:30	13:48	Get the process used for the part;
3	13:48	14:00	Identify part file for 'ChasisBaseMS'
4	14:00	15:06	Extract part information i.e. mass, function, finishing, eco-label and version
Time Duration = 2:06			
1	15:10	15:50	Identify part file for 'PowerSupplyCoverMS'
2	15:50	16:28	Extract part information i.e. mass, function, finishing, eco-label and version
3	16:28	16:55	Get the process used for the part;
4	16:55	18:40	Get the parent assembly; *Frustration; Sighs*
Time Duration = 3:30			
1	18:45	19:03	Identify part file for 'PCB1MS'
2	19:03	19:55	Extract part information i.e. mass, function, finishing, eco-label and version
3	19:55	20:27	Get the parent assembly
4	20:27	21:02	Get the process used for the part
Time Duration = 2:17			

Table 5.6. Transcript of the Sametime Document Repository PA session of the Time-Metric test for one of the subjects

Time Metric Test - *MADIS Agents*			
Observer's notes: • No problem with verbalization. • Agents were found easy to use and intuitive • The search service provided by user agents was preferred • A more relaxed approach to performing the task as observed			
NO	**Time Start**	**Time End**	**Notes**
1	22:30	24:02	Uses the search service to locate the required part i.e. name like "BezelMS"
2	24:02	25:09	Extracts part information from Agent Query Results i.e. mass, function, finishing, eco-label, process, parent assembly and version; *"This is a lot more informative"; "This is easier: with a simple search you get what you want"*
Time Duration = 2:39			
1	25:38	25:45	Uses the search service to locate the required part i.e. name like "MechHardwareMS"
2	25:45	26:30	Extracts part information from Agent Query Results i.e. mass, function, finishing, eco-label, process, parent assembly and version
Time Duration = 0:56			
1	26:35	27:00	Uses the search service to locate the required part i.e. name like "Plugs01MS"; *"Learning curve is a lot quicker"*
2	27:00	27:40	Extracts part information from Agent Query Results i.e. mass, function, finishing, eco-label, process, parent assembly and version; *"This is a lot better!"*
Time Duration = 1:25			
1	27:44	28:09	Uses the search service to locate the required part i.e. name like "PCB2MS"; *"Search is accurate"*
2	28:09	28:57	Extracts part information from Agent Query Results i.e. mass, function, finishing,

			eco-label, process, parent assembly and version; Subject very happy with the agent performance
Time Duration = 1:13			
1	29:04	29:24	Uses the search service to locate the required part i.e. name like "LedsMS"; Not frustrated;
2	29:24	29:58	Extracts part information from Agent Query Results i.e. mass, function, finishing, eco-label, process, parent assembly and version; The interface is easy to use, very intuitive.
Time Duration = 0:54			

Table 5.7. Transcript of the MADIS Agents PA session of the Time-Metric test for one of the subjects

Time Metric Test - *MADIS Web Portal*			
Observer's notes: • No problem with verbalization. • The Web Portal was easy to navigate • The interface was considered friendly • Most subjects experienced both Browse and Search services provided			
NO	**Time Start**	**Time End**	**Notes**
1	30:45	31:40	Identify the part "MetalSheet1MS" using Browse service
2	31:40	32:42	Extracts part information i.e. mass, function, finishing, eco-label, process, parent assembly and version; *"This is very good"*
Time Duration = 1:57			
1	32:46	33:30	Identify the part "MetalSheet2MS" using Browse service
2	33:30	34:10	Extracts part information i.e. mass, function, finishing, eco-label, process, parent assembly and version; *"This is easy"*
Time Duration = 1:24			
1	34:15	34:25	Identify the part "NetworkSocketSupportMS" using Browse service
2	34:25	35:10	Extracts part information i.e. mass, function, finishing, eco-label, process, parent assembly and version; *"The interface is nice, easy to navigate"*
Time Duration = 0:55			
1	35:10	35:40	Identify the part "PinsMS" using Browse service
2	35:40	36:15	Extracts part information i.e. mass, function, finishing, eco-label, process, parent assembly and version
Time Duration = 1:05			
1	36:20	36:30	Identify the part "LabelMS" using Browse service
2	36:30	37:20	Extracts part information i.e. mass, function, finishing, eco-label, process, parent assembly and version
Time Duration = 1:00			

Table 5.8. Transcript of the MADIS Web Portal PA session of the Time-Metric test for one of the subjects

Each PA transcript created for the Time-Metric test contains the observer's notes, the segmentation of the episodes and a description for each episode.

Results

All subjects experienced difficulties when identifying and extracting the required information using the Sametime Document Repository (see Figure 5.3). After all the information about the first part was extracted, some improvement in the times registered

was noticed but the usage of the environment still generated verbal episodes such as *"This is annoying"* or *"This is very bad"*. The most difficult part of the task was identifying the parent assembly of the given part (screens E, F, G and H supporting task S4). A series of non-verbal codes was also registered including frustration, grumble and irritation.

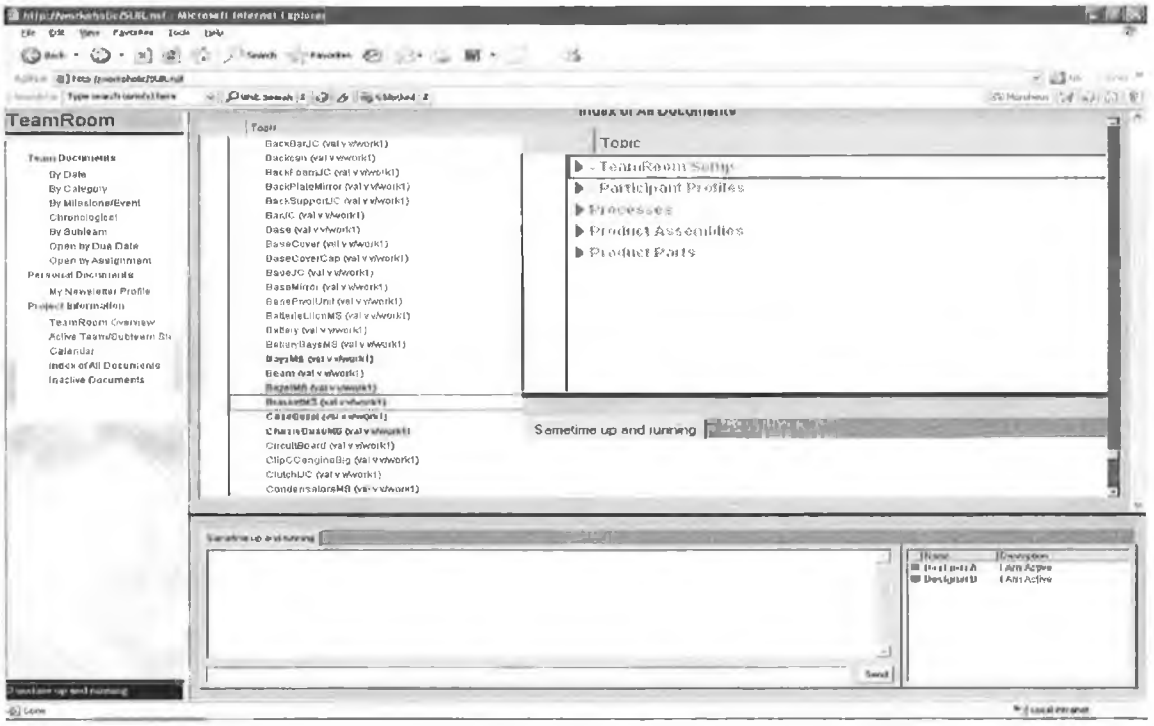
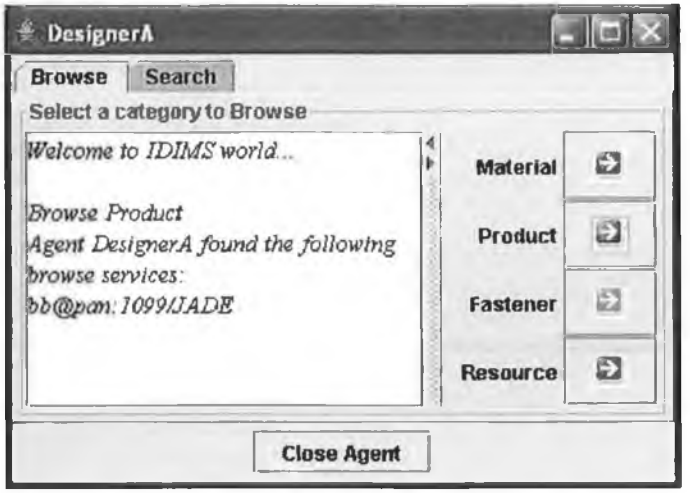
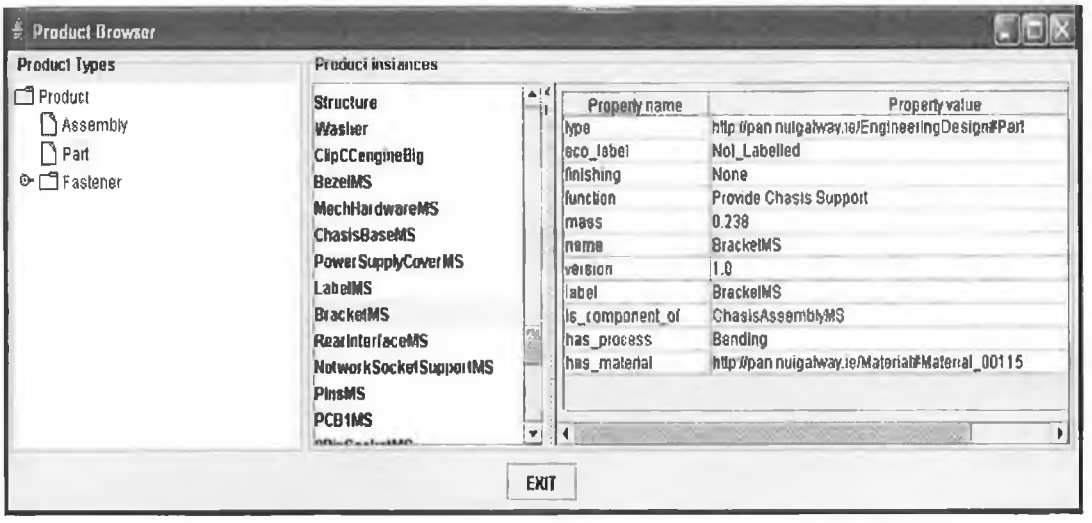


Figure 5.3. Sametime Document Repository

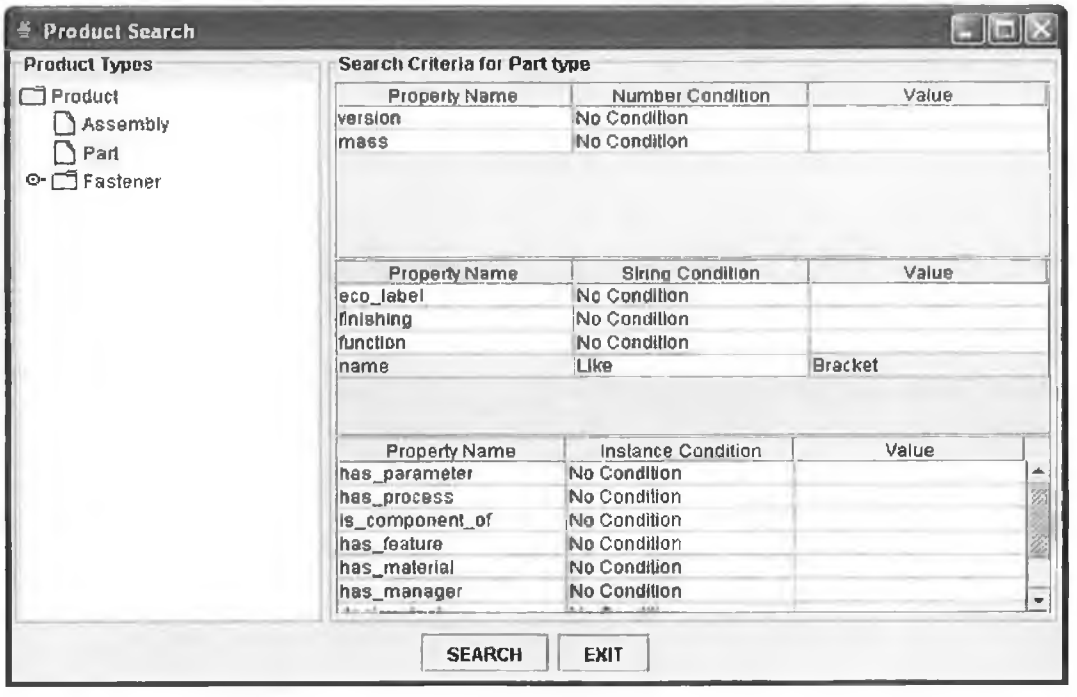
Although this was the first time they were using the MADIS system, none of the users had any functionality problems. Frustration was replaced with a more relaxed behaviour while the MADIS agents/web agents were trusted to act as instructed by the user (e.g. *"This is much better"*, *"This is easier: with a simple search you get what you want"*, *"This is very easy"*). Agent properties such as cooperation (mainly between the Personal Agent and the Browse and Search Agents) and mobility (e.g. Search Agent) were highly beneficial to the successful completion of the given tasks. While using the MADIS agents, both browse and search services were engaged (see Figure 5.4) but most subjects preferred to use the search agent.



(a) Personal Agent



(b) Browse Agent



(c) Search Agent

Figure 5.4. MADIS agents used during the Time-Metric Test

While using the MADIS Web Portal, the browse and search services were both tried out by users but most of them preferred the browse web page (see Figure 5.5). It was noticed that once the user obtained the desired results from a service becoming more familiar with its interface, he/she would engage the same service for the rest of the tasks.

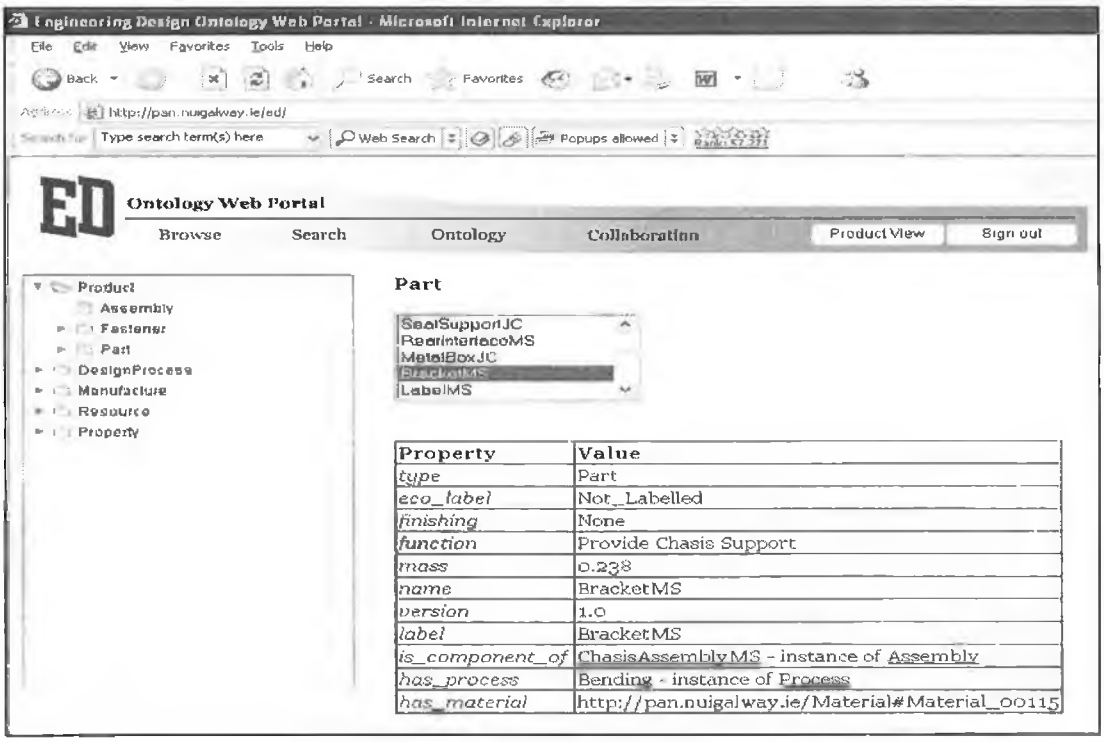
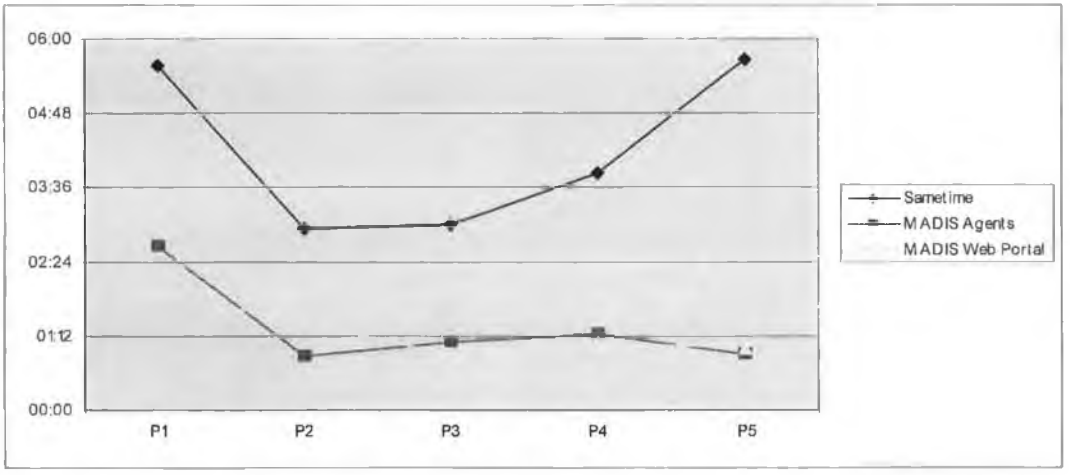
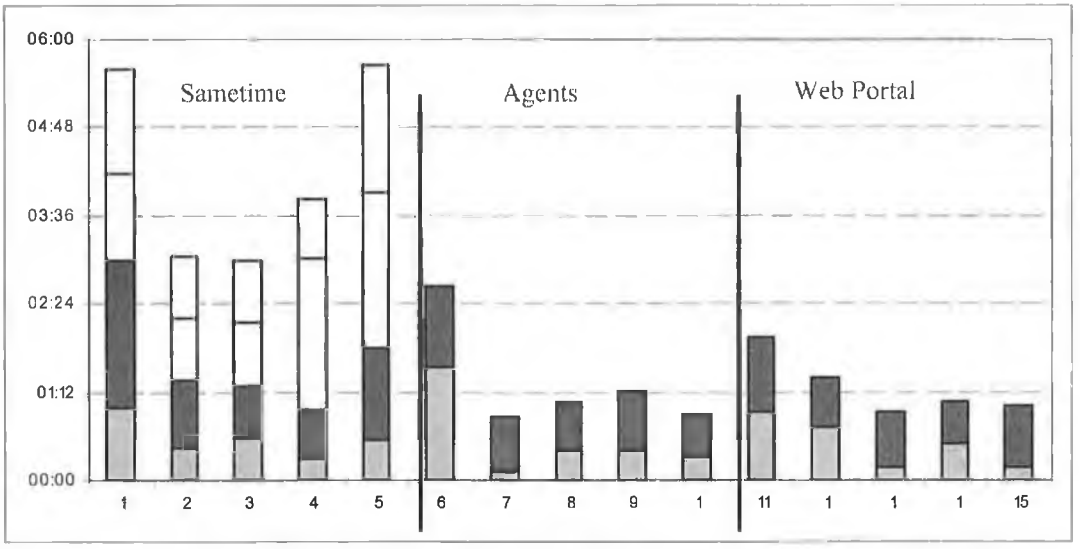


Figure 5.5. MADIS Web Portal – Browse Page

The information retrieval times improved significantly after subjects started to use the MADIS Agents and Web Portal to complete the tasks. Figure 5.6-a shows the time line chart of part information retrieval times for one of the participating subjects. Clearly, the Sametime Document Repository required most of the user's time whereas the MADIS Agents and Web Portal have about the same amount of time invested by the user. This is further demonstrated in Figure 5.6-b, which reflects the number and length of the PA episodes for each component. The higher number of episodes in the Sametime sessions suggests that the MADIS system was easier and more straightforward to use (besides providing the requested information faster).



(a) Line Chart for times recorded using Sametime, MADIS Agents and MADIS Web Portal



(b) PA Episodes in the Time-Metric test

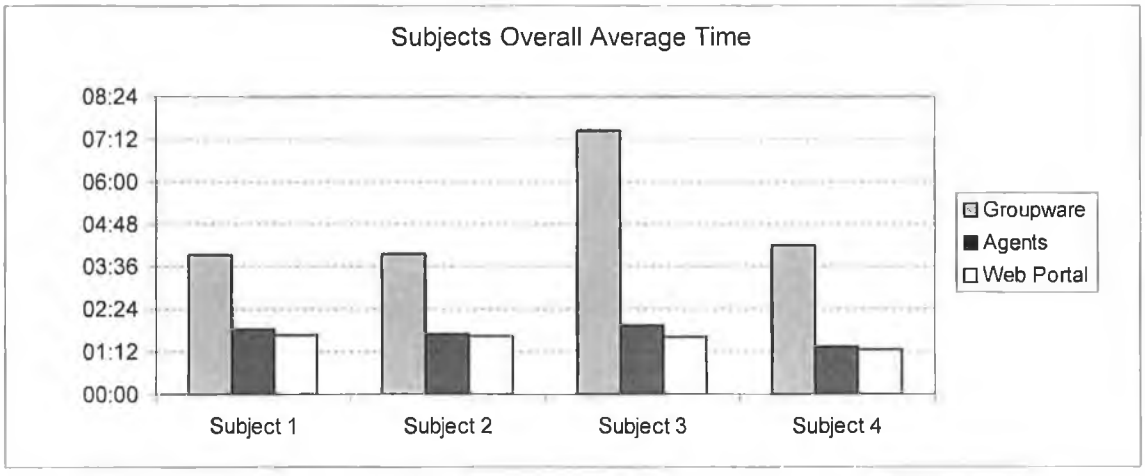
Figure 5.6. Time-Metric Charts for one of the subjects

Moreover, it should be noticed that there is a major difference between the periods of time invested by the user in the two systems tested as follows:

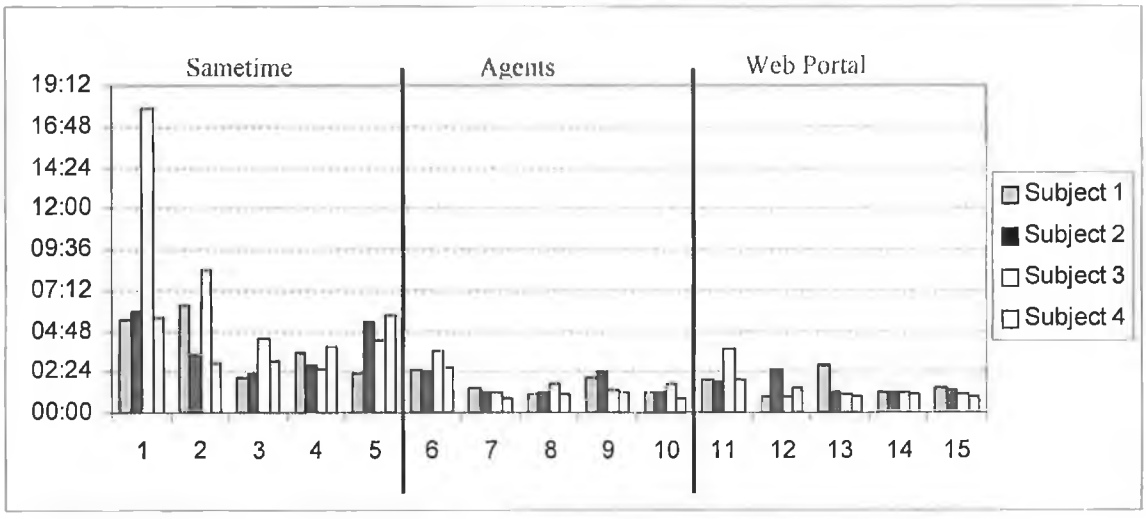
- Using *Sametime Document Repository* (and more than probably any other groupware technology or product data management system), the user is forced to dedicate the entire amount of time registered *using* the system and is not able to concentrate on anything else until the task is finished. For example, one Time-Metric Test subject spent 6 minutes and 20 seconds to extract information about a part called 'RearInterface' and those minutes required total focus and commitment on behalf of the user.
- Using the *MADIS system*, the user sets an objective for his/her Personal Agent (e.g. get me all information available on the part called 'RearInterface') and is announced by

an agent when an answer comes back. The response time can range from a few seconds to a few minutes (depending on the size of the knowledge base and the network speed) but this time can be valuably used by the designer for other activities. This demonstrates the great valuable potential of agent autonomy and pro-activeness.

Figure 5.7 shows the consistency on the times registered for all subjects that participated in the Time-Metric Test by showing the average retrieval times for each system and the overall time values recorded.



(a) Average Times registered in the Time-Metric Test



(b) Retrieval Times for all subjects in the Time-Metric Test

Figure 5.7. Overall Time Charts for the Time-Metric Test

Furthermore, the Time-Metric Test protocols revealed some usability problems regarding the user interface of some of the MADIS agents. For example, the Search Agent interface i.e. screen K(M2) should be easier to use as some subjects experienced difficulties in setting the search criteria (once the user learnt that the <ENTER> key has to be pressed in order for a criteria to be saved, no more similar problems were experienced). The Web

Portal browse page i.e. screen X(M1, M2, M3) should be more user-friendly as a number of subjects indicated (more through body posture and non-verbal codes than through actual verbalizations) that it is difficult to navigate through the list of parts using scroll buttons attached to a list. However, these usability problems need further testing and research, as the focus of this research was to propose a viable multi-agent design framework and not to create the best interface for the underlying components (see Chapter six for a more detailed discussion on the user interface in connection with future work). Moreover, the focus of the current testing procedure was not the evaluation of the graphical user interface of the MADIS Agents and Web Portal. Nevertheless, the observers were positively surprised by subject verbalizations such as *"The interface is nice, easy to navigate"* regarding the MADIS user interface.

To summarize, the results of the Time-Metric Test show that the MADIS agents represent a powerful infrastructure for design information management activities supporting the user in accessing the information needed for the current task in a timely manner.

5.3.2. The Collaboration Test

The Collaboration test involved two subjects (called *Designer A* and *Designer B*) simultaneously undertaking a series of tasks that required them to collaborate closely for successful completion. The subjects were distributed in two different locations with concurrent access to the same resources through a computer network. The main task of Designer A was to calculate the mass of a simple product called Smoke Alarm (see Figure 5.8) with the help of Designer B. For this reason, the mass information usually available in the MADIS ontology was intentionally omitted from the knowledge base for the Smoke Alarm assembly and all its subassemblies (only the parts - i.e. the components that can not contain any other components - had a mass).

Figure 5.8. The Smoke Alarm product used in the Collaboration Test

For a better understanding of the given tasks, Figure 5.9 presents the structure of the Smoke Alarm product.

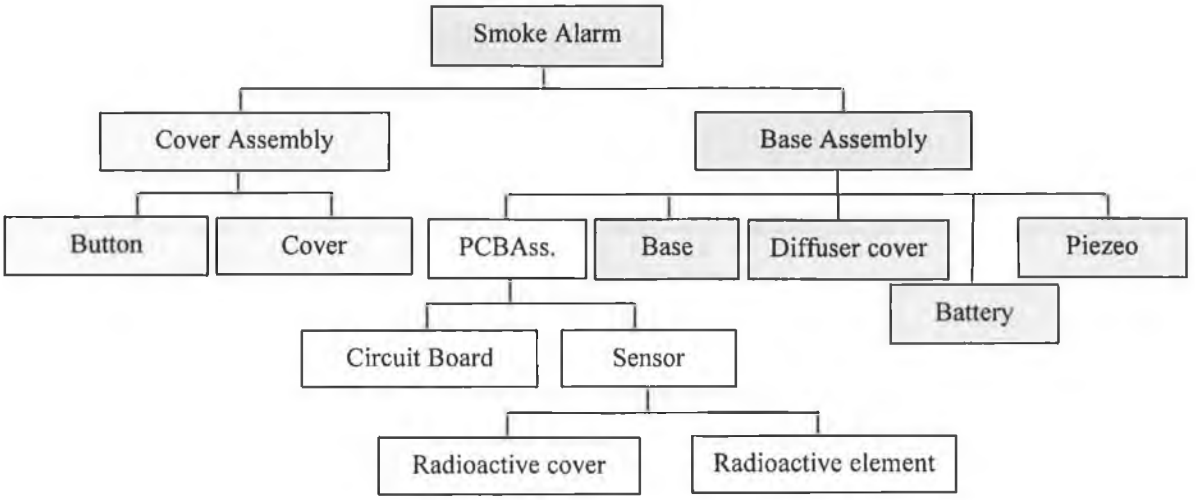


Figure 5.9. The Smoke Alarm structure (Bill of Materials)

One of the tasks of Designer B was to calculate the mass of one specific subassembly of the Smoke Alarm (i.e. PCBAssembly) and communicate it to Designer A. Furthermore, Designer B was responsible for the CoverAssembly (one of the Smoke Alarm's main subassemblies) component which he/she had to upload to the system when requested (see Appendix 5 for a full task description for both subjects).

The subject engineers were asked to undertake these tasks using first the Lotus Sametime Document Repository and then, after switching roles (under the observers' supervision), using MADIS. In both cases, the collaboration process was supported by the Lotus Sametime Meeting Room (see Figure 5.10). Previously created by the evaluators, the Sametime virtual meeting room allowed users to transmit real time audio and video, share applications and a whiteboard, and exchange instant messages (http://www.lotus.com).

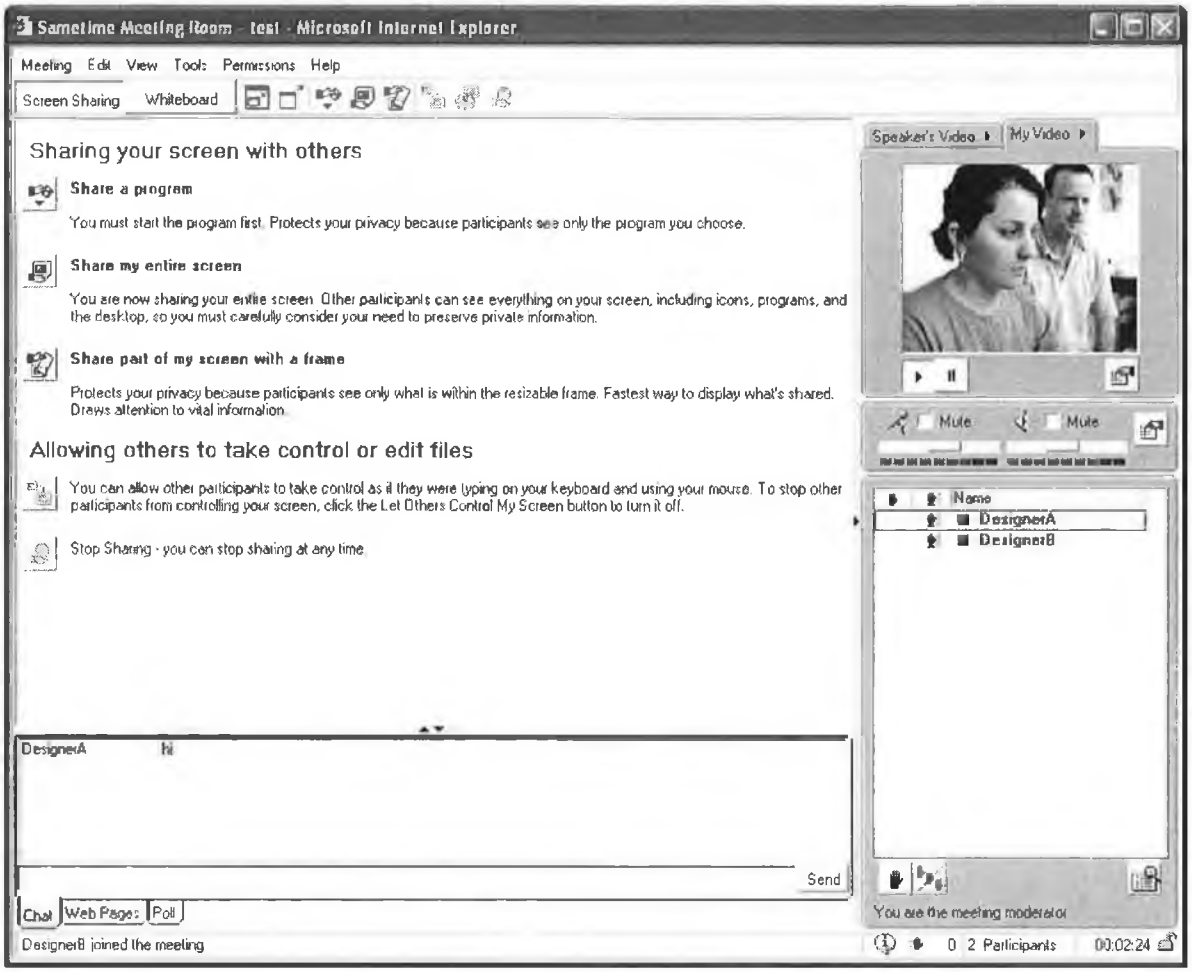


Figure 5.10. The Lotus Sametime Meeting Room

Furthermore, each designer had access to the MADIS system through a Personal Agent as well as a ProEngineer Application Agent. Being responsible for some missing parts requested by the tasks of Designer A, Designer B had to use the normal 'Upload Document' function when using the Sametime Document Repository to upload the corresponding spreadsheet for the omitted component. However, when using MADIS, Designer B had access to the ProEngineer CAD model of the missing part(s) and had to use the ProEngineer Application Agent to upload the new information to the MADIS knowledge base. Figure 5.11 presents the virtual environment used for the Collaboration Test.

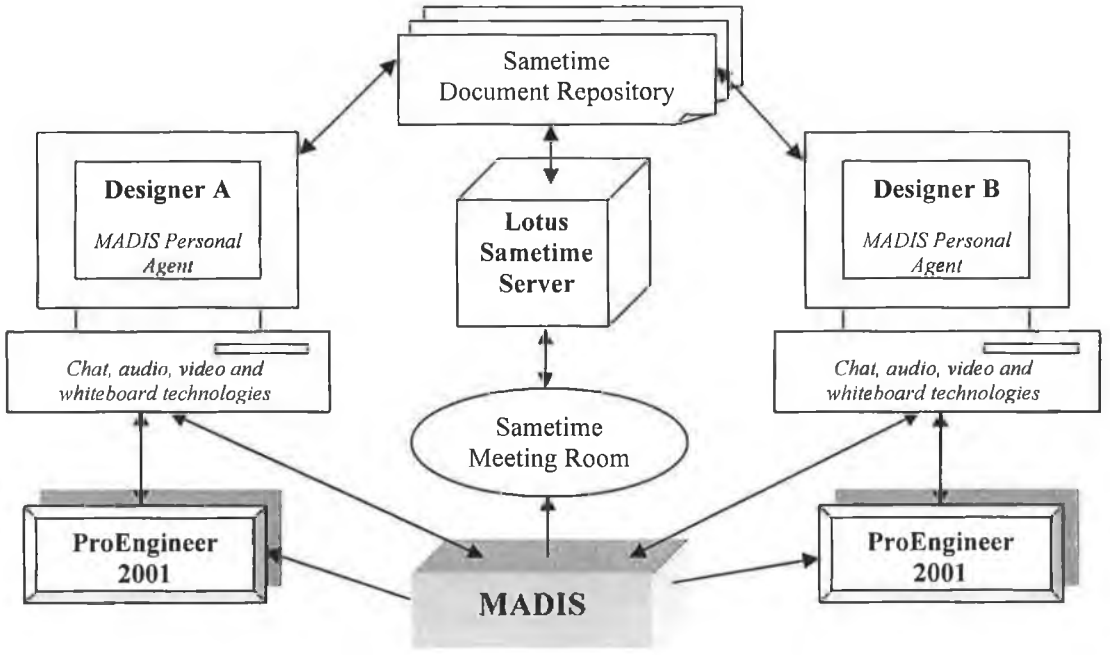


Figure 5.11. The environment of the Collaboration Test

An observer was present in the same room with the each of the users to monitor the subject's actions and behaviour.

Data Analysis

For each team that participated in the Collaboration Test, two transcripts (i.e. one referring to the Sametime Document Repository usage and the other referring to the MADIS system usage) have been created containing the observer's notes and the user's actions. This procedure was supported by the following two protocols recorded during the sessions:

1. *Communication protocol* – contains the communication episodes registered (mainly audio data).
2. *Mouse tracking protocol* – consists of the exact screens that had to be used by each subject in order to complete the given set of tasks.

Figure 5.12 shows the process flow of the protocol recording activity and depicts the main tasks of Designer A and Designer B and the interrelationships between them. The mouse tracking protocol is continuously recorded while the communication protocol was recorded whenever subjects used the audio/video tools or other technologies available in the virtual meeting room to collaborate. The Sametime Document Repository tasks are similar with the MADIS tasks but the way in which they are performed is naturally different. For example, when missing information has to be uploaded in tasks B2 and B3, Designer B will upload a local file using Sametime Document Repository as opposed to using the

MADIS Application Agent integrated in ProEngineer to upload part information to the MADIS system.

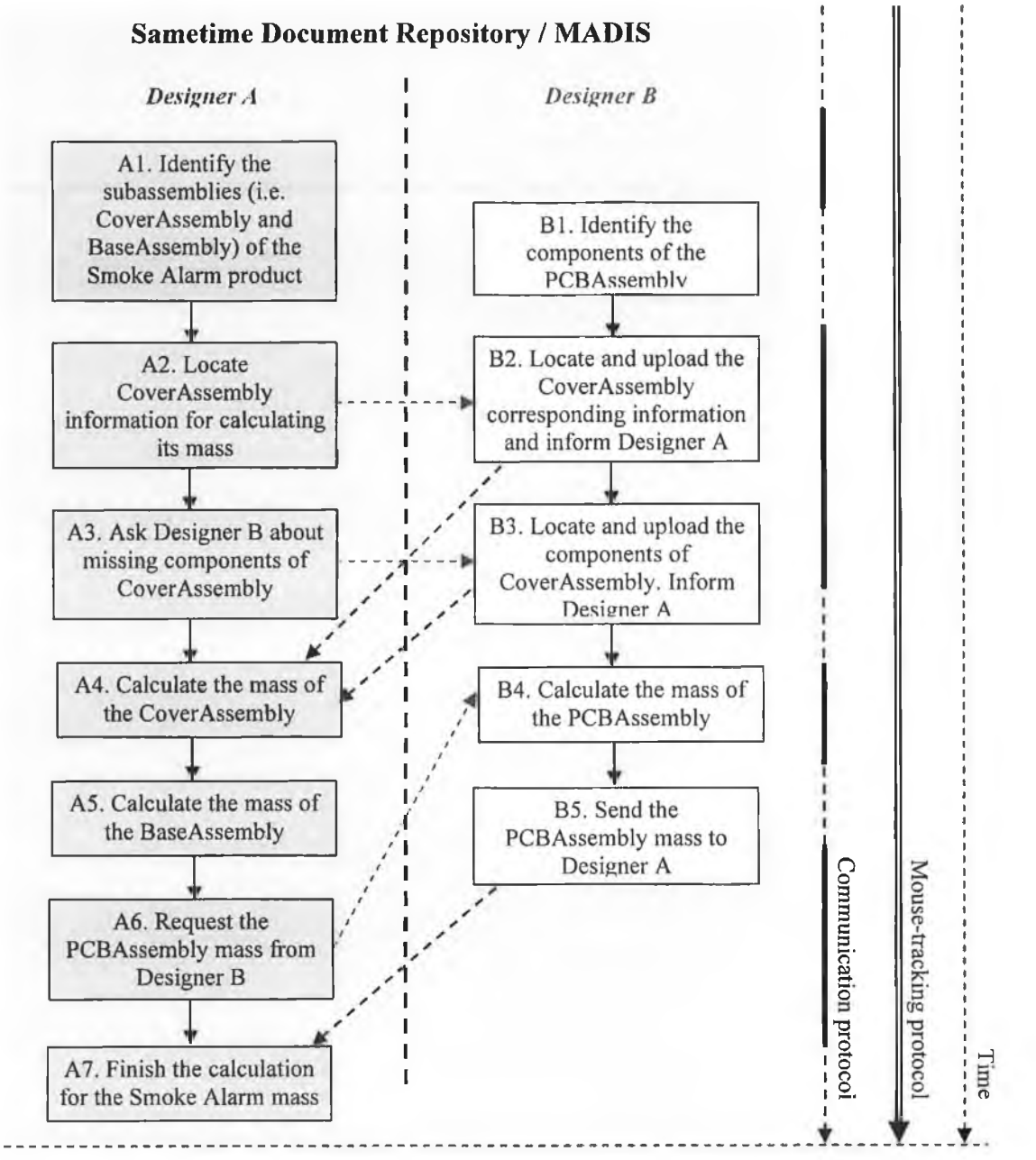


Figure 5.12. Collaboration Test: tasks and protocols

The process flow shown in Figure 5.12 was modelled for the screens necessary to complete the Collaboration Test tasks using the Sametime Document Repository (see Table 5.9) as well as for the MADIS screens required to complete the same tasks (see Table 5.10). In the same way in which the data was analysed for the Time-Metric Test, each screen was assigned a screen code e.g. screen S(A1, B1, A2, A3, A4, A6, B3) represents screen 'Product Browser Agent' from MADIS system that can potentially support tasks A1, A2,

A3, A4 and A6 assigned to Designer A and tasks B1 and B3 assigned to Designer B (as represented in Figure 5.12).

Screen Code	Screen Name	Steps
M(A1, A2, B1)	SameTime Document Repository – Product Assemblies	1. Identify the required assembly 2. Open the file by double-clicking
N(A1, B1)	Main Document View – Assembly file	1. Follow the assembly link 2. Extract information regarding the children of the assembly
O(A2, B2, A4, B3, B5)	SameTime Meeting Room	1. Communicate problems
P(B2, B3)	Document Upload	1. Choose "Create New Document" 2. Select type of document and specify subject 3. Select the file from local directory 4. Upload the file
Q(A3, A5, A7, B4)	SameTime Document Repository – Product Parts	1. Identify the required part 2. Open the file by double-clicking
R(A3, A5, A7, B4)	Main Document View – Part file	1. Follow the part link 2. Extract mass information

Table 5.9. Screens necessary to complete the Collaboration Test tasks using the Sametime Document Repository

Screen Code	Screen Name	Steps
R(A1, B1)	Designer Personal Agent	1. Request service
S(A1, B1, A2, A3, A4, A6, B3)	Product Browser Agent	1. Select required assembly/part 2. Extract information
T(A1, B1, A2, A3, A4, A6, B3)	Product Search Agent	1. Select Assembly/Part from concepts tree 2. Set search criteria 3. Click Search button
U(A1, B1, A2, A3, A4, A6, B3)	Product Search Agent - Search Results	1. Select required assembly/part from the result list 2. Extract information
V(A1, B1, A2, A3, A4, A6, B3)	Web Portal – Browse Page	1. Select Assembly/Part from the concepts tree 2. Select required assembly/part from the list 3. Extract information
W(A1, B1, A2, A3, A4, A6, B3)	Web Portal – Search Page	1. Select Assembly/Part from the concepts tree 2. Set search criteria 3. Click the Search button
X(A1, B1, A2, A3, A4, A6, B3)	Web Portal – Search Results Page	1. Click on required part from the result list 2. Extract information
Y(A2, B2, A5, B5)	Collaboration Meeting Room	1. Communicate with the other designer
Z(B2, B3)	Application Agent	1. Open CAD file in ProE 2. Save part information using IDIMS Application Agent

Table 5.10. Screens necessary to complete the Collaboration Test tasks using MADIS

Table 5.10 contains all possible choices the user can make using MADIS to perform the given tasks. Therefore, only some of the screens S, T, U, V, W and X will actually be

engaged by subjects depending on the MADIS component selected (i.e. Personal Agent or Web Portal) and furthermore on the service chosen (i.e. browse or search).

Using the mouse tracking protocol and the video capture of the subjects' face, each session was segmented based on the tasks and the screens used to complete those tasks. Therefore, the transcripts of both Sametime Document Repository and MADIS sessions resulted in the same number of episodes that represented the same sub-tasks making possible a detailed per-episode comparison. As an indication of collaboration efficiency, interpersonal communication was examined as a set of verbal and nonverbal codes such as facial expression, gaze, gestures, bodily movements, bodily posture, orientation and nonverbal aspects of speech (Hartley 1993; Chira 2002). Furthermore, the collaboration process between the two designers of each team was analysed using the behaviour categories in Interaction Process Analysis (Hartley 1993) i.e. shows solidarity, shows tension release, agrees, gives suggestion, gives opinion, gives orientation, asks for suggestion, asks for opinion, asks for orientation, disagrees, shows tension, shows antagonism. During each episode, the following seven categories were measured for each participant:

1. Gives suggestion/opinion/orientation (GS)
2. Asks for suggestion/opinion/orientation (AS)
3. Agrees (A)
4. Disagrees (D)
5. Shows solidarity (S)
6. Shows tension (T)
7. Shows tension release (TR)

These seven behaviour categories were identified through verbal or nonverbal codes observed in the communication protocol and the video tape of each subject working on the scenario.

Supported by all the protocols registered, the following transcripts have been created for each team of designers participating in the Collaboration Test:

(1) A transcript of the Sametime Document Repository PA session (see Table 5.11)

(2) A transcript of the MADIS PA session (see Table 5.12)

Observer's notes:

- The two video cameras used in the testing were completely ignored by subjects
- Confusion and irritation was observed throughout the scenario
- Audio, instant messaging and white board were used
- In some cases, subjects did not follow the exact instructions given e.g. delegation of work by Designer A, Designer B computing the total mass for the Smoke Alarm instead of Designer A

No	Time Start	Duration	Give suggestion/opinion/orientation (GS) Ask for suggestion/opinion/orientation (AS) Agree (A) Disagree (D) Show solidarity (S) Show tension (T) Show tension release (TR)		Notes
			Designer A	Designer B	
1	0:00	5:02		1T	A - Identify SmokeAlarm subassemblies; B – PCBAssembly mass
2	5:02	3:24	5As; 1A; 1S; 2T	2Gs; 1As; 1A;1S	A asks for CoverAssembly using chat first but then ONLY audio A - Extracts mass information; A - Frustration through non-verbal codes; B – Uploads requested document i.e. CoverAssembly
3	8:26	4:23	3AS; 1GS; 3A; 2T; 1TR; 2D; 1S	1GS; 2AS; 2T; 1TR; 1S	B informs A about the location of CoverAssembly; B - Confusion; A- Frustration; Observer at 12:20
4	12:49	3:10	1AS; 1S; 1T; 1TR	3Gs; 2As; 1A; 4S; 2T; 2TR	A asks for CoverAssembly components; Confusion; A – *"I think we're loosing time here"* B uploads Button & Cover A, B – frustration; A asks B again about uploaded docs B – Observer at 14:00
5	15:59	1:21	2AS; 3S; 1GS	2GS; 1AS	A starts to compute SmokeAlarm mass A asks B for PCBAssembly mass B informs A with the PCBAssembly mass A finishes the task

Table 5.11. Transcript of the Sametime Document Repository PA session of the Collaboration Test for one of the teams

Observer's notes:

- The two video cameras used in the testing were completely ignored by subjects
- Audio technology was mainly used for communication
- Subjects were generally relaxed
- Both the Web Portal and the Agents were engaged to support the task performance; Some subjects relied more on their Personal Agent while others preferred the interface of the Web Portal.
- The search service was preferred to the browse
- The tasks were easier to complete using the MADIS system

No	Time Start	Duration	Give suggestion/opinion/orientation (GS) Ask for suggestion/opinion/orientation (AS) Agree (A) Disagree (D) Show solidarity (S) Show tension (T) Show tension release (TR)		Notes
			Designer A	Designer B	
1	0:00	3:01			A identifies Smoke Alarm components using Search Agent first but then Web Portal – browse B uses Search Agent to identify PCBAssembly components
2	3:01	1:28	1AS	1A	A asks B for PCBAssembly mass A calculates mass for BaseAssembly B calculates mass for PCBAssembly
3	4:29	4:33	2GS; 2AS;2S	5GS; 2AS; 2S	A asks for CoverAssembly information using audio & chat

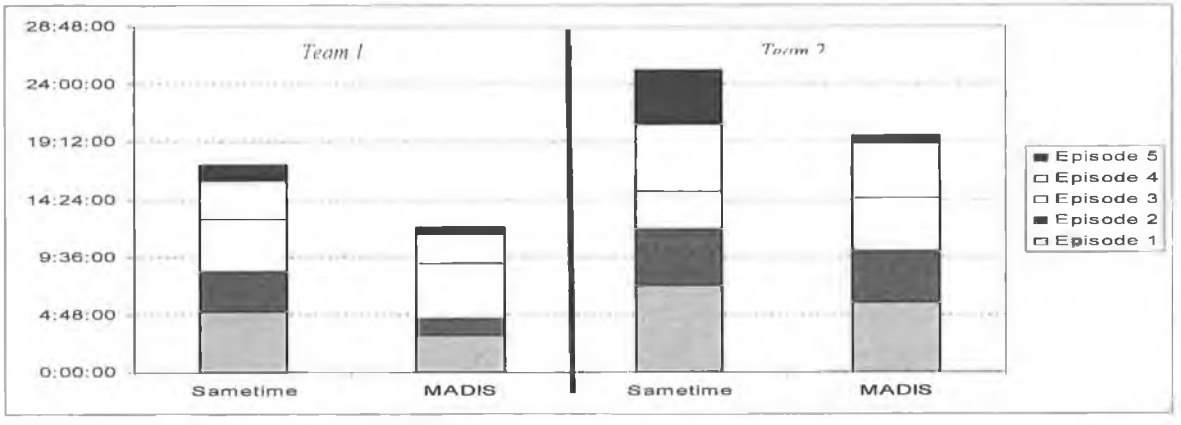
					A waits for B to reply B saves Button & Cover parts from ProE using Application Agent
4	9:02	2:30	1AS	1S	A calculates the CoverAssembly mass A – smile, happy face, relaxation (body posture) B continues mass calculation for PCBAssembly using Web Portal – search A reminds B about the previously requested PCBAssembly mass
5	11:32	0:35	1S	1GS	B communicates the PCBAssembly mass A finishes the task of calculating the SmokeAlarm mass

Table 5.12. Transcript of the MADIS PA session of the Collaboration Test for one of the teams

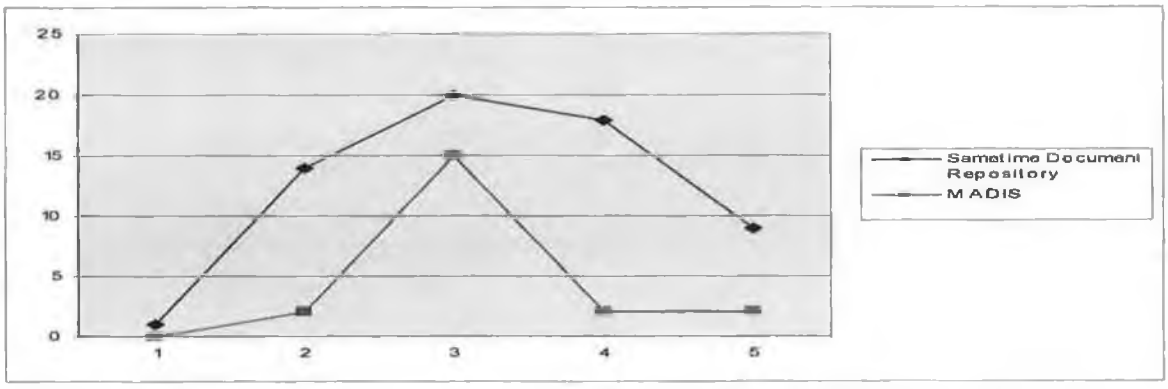
Each transcript contains the observer's notes, the segmentation of the episodes, a description for each episode and the number of occurrences for each behaviour category identified earlier. Appendix 6 contains a complete list of the transcripts built for the Collaboration Test.

Results

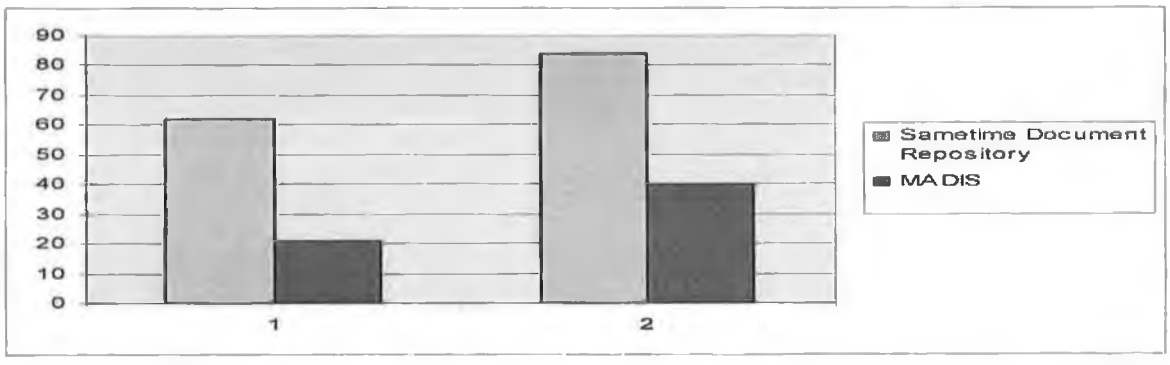
The Collaboration Test showed that the same tasks were more difficult to be performed using the Sametime Document Repository than using MADIS. This is not only indicated by the longer time took to complete each episode using Sametime (see Figure 5.13-a) but also by the higher number of behaviour categories registered during the Sametime session (see Figure 5.13-b,c).



(a) Episode times for each team in Sametime and MADIS sessions



(b) Communication measurement in each episode for one of the teams



(c) Overall Communication Chart for each team

Figure 5.13. Time and Communication Charts relative to team

Each behaviour category measured was generally more frequent in the Sametime sessions than in the MADIS sessions. Figure 5.14 exemplifies this very well by presenting all the codes registered during each similar episode for each member of the team (e.g. GS-a means behaviour category 'Give suggestion/opinion/orientation' used by Designer A, D-b means 'Disagree' on behalf of Designer B).

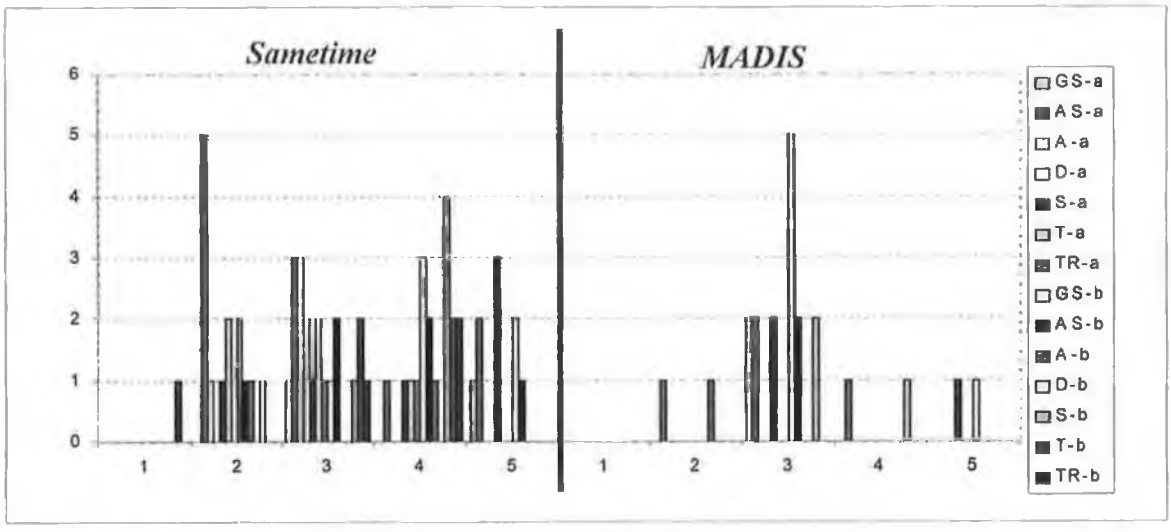


Figure 5.14. Behaviour categories for each episode in Sametime and MADIS

During the Sametime session, more communication was needed to clarify technological issues and to support the collaboration aspect of the tasks. This means that the efficiency of the collaboration process was higher during the MADIS session mainly because the software used by subjects facilitated and meaningfully supported communication through readily access to required information. The best example supporting this result refers to tasks A3, B2 and B3 requiring screens P and Q in Sametime Document Repository and mainly screen Z in MADIS. To upload a document in Sametime, the user had to locate the file on his/her local machine, provide a description for the new document, specify the virtual location and upload the file. This process was not only time-consuming but also unreliable for the other member of the team who was still unable to find a correctly uploaded document (e.g. *"Where did you upload the Button?"*). Using the MADIS Application Agent (see Figure 5.15) on the other hand, the collaboration process was well supported by the common ontological knowledge base instantly updated when the user selects the appropriate option.

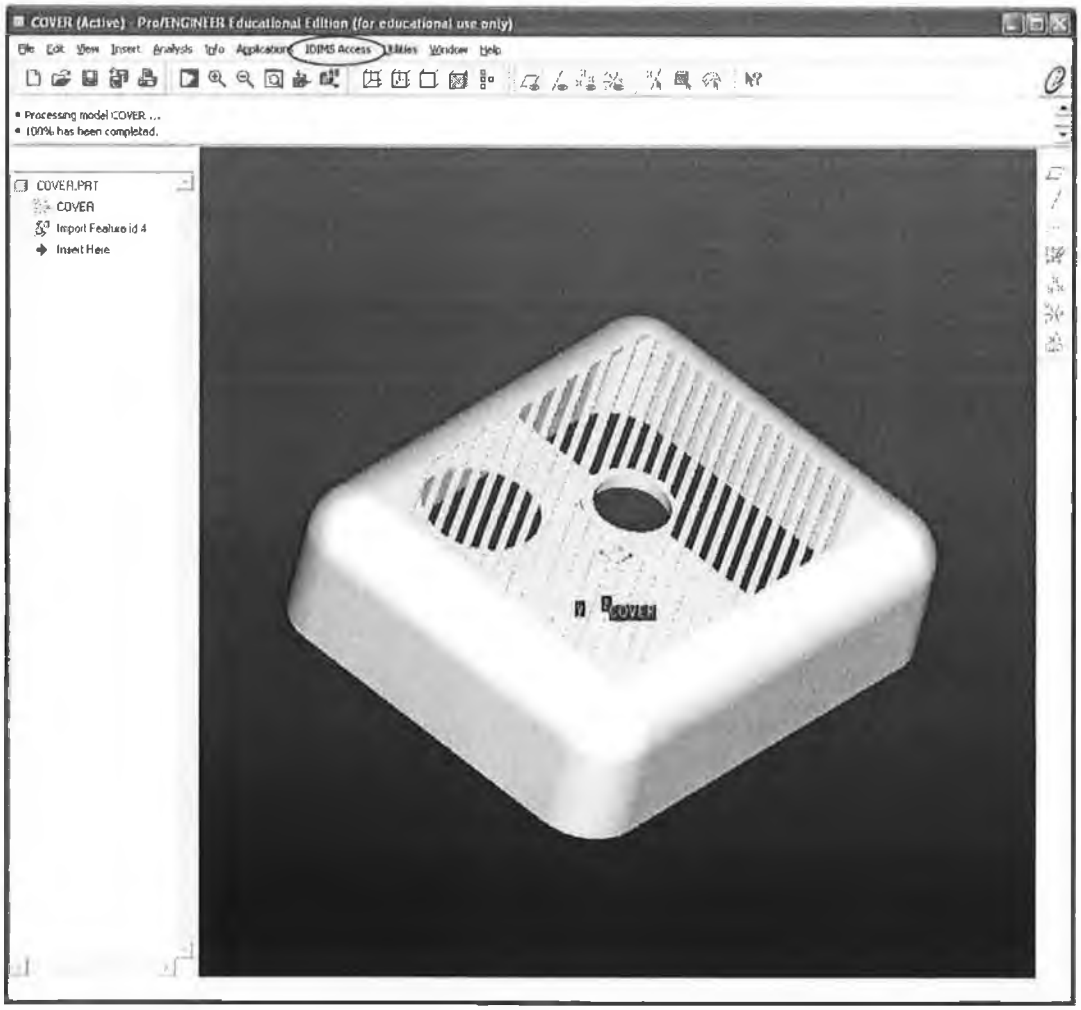


Figure 5.15. The Application Agent in ProEngineer

Having the CAD file of the requested component opened in ProEngineer, the Application Agent extracts all relevant information from the CAD model and forwards it for storage purposes to the Ontology Agents. This way, the other member of the team has instant access to the new updated information through his/her Personal Agent or the Web Portal (e.g. *"Ok, I see CoverAssembly components now!"*). Autonomy, cooperation and pro-activeness are the agent properties implemented in MADIS that facilitate the above scenario. Figure 5.16 presents the MADIS Web Portal interface before and after new information has been added using the Application Agent.

Consistent with the Time-Metric Test results, the confusion and frustration observed during the Sametime Document Repository sessions were replaced by relaxation and confidence in the MADIS sessions.

During the MADIS sessions, some users preferred the friendlier interface of the Web Portal while others showed more confidence in the MADIS Agents considering the web interface unreliable. Both browse and search services were engaged by subjects but the search one proved to be more efficient in most instances. Although outside the focus of the current testing, some usability issues mainly regarding the agent interfaces were discovered e.g. search criteria is hard to set in certain circumstances, browse interface is not easy to use mainly due to the lack of intelligent structure. However, as already indicated in the Time-Metric Test results, more in-depth research is necessary to address these problems.

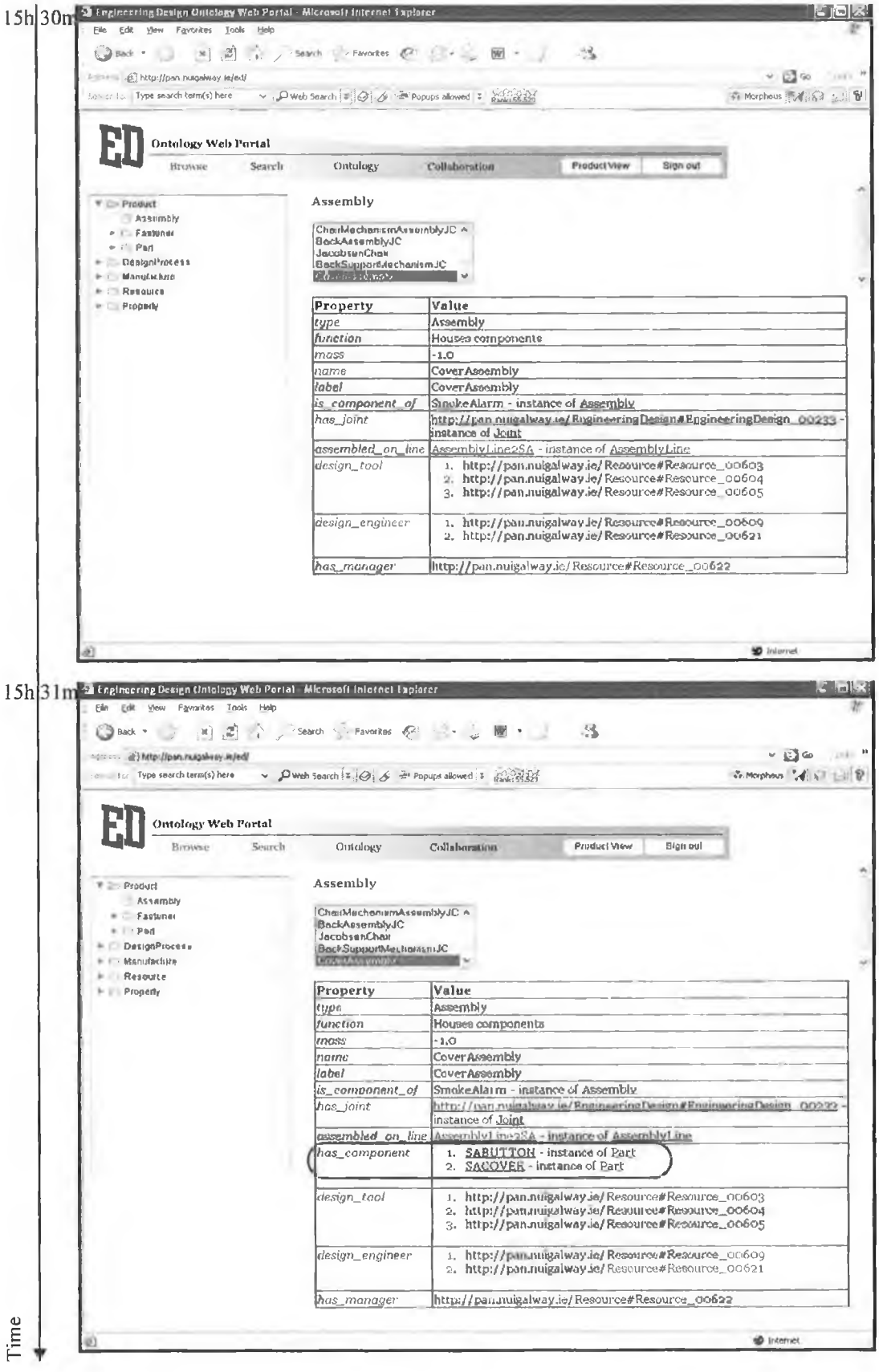


Figure 5.16. The MADIS Web Portal – browse page seen by Designer A before and after the parts (i.e. SAButton and SACOver) of the CoverAssembly have been saved by Designer B

5.3.3. Feedback

The PA testing phase of MADIS was completed with a feedback report completed by each subject where opinions and suggestions were collected. Table 5.13 summarizes the main questions and answers collected during the feedback phase.

Question	Answer
Did you feel constrained in any way by the video camera?	None of the subjects were constrained by the video cameras.
Was the MADIS system (i.e. MADIS agents and MADIS web portal) useful in supporting your task? What is your general opinion about MADIS?	All subjects found the MADIS system very useful in supporting their task. Both the Web Portal and the Agents were used, some subjects preferring one to the other.
Did you feel restricted in any way by the communication technology (i.e. Sametime Meeting Room) used?	Overall, the communication technology supported the collaboration process. The audio technology was mainly used but, in some instances, subjects found the audio quality poor. Most subjects considered that the video was not really necessary.
Rate the collaboration process between you and the other member of your team on the following scale.	Average 6 (on a scale 1 to 7, where 1=very poor and 7=very good).
Rate the ease of use of the Sametime Document Repository on the following scale.	Average 5 (on a scale 1 to 7, where 1=very easy and 7=very difficult)
Rate the ease of *learning* of the MADIS *agents* on the following scale.	Average 2 (on a scale 1 to 7, where 1=very easy and 7=very difficult)
Rate the ease of *use* of the MADIS *agents* on the following scale.	Average 2 (on a scale 1 to 7, where 1=very easy and 7=very difficult)
Rate the ease of *learning* of the MADIS *web portal* on the following scale.	Average 2.5 (on a scale 1 to 7, where 1=very easy and 7=very difficult)
Rate the ease of *use* of the MADIS *web portal* on the following scale.	Average 1.5 (on a scale 1 to 7, where 1=very easy and 7=very difficult)
Any comments/suggestions on the *browse* service provided by the MADIS *agents*.	The browse service provided by the Agents was considered reliable. However, some subjects found it time consuming mainly because there is not enough information structure.
Any comments/suggestions on the *search* service provided by the MADIS *agents*.	Subjects found the search service provided by the Agents much better than the browse. However, the GUI should be improved.
Any comments/suggestions on the *browse* service provided by the MADIS *web portal*.	The browse service provided by the Web Portal was better appreciated than the one provided by the Agents mainly because information was better linked. Some subjects thought that the browse service is useful only if search criteria can not be found.
Any comments/suggestions on the *search* service provided by the MADIS *web portal*	The search service provided by the Web Portal was considered very useful. Some subjects felt that improvements can be made e.g. GUI, the need for case-sensitive search.

Table 5.13. Feedback Results

The feedback report shows that the subjects involved in this testing phase understood the MADIS environment and valued the services offered by the Agents and the Web Portal. Asked to rate different aspects of MADIS on a 1 to 7 scale (see Table 5.13), participants indicated that the MADIS Agents and Web Portal were relatively easy to use for searching and retrieving information (MADIS was averagely rated 2 as opposed to 5 for Sametime Document Repository, where 1 means very easy and 7 means very difficult). Furthermore,

subjects were more relaxed and less frustrated using MADIS. However, the user interface needs improvement and the ontology structure can be enhanced. Also, a collaboration technology should be more integrated in MADIS and more functionality can be implemented into the system e.g. print function.

The feedback results are consistent with the Time-Metric and Collaboration Tests results demonstrating the great potential of MADIS to significantly improve the efficiency of the design process in a distributed collaborative engineering design environment.

5.4. Conclusions

Based on the protocol analysis technique, the testing phase of MADIS involved subjects with an engineering background working on their own and as part of a distributed team. The data analysis phase of the resulted protocols offered rich in-depth information about the benefits and the limitations of the proposed MADIS software system. Supporting readily access to distributed design knowledge, MADIS was efficiently employed by subjects in performing the given tasks. The collaboration process between dispersed designers was meaningfully facilitated by MADIS through agent cooperation and the common ontology library. Compared to traditional groupware technologies (e.g. Sametime Document Repository), the multi-agent approach has clear potential benefits including reliability, robustness, faster access to required information. The PA test results showed that agent properties such as autonomy, pro-activeness, cooperation and mobility are highly beneficial to the distributed designer during the information-intensive problem solving process. Some limitations of MADIS have also been revealed by this testing phase. The graphical user interface of the user-dedicated agents needs improvement (however, usability issues are outside the focus of the current research and testing). Besides usability problems, MADIS functionality lacks important features which should be researched and implemented in a future version (see Chapter six for more details).

Offering computational efficiency, dependability and flexibility, multi-agent systems coupled with ontologies represent a promising approach to support the design process in a distributed collaborative design environment facilitating interoperation among distributed resources, interdisciplinary cooperation and information sharing.

References

Benbunan-Fich, R. (2001). "Using protocol analysis to evaluate the usability of a commercial web site." Information & Management.

Branch, J. L. (2000). "Investigating the Information-Seeking Processes of Adolescents: The Value of Using Think Alouds and Think Afters." Library & Information Science Research.

Chan, C. S. (2000). "An examination of the forces that generate a style." Design Studies.

Chira, C. (2002). Design, Development and Testing of a CAD Integrated Design for Environment Software Tool. Galway, Galway Mayo Institute for Technology.

Cross, N. and A. C. Cross (1995). "Observations of teamwork and social process in design." Design Studies 16(2): 143-170.

Ericsson, K. A. and H. A. Simon (1999). Protocol Analysis: Verbal Reports as Data, The MIT Press.

Gero, J. S. and T. McNeil (1997). "An approach to the analysis of design protocols." Design Studies.

Gero, J. S. and H. H. Tang (2001). "The differences between retrospective and concurrent protocols in revealing the process-oriented aspects of the design process." Key Centre of Design Computing and Cognition.

Goldschmidt, G. (1995). "The designer as a team of one." Design Studies.

Greenberg, S. (1996). "Teaching Human Computer Interaction to Programmers." ACM Interactions 3(4): 62-76.

Hartley, P. (1993). Interpersonal Communication, Routledge.

Henderson, R., J. Podd, M. Smith and H. Varela-Alvarez (1995). "An examination of four user-based software evaluation methods." Interacting with computers.

http://www.lotus.com, Lotus Sametime, Last Accessed August 2005.

Roche, T. (1999). Development of a Design for the Environment Workbench. CIMRU, Industrial Engineering Dept. Galway, UCG.

Chapter 6

Conclusions and Future Work

6.1. Thesis Summary

This thesis proposes a Multi-Agent Design Information Management and Support System (MADIS) to address the key information needs of distributed collaborative engineering design. The structural and functional development of MADIS is presented by describing the phases of system specification, design, implementation and testing. This process is supported and informed by the research carried out over a period of more than three years in the following areas (see Figure 6.1):

- *Distributed engineering design*: definition, characteristics, benefits, problems and current trends in software support.
- *CAD systems*: area of integration of software agents into virtual prototyping environments through the use of Application Programming Interfaces.
- *Multi-agent systems*: definition, properties, typologies, architectures, benefits, coordination, negotiation, communication, trust, standards, methodologies, languages, toolkits and applications.
- *Ontologies*: definition, typologies, methodologies, languages, editor tools and applications.
- *Programming languages and paradigms*: the selection of appropriate software tools and architectural models for the development of software agents, CAD integrated components and web interfaces.
- *Protocol Analysis*: the application of the protocol analysis technique in the testing and validation of distributed software systems.

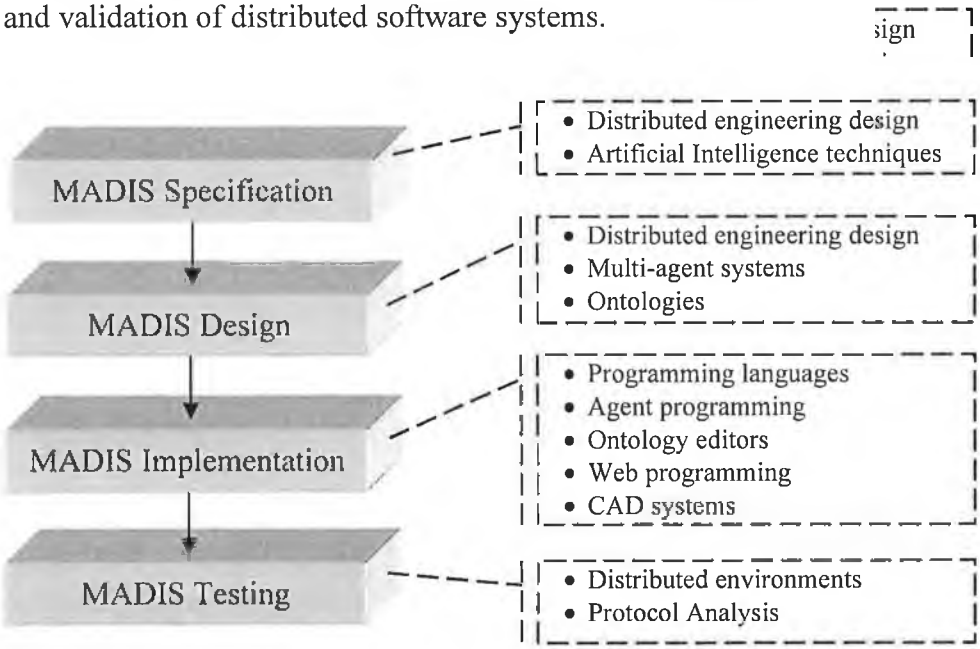


Figure 6.1. Thesis research areas

The system specification is based on a comprehensive review of the distributed engineering design domain and an analysis of the main problems engineers have during the process of design in a distributed virtual and computer based environment. It has been shown that distributed engineering design is characterised by dispersed human and physical resources (from a geographical, temporal, functional and semantic perspective), computer-based cooperation and highly heterogeneous design teams and tools. The main problems that need to be addressed include the big volume and dispersion of design data, information and knowledge, lack of cooperation support, limited awareness and low integration (see Chapter two). Based on these problems and the examination of other approaches to offer computational support for distributed design, a set of initial solution requirements and a high level view of the MADIS architectural framework are presented at the end of Chapter two.

MADIS employs multi-agent systems supported by an ontology library in order to tackle important distributed design issues such as interdisciplinary cooperation among distributed designers, exchange of design data, information and knowledge and integration of heterogeneous software tools. The system design phase is supported by an extensive review of software agents and multi-agent systems (see Chapter three). The four proposed agent societies (i.e. user management, application management, ontology management and agent interconnection and management) that form the MADIS system are formally described using the AUML methodology (see Chapter four). Compliant with the FIPA specifications, the MADIS prototype implementation demonstrates the functionality of the proposed architectural model (see Chapter four).

Finally, the testing and validation phase of MADIS uses the protocol analysis technique to evaluate the system in a distributed virtual environment (see Chapter five). The system is compared with traditional groupware technologies in similar design scenarios with the focus on information retrieval times and collaboration efficiency.

The current chapter presents the conclusions and the contributions of this thesis and suggests a set of recommendations for further research and development of computational infrastructures to support distributed engineering design.

6.2. Research Results and Conclusions

The initial overall objective of the current research was to design, implement, test and validate an intelligent system for distributed and collaborative engineering design. As already indicated in the previous section, this thesis successfully delivers a multi-agent

design information support system presenting the defining phases of design, implementation and testing. However, the author believes that the intelligence dimension proposed as part of the initial objective was not achieved as this is probably still a very high (if not unattainable) aim for practical environments. Nevertheless, the proposed MADIS multi-agent system employs cooperating agents that can support the user through learning, autonomous agents for information retrieval, mobile agents to address various designer needs, web interfaces for easy access to design knowledge and ontologies for semantic management of design information structures. This means that MADIS is characterised by important properties including autonomy, cooperation, mobility, flexibility and learning but they do not necessarily translate to intelligence.

The research results can be summarized along the initial set of objectives (presented in Chapter one) as follows:

- *Objective 1: Research distributed engineering design in terms of definition, characteristics, potential benefits and problematic aspects.*
  - ✓ Distributed engineering design involves multidisciplinary design teams dispersed across the enterprise that have to cooperate in a computer-based medium in order to identify the 'optimal' solution to the current design problem.
  - ✓ Distributed engineering design aims to achieve benefits such as savings in project life-cycle and costs, added value to team efforts, access to a comprehensive knowledge-based system, reliable communication among design teams and members, flexible access and retrieval of information and timely connectivity with global experts.
  - ✓ The main distributed engineering design issues refer to information related problems, coordination and communication problems, knowledge sharing problems and information technology support.
- *Objective 2: Investigate the current approaches to support the process of design in a distributed environment.*
  - ✓ Many of the relevant research studies indicate that the complex activity of distributed cooperative design may be effectively supported by the provision of a collection of interacting autonomous software components incorporating AI specific problem-solving mechanisms.
  - ✓ Software agents and multi-agent systems represent a potential successful solution for distributed design issues such as interdisciplinary collaboration,

sharing of diverse and irreducible representations of design data, information and knowledge and integration of heterogeneous software tools.

- *Objective 3: Review state-of-the-art AI technologies including software agents, agent-based systems, multi-agent systems, ontologies and semantic web.*
  - ✓ The current thesis identifies an agent as a software system situated in an environment that autonomously acts on behalf of its user and is able to cooperate with other agents and/or humans in order to accomplish its objectives.
  - ✓ Agents and multi-agent systems deliver techniques to manage the complexity inherent in software systems and appropriate to domains in which data, control, expertise and/or resources are inherently distributed.
  - ✓ Ideal for solving complex problems with multiple solving methods, perspectives and/or problem solving entities, multi-agent systems present many potential advantages including robustness, efficiency, flexibility, adaptivity, scalability, inter-operation of multiple existing legacy systems, enhanced speed, reliability and extensibility.
  - ✓ Ontologies specify content specific agreements to facilitate knowledge sharing and reuse among systems that submit to the same ontology.
  - ✓ Emerging as the next generation of World Wide Web (where data is defined and linked in such a way that it can be used by people and processed by machines), the Semantic Web is considered as a potential application environment for various MADIS agents in the context of future work.
- *Objective 4: Specify and analyse the requirements of a computational system intended to support distributed engineering design.*
  - ✓ The MADIS specification phase delivers the definition and analysis of the system requirements i.e. management of design data, information and knowledge, cooperation support, facilitation of knowledge sharing and reuse, integration of distributed resources.
  - ✓ It is intended to design a multi-agent system consisting of several interacting agent societies and enabled by an ontology library in order to meet the system requirements.
- *Objective 5: Design the architectural framework of the proposed MADIS system (e.g. architectural components, role, structure, properties, interrelationships, enabling technologies).*

- ✓ Supported by the AUML methodology, the MADIS design phase defines and characterizes the architectural components, roles, structure and properties of each MADIS agent society.
- ✓ The user agent society represents the interface between MADIS and the designer providing different services to the user and responding to queries and events initiated by the user (or on behalf of the user) with the help of the ontological agents.
- ✓ The application agent society contains various application integrated agents that autonomously (or semi-autonomously) retrieve information from the applications used by the designer and forward it for storage to the ontological agents.
- ✓ The ontology agent society provides ontology management services.
- ✓ The interconnection agent society manages the cooperation process among other agents based on the needs and the services advertised by them.

- *Objective 6: Develop a MADIS prototype by implementing the main components of the proposed system (proof-of-the-concept model).*
  - ✓ The main architectural components of MADIS have been implemented in a working prototype model using the Java programming language and the JADE framework.
  - ✓ Complying with the FIPA specifications, MADIS agents are able to communicate by exchanging messages using the FIPA Agent Communication Language.
  - ✓ A MADIS Web Portal has also been developed to facilitate flexible intra/internet access to the information structures managed by the system.
  - ✓ Representing for the most controversy of this research, a minimal ontology library has been developed to support the functionality of the multi-agent system.
- *Objective 7: Test and validate the proposed architecture and system.*
  - ✓ The MADIS prototype was successfully tested in a distributed virtual environment using the protocol analysis technique.
  - ✓ The results of the analysed protocols show that MADIS supports readily access to distributed design knowledge being efficiently employed by subjects in performing the given tasks.
  - ✓ Although some usability problems have been detected, the multi-agent approach has clear potential benefits including reliability, robustness and faster

access to required information essential for the distributed designer during the information-intensive problem solving process of design.

The research and development work supporting the current thesis indicates that a multi-agent system consisting of agents characterised by autonomy, pro-activeness, cooperation, learning and/or mobility can efficiently support the process of distributed design by facilitating interoperation among distributed resources, interdisciplinary cooperation and information sharing. The provision of an ontology library is important for content-related support of information exchange. Referring to MADIS, the development of the ontology was considered outside the main focus of the research and, therefore, used minimal human and time resources. Nevertheless, a viable MADIS ontology library had to be developed to enable and demonstrate the capabilities of the proposed multi-agent system in a real engineering design environment.

Although the protocol analysis testing and validation phase of MADIS offered rich important results, the author considers that a real distributed design environment would have been more beneficial to the evaluation of MADIS. However, a MADIS testing phase in the engineering department of an actual company would necessitate previous system setup on several machines including a server machine, system running for a certain period of time before actual testing (the ontology library becomes populated with different product instances over time) as well as full access to information through various applications used by the designer.

6.3. Contributions

The main contribution of the current thesis refers to the application of software agents to the domain of distributed engineering design through the development of a multi-agent design information management and support system. As already indicated in the previous section, this development process includes the phases of system design, implementation and evaluation. More specifically, the following steps (reflecting both the research and the thesis structure) summarize the main achievements of the current work:

- The examination of the distributed engineering design domain and the specification and analysis of system requirements.
- The identification and design of the multi-agent societies (i.e. user agent society, application agent society, ontology agent society and interconnection agent society) that form the proposed system and the definition of inter-agent interactions.

- The implementation of a FIPA-compliant prototype of the proposed system that includes personal designer agents with a graphical user interface, application agents integrated in a CAD environment frequently used by engineers whilst in the design process, system manager agents, directory facilitator agents, web-based components, ontology library and cooperation platforms.
- The evaluation of the proposed system through a system assessment in the context of other existing AI approaches to distributed engineering design and through a testing and validation phase of the implemented prototype using the protocol analysis technique.

The employment of multi-agent systems to support the process of design in a distributed engineering environment offers robustness, reliability and flexibility. Agents characterised by autonomy, pro-activeness, cooperation, learning and/or mobility can efficiently address distributed design issues such as the integration of heterogeneous software tools, interdisciplinary cooperation among distributed designers and exchange of design data, information and knowledge. Enabled by agents and ontologies, the proposed multi-agent design information management and support system facilitates the management of the data-information-knowledge value chain aiming to ultimately improve engineering design operation and management.

6.4. Recommendations for Future Work

More research and development work is necessary to improve and extend the functionality of the proposed MADIS computational infrastructure towards an intelligent system. Further research into all the areas (e.g. distributed engineering design, multi-agent systems, ontologies, programming languages and paradigms, protocol analysis) covered by the current thesis can always trigger more and better results. Furthermore, the author believes that these results should be completed with the study of the following fields (see Figure 6.2):

- *Human Designer (Kolb 1984; Lawson 1990; Baya 1996; Brennan 1996; Petroski 1996; Cross 1999; Nakakoji, Yamamoto et al. 1999; Roche 1999; Gero 2000; Bal and James-Gordon 2001; Hnug and Der-Thang 2001; Karuppan 2001; Kruger and Cross 2001; Shneiderman and Hochheiser 2001; Lawson, Bassanino et al. 2003)*: the study of the cognitive active of the human designer, the representation of the designer profile, the examination of elements that foster understanding, learning and creativity.

- *Human-Computer Interaction (Baecker and Buxton 1987; Shneiderman 1992; Brennan 1996; Salvendy 1997; Zhang and Li 2004; Lubart 2005)*: the design and development of intelligent user interfaces, the presentation of information and knowledge for the accommodation of different designers with different learning strategies, the visualization of knowledge to promote designers' creativity.
- *Semantic Web (Berners-Lee 1998; Decker, Harmelen et al. 2000; Fensel 2000; Ramsdell 2000; Berners-Lee, Hendler et al. 2001; Dumbill 2001; Swartz and Hendler 2001; Benjamins, Contreras et al. 2002; Hendler, Berners-Lee et al. 2002)*: the exploration of the new generation of Web for the benefit of distributed engineering design, the development of semantic tools to explore the information available in wide area networks.

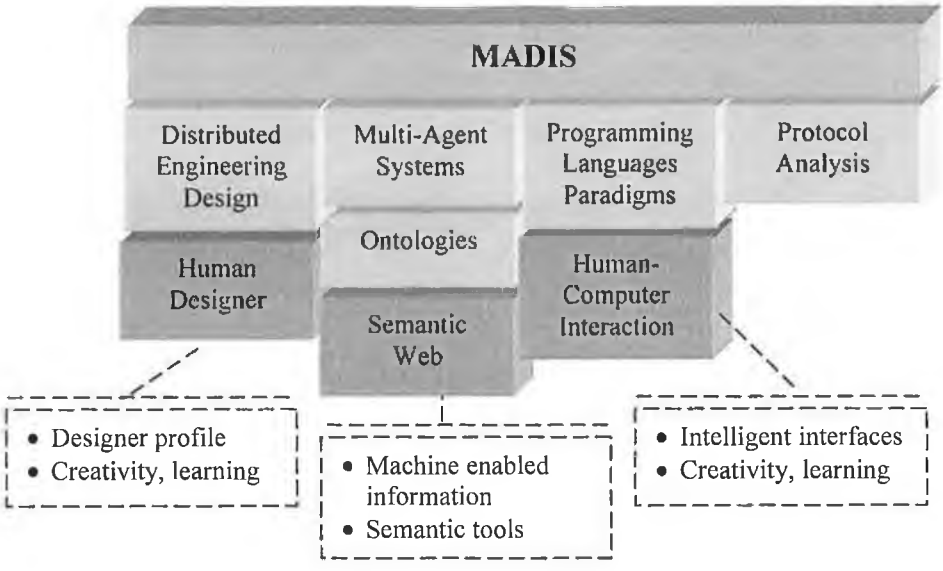


Figure 6.2. Recommendations for future research

Further research into these areas can potentially deliver many benefits and major improvements for the application of AI technologies to distributed engineering design.

More specifically, the suggestions for further development of the proposed MADIS architecture and prototype can be summarised as follows:

- Specification and implementation of an improved version of the Profile Manager agent that can actually 'learn' user preferences over time and cooperate with the User Interface Controller agent to better serve the designer. The learning process should cover not only graphical preferences but also the types of information the user is most interested in.

- Dynamic creation of the graphical user interface for the agents that necessitate one based on the information provided by the designer profile. This is considered a step forward towards 'intelligent' user interfaces.
- Development of new capabilities within the User Agent Society such as autonomous user awareness on the availability of new relevant information in the context of the current designer profile (with the help of the Ontology Agent Society and new agents that can explore the Semantic Web), new components for collaboration support. These new functionalities should not obstruct but support and encourage designer's cognitive activities.
- Discovery and implementation of new services (besides browse and search for knowledge) that can be provided to the user in order to support him/her during the process of design.
- Investigation (or proposal) of new collaborative environments and integration with the MADIS User Agent Society and/or Semantic Web Portal to better support the cooperation process (i.e. communication, coordination, collaboration, co-location).
- Integration of new agents from the Application Agent Society in other software tools (besides CAD systems such as Pro/Engineer) used in the distributed engineering design process. This would enable the access to extra information widening the designer's view of the world.
- Specification and implementation of a new agent society (maybe called SW Information Agent Society) dealing with the information available in the Semantic Web. Having the objectives set by a personal user agent, an SW Information Agent can explore the Semantic Web data to autonomously (or semi-autonomously) provide the designer with the relevant information for the task at hand.
- Refinement of the ontology library.

The author believes that further study and development of MADIS along the suggestions described above can potentially deliver a more intelligent system to efficiently support the distributed designer during an information-intensive problem solving process that requires many knowledgeable decisions. Exploring the fields of distributed artificial intelligence and human-computer interaction, future work should focus on the extension of MADIS to an intelligent system that supports and improves the distributed engineering design process and also has the capability to trigger designer's creativity and encourage new ideas and perspectives.

References

Baecker, R. M. and W. A. S. Buxton (1987). Readings in Human-Computer Interaction: A Multidisciplinary Approach, Morgan-Kaufmann Publishers.

Bal, J. and Y. James-Gordon (2001). "Learning Style Preferences of Engineers in Automotive Design." Journal of Workplace Learning 13(6).

Baya, V. (1996). Information handling behavior of designers during conceptual design: three experiments. Department of Mechanical Engineering, Stanford University.

Benjamins, V. R., J. Contreras, O. Corcho and A. Gomez-Perez (2002). Six Challenges for the Semantic Web. International Semantic Web Conference (ISWC2002), Sardinia, Italia.

Berners-Lee, T. (1998). Semantic Web Road Map. http://www.w3.org/DesignIssues/Semantic.html, World Wide Web Consortium.

Berners-Lee, T., J. Hendler and O. Lassila (2001). "The semantic web." Scientific American 284(5): 34-43.

Brennan, A. (1996). A Graphical User Interface Design Tool to Facilitate Managerial Learning. CIMRU, University College Galway.

Cross, N. (1999). "Natural intelligence in design." Design Studies 20(1): 25-39.

Decker, S., F. v. Harmelen, J. Broekstra, M. Erdmann, D. Fensel, I. Horrocks, M. Klein and S. Melnik (2000). "The Semantic Web - on the respective Roles of XML and RDF." IEEE Internet Computing.

Dumbill, E. (2001). Building the Semantic Web. http://www.xml.com/pub/a/2001/03/07/buildingsw.html., XML.com.

Fensel, D. (2000). Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce. Berlin, Springer.

Gero, J. (2000). "Computational Models of Inovative and Creative Design Process." Technological Forecasting and Social Change 64: 183-196.

Hendler, J., T. Berners-Lee and E. Miller (2002). "Integrating Applications on the Semantic Web." Journal of the Institute of Electrical Engineers of Japan 122(10): 676-680.

Hnug, D. W. L. and C. Der-Thang (2001). "Situated Cognition, Vygotskian Thought and Learning from the Communities of Practice Perspective : Implications for the Design of Web-Based E-Learning." Education Media International.

Karuppan, C. M. (2001). "Web based teaching materials : a user's profile." Research : Electronic Networking Applications and Policy 11(2).

Kolb, D. (1984). Experiential Learning: Experience as the Source of Learning and Development, Prentice-Hall.

Kruger, C. and N. Cross (2001). Modelling Cognitive Strategies in Creative Design. Computational and Cognitive Models of Creative Design V. J. Gero and M. Maher. University of Sydney, Australia.

Lawson, B. (1990). How Designers Think 2nd Ed.

Lawson, B., M. Bassanino, M. Phiri and J. Worthington (2003). "Intentions, practices and aspirations: Understanding learning in design." Design Studies 24(4): 327-339.

Lubart, T. (2005). "How can computers be partners in the creative process: Classification and commentary on the Special Issue." International Journal of Human-Computer Studies.

Nakakoji, K., Y. Yamamoto and M. Ohira (1999). "A Framework that Supports Collective Creativity in Design using Visual Images." Creativity and Cognition: 166-173.

Petroski, H. (1996). Invention by Design: How Engineers Get from Thought to Thing, Harvard University Press.

Ramsdell, J. D. (2000). A Foundation for a Semantic Web.

Roche, T. (1999). Development of a Design for the Environment Workbench. CIMRU, Industrial Engineering Dept. Galway, UCG.

Salvendy, G. (1997). Handbook of Human Factors. New York, John Wiley & Sons.

Shneiderman, B. (1992). Designing the User Interface: Strategies for Effective Human-Computer Interaction, Addison-Wesley Publishing Co.

Shneiderman, B. and H. Hochheiser (2001). "Universal usability as a stimulus to advanced interface design." draft for Behaviour and Information Technology.

Swartz, A. and J. Hendler (2001). The Semantic Web: A Network of Content for the Digital City. Proceedings Second Annual Digital Cities Workshop, Kyoto, Japan.

Zhang, P. and N. Li (2004). "An assessment of human–computer interaction research in management information systems: topics and methods." Computers in Human Behavior 20(2): 125-147.

References

This list of references is accumulated from those recorded at the end of each chapter.

1. Anumba, C. J., Z. Ren, A.Thorpe, O. O. Ugwu and L.Newnham (2003). "Negotiation within a multi-agent system for the collaborative design of light industrial buildings." Advances in Engineering Software 34: 389-401.
2. Anumba, C. J., O. O. Ugwu, L.Newnham and A.Thorpe (2002). "Collaborative design of structures using intelligent agents." Automation in Construction 11: 89-103.
3. Archer, L. (1984). Systematic Method for Designers. Developments in Design Methodology. N. Cross. London, John Wiley & Sons Ltd: pp 57 - 82.
4. Arias, E., H. Eden, G. Fischer, A. Gorman and E. Scharff (2000). " Transcending the Individual Human Mind - Creating Shared Understanding through Collaborative Design." ACM transactions on Computer-Human Interaction Vol. 7, No. 1: 84 - 113.
5. Artala, A., E. Franconi, N. Guarino and L. Pazzi (1996). "Part-Whole Relations in Object-Centered Systems: an Overview." Data and Knowledge Engineering 20(3): 347-383.
6. Baecker, R. M. and W. A. S. Buxton (1987). Readings in Human-Computer Interaction: A Multidisciplinary Approach, Morgan-Kaufmann Publishers.
7. Bal, J. and Y. James-Gordon (2001). "Learning Style Preferences of Engineers in Automotive Design." Journal of Workplace Learning 13(6).
8. Ballmann, S. and D. Wieczorek (1998). Java Intelligent Agent Component Ware (JIAC) - technical documentation. Berlin, DAI Laboratory Technical University of Berlin.
9. Bauer, B. (2001). UML Class Diagrams: Revisited in the Context of Agent-Based Systems. Agent-Oriented Software Engineering, Montreal.
10. Bauer, B., J. P. Müller and J. Odell (2001). Agent UML: A Formalism for Specifying Multiagent Interaction. Agent-Oriented Software Engineering, Springer-Verlag, Berlin.
11. Baya, V. (1996). Information handling behavior of designers during conceptual design: three experiments. Department of Mechanical Engineering, Stanford University.
12. Bellifemine, F., A. Poggi and G. Rimassa (1999). JADE - A FIPA-compliant agent framework. Proceedings of PAAM'99, London.
13. Benbunan-Fich, R. (2001). "Using protocol analysis to evaluate the usability of a commercial web site." Information & Management.
14. Benjamins, V. R., J. Contreras, O. Corcho and A. Gomez-Perez (2002). Six Challenges for the Semantic Web. International Semantic Web Conference (ISWC2002), Sardinia, Italia.
15. Bergamaschi, S., S. Castano, S. D. C. d. Vimercati and M. Vincini (1998). An Intelligent Approach to Information Integration. Formal Ontology in Information System. N. Guarino. Amsterdam, IOS Press.
16. Berners-Lee, T. (1998). Semantic Web Road Map. http://www.w3.org/DesignIssues/Semantic.html, World Wide Web Consortium.
17. Berners-Lee, T., J. Hendler and O. Lassila (2001). "The semantic web." Scientific American 284(5): 34-43.
18. Bertola, P. and J. C. Teixeira (2003). "Design as a knowledge agent. How design as a knowledge process is embedded into organizations to foster innovation." Design Studies 24(2): 181-194.
19. Bigus, J. P., D. A. Schlosnagle, J. R. Pilgrim, W. N. M. III and Y. Diao (2002). "ABLE: A toolkit for building multiagent autonomic systems." IBM Systems Journal 41(3).

20. Blazquez, M., M. Fernandez, J. M. Garcia-Pinar and A. Gomez-Perez (1998). Building Ontologies at the Knowledge Level using the Ontology Design Environment. 11th Knowledge Acquisition Workshop, KAW98, Bamff, Canada.
21. Borst, P., H. Akkermans and J. Top (1997). "Engineering Ontologies." International Journal of Human-Computer Studies 46(Special Issue on Using Explicit Ontologies in KBS Development): 365-406.
22. Bradshow, J. M. (1997). An Introduction to Software Agents. Software Agents. J. M. Bradshow. Cambridge, MIT Press.
23. Branch, J. L. (2000). "Investigating the Information-Seeking Processes of Adolescents: The Value of Using Think Alouds and Think Afters." Library & Information Science Research.
24. Brazier, F. M. T., B. M. Dunin-Keplicz, N. R. Jennings and J. Treur (1997). "DESIRE: Modelling Multi-Agent Systems in a Compositional Formal Framework." International Journal of Cooperative Information Systems 6(Special Issue on Formal Methods in Cooperative Information Systems: Multi-Agent Systems): 67-94.
25. Brazier, F. M. T., P. A. T. v. Eck and J. Treur (2001). "Modelling a Society of Simple Agents: From Conceptual Specification to Experimentation." Journal of Applied Intelligence 14: 161-178.
26. Brazier, F. M. T., L. V. Moshkina and N. J. E. Wijngaards (2001). "Knowledge level model of an individual designer as an agent in collaborative distributed design." Artificial Intelligence in Engineering 15: 137-152.
27. Brennan, A. (1996). A Graphical User Interface Design Tool to Facilitate Managerial Learning. CIMRU, Univeristy College Galway.
28. Brooks, R. A. (1986). "A robust layered control system for a mobile robot." IEEE Journal of Robotics and Automation 2: 14-23.
29. Brown, D. C., B. V. Dunskus, D. L. Grecu and I. Berker (1995). SINE: Support For Single Function Agents. Proceedings of the International Conference on Artificial Intelligence in Engineering, Udine, Italy.
30. Caldecote, V. (1963). "The Design Team in Relation to The Individual Designer." The Practice of and Education for Engineering Design 178(Part B): 16-19.
31. Calisti, M. (2003). FIPA standards for promoting interoperability of industrial agent systems. Agencities Info Days, Helsinki.
32. Campbell, M., J. Cagan and K. Kotovsky (1999). "A-Design: An Agent-Based Approach to Conceptual Design in a Dynamic Environment." Research in Engineering Design 11(3): 172-192.
33. Carver, N., V. Lesser and Q. Long (1993). Distributed sensor Interpretation: Modelling Agent Interpretations in DRESUN, UMass Technical Report, UMCS 93-75.
34. Case, M. P. and S. C.-Y. Lu (1996). "Discourse Model for collaborative design." Computer-Aided Design 28(5): 333-345.
35. Chaib-draa, B. (1996). "Interaction Between Agents in Routine, Familiar and Unfamiliar Situations." International Journal of Intelligent & Cooperative Information Systems 5(1): 1-25.
36. Chaib-draa, B. and F. Dignum (2002). "Trends in Agent Communication Language." Computational Intelligence 18(2).
37. Chan, C. S. (2000). "An examination of the forces that generate a style." Design Studies.
38. Chao, K.-M., P. Norman, R. Anane and A. James (2002). "An agent-based approach to engineering design." Computers in Industry 48: 17-27.
39. Chen, J. J.-Y. and S.-W. Su (2003). "AgentGateway: A communication tool for multi-agent systems." Information Sciences 150: 153-164.

40. Cheyer, A. and D. Martin (2001). "The Open Agent Architecture." Journal of Autonomous Agents and Multi-Agent Systems 4(1): 143-148.
41. Chira, C. (2002). Design, Development and Testing of a CAD Integrated Design for Environment Software Tool. Galway, Galway Mayo Institute for Technology.
42. Chira, O., C. Chira, D. Tormey, A. Brennan and T. Roche (2003). An agent-based approach to knowledge management in distributed design. 10th ISPE International Conference on Concurrent Engineering: Research and Applications, Madeira Island, Portugal.
43. Chira, V. O. (2004). Towards a Machine Enabled Semantic Framework for Distributed Engineering Design. Department of Mechanical & Industrial Engineering. Galway, Galway-Mayo Institute of Technology.
44. Chu, E., K. Srihari and C. R. Emerson (1996). "Distributed Artificial Intelligence in Process Control." 19th International Conference on Computers and Industrial Engineering.
45. Cohen, P. R. and H. J. Levesque (1995). Communicative actions for artificial agents. Proceedings of the International Conference on Multi-Agent Systems, San Francisco, AAAI Press.
46. Court, A. W., S. J. Culley and C. A.McMahon (1997). "The influence of information technology in new product development: Observations of an empirical study of the access of engineering design information." International Journal of Information Management 17(5): 359-375.
47. Court, A. W., S. J. Culley and C. A. McMahon (1993). The Information Requirements of Engineering Designers. International Conference on Engineering Design, The Hague.
48. Coyne, R. D., M. A. Rosenman, M. A. Radford, M. Balachandran and J. S. Gero (1990). Knowledge based Design Systems, Addison Wesley.
49. Crabtree, R. A., M. S. Fox and N. K. Baid (1997). "Towards an Understanding of Collaborative Design Activities." Research in Design Engineering 9: 70-84.
50. Cross, N. (1994). Engineering Design Methods, J. Wiley & Sons.
51. Cross, N. (1999). "Natural intelligence in design." Design Studies 20(1): 25-39.
52. Cross, N. and A. C. Cross (1995). "Observations of teamwork and social process in design." Design Studies 16(2): 143-170.
53. Cutkosky, M. R., R. S. Englemore, R. E. Fikes, M. R. Genesereth, T. R. Gruber, W. S. Mark, J. M. Tenenbaum and J. C. Weber (1997). PACT: An Experiment in Integrating Concurrent Engineering Systems. Readings in Agents. M. N. Huhns and M. P. Singh. San Francisco, CA, USA, Morgan Kaufmann: 46-55.
54. Dale, J. and E. Mamdani (2001). "Open Standards for Interoperating Agent-Based Systems." Software Focus, Wiley.
55. Decker, S., F. v. Harmelen, J. Broekstra, M. Erdmann, D. Fensel, I. Horrocks, M. Klein and S. Melnik (2000). "The Semantic Web - on the respective Roles of XML and RDF." IEEE Internet Computing.
56. DeLoach, S. A. (1999). Multiagent Systems Engineering: A Methodology And Language for Designing Agent Systems. Agent-Oriented Information Systems (AOIS) '99.
57. DeLoach, S. A., M. F. Wood and C. H. Sparkman (2001). "Multiagent Systems Engineering." The International Journal of Software Engineering and Knowledge Engineering 11(3).
58. Devedzic, V. (2001). "Knowledge Modeling - State of the Art." Integrated Computer-Aided Engineering 8(3): 257-281.
59. Dickinson, I. (1997). "Agents Standards." Hewlett-Packard Company.

60. Dumbill, E. (2001). Building the Semantic Web. http://www.xml.com/pub/a/2001/03/07/buildingsw.html., XML.com.
61. Durfee, E. H. (2001). "Scaling Up Agent Coordination Strategies." IEEE Computer 34(7): 39-46.
62. Durfee, E. H. and V. R. Lesser (1991). "Partial Global Planning: A Coordination Framework for Distributed Hypothesis Formation." IEEE Transactions on Systems, Man, and Cybernetics, Special Issue on Distributed Sensor Networks SMC-21(5): 1167-1183.
63. Eder, W. E. (1998). "Design Modelling - A Design Science Approach (And Why Does Industry Not Use It?)." Journal of Engineering Design 9(4).
64. Edmonds, E. A., L. Candy, R. Jones and B. Soufi (1994). "Support for Collaborative Design : Agents and Emergence." Communications of the ACM 37(7).
65. Ericsson, K. A. and H. A. Simon (1999). Protocol Analysis: Verbal Reports as Data, The MIT Press.
66. Evbuomwan, N., S. Sivaloganathan and A. Jebb (1996). "A survey of design philosophies, models, methods and systems." Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture 210: 301-319.
67. Fatima, S. S., M. Wooldridge and N. R. Jennings (2004). "An agenda-based framework for multi-issue negotiation." Artficial Intelligence 152: 1-45.
68. Feilden, G. B. R. (1963). Engineering Design. London, Report of Royal Commission - HMSO.
69. Fensel, D. (2000). Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce. Berlin, Springer.
70. Ferguson, I. A. (1992). "TouringMachines: Autonomous Agents with Attitudes." IEEE Computer 25(5).
71. Fernandez, M., A. Gomez-Perez and N. Juristo (1997). METHONTOLOGY: From Ontological Art Towards Ontological Engineering Workshop on Ontological Engineering. Symposium on ONtological Engineering of AAAI, Standford, California.
72. Fernandez-Lopez, M. (2001). "Overview Of Methodologies for Building Ontologies." Intelligent Systems 16(1): 26-34.
73. Fernandez-Lopez, M., A. Gomez-Perez, A. Pazos-Sierra and J. Pazos-Sierra (1999). "Building a Chemical Ontology Using Methontology and the Ontology Design Environment." IEEE Intelligent Systems and their applications January/February: 37-46.
74. Fikes, R., Farquhar, A. (1999). "Distributed Repositories of Highly Expressive Reusable Ontologies." IEEE Intelligent Systems 14(2): 73-79.
75. Finin, T., R. Fritzson, D. McKay and R. McEntire (1994). KQML as an Agent Communication Language. Proceedings of the Third International Conference on Information and Knowledge Management.
76. Finin, T., Y. Labrou and J. Mayfield (1997). KQML as an agent communication language. Software Agents. B. M. Jeffrey, MIT Press.
77. Finkelstein, L. and A. C. W. Finkelstein (1983). Review of Design Methodology. IEE Proceedings.
78. FIPA (2004). FIPA Agent Management Specification.
79. Fischer, G. (2002). "Knowledge Management : Problems, Promises, Realities and Challenges." IEEE Intelligent Systems.
80. Fisher, M. (1994). A Survey of Concurrent METATEM - The Language and its Applications. Proceedings of First International Conference on Temporal Logic (ICTL), Bonn, Germany, Springer-Verlag.

81. Foner, L. N. (1993). What's An Agent, Anyway? A Sociological Case Study, Media Laboratory, Massachusetts Institute of Technology.
82. Franklin, S. and A. Graesser (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, Springer-Verlag, 1996, Berlin, Germany.
83. Fruchter, R., K. A. Reiner, G. Toye and L. J. Leifer (1996). "Collaborative Mechatronic System Design." Concurrent Engineering: Research and Applications 4(4): 401-413.
84. Gaines, B. (1997). "Editorial: Using Explicite Ontologies in Knowledge-based System Development." International Journal of Human-Computer Systems 46: 181.
85. Gammack, J. and S. Poon (1999). Communication Media for Supporting Distributed Engineering Design. 32nd Hawaii International Conference on System Sciences, Hawaii.
86. Gasparski, W. and A. Strzalecki (1990). "Contributions to design science: Praxeological perspective, Design Methods and Theories." Journal of DMG 24(2): 1186-1194.
87. Gasser, L. (1998). Social conceptions of knowledge and action: DAI foundations and open systems dynamics. Readings in Agents. M. N. Huhns and M. P. Singh, Morgan Kaufmann Publishers.
88. Genesereth, M. R. and S. P. Ketchpel (1994). "Software Agents." Communications of the ACM, ACM Press.
89. Georgeff, M., B. Pell, M. Pollack, M. Tambe and M. Wooldridge (1999). The Belief-Desire-Intention Model of Agency. Intelligent Agents. J. P. Muller, M. Singh and A. Rao, Springer-Verlag. 1365.
90. Gero, J. (2000). "Computational Models of Inovative and Creative Design Process." Technological Forecasting and Social Change 64: 183-196.
91. Gero, J. S. and T. McNeil (1997). "An approach to the analysis of design protocols." Design Studies.
92. Gero, J. S. and H. H. Tang (2001). "The differences between retrospective and concurrent protocols in revealing the process-oriented aspects of the design process." Key Centre of Design Computing and Cognition.
93. Giacomo, G. D., Y. Lespérance and H. J. Levesque (2000). "ConGolog, a concurrent programming language based on the situation calculus." Artficial Intelligence 121: 109-169.
94. Goldschmidt, G. (1995). "The designer as a team of one." Design Studies.
95. Gomez-Perez, A. (1998). Knowledge Sharing and Reuse. The Handbook on Expert Systems. Liebowitz, CRC Press.
96. Gomez-Perez, A. (1999). "Ontological Engineering: A State Of The Art." Expert Update. Ontono 2(3): 38-43.
97. Greaves, M., V. Stavridou-Coleman and R. Laddaga (2004). "Dependable Agent Systems." IEEE Intelligent Systems.
98. Green, S., L. Hurst, B. Nangle, P. Cunningham, F. Somers and R. Evans (1997). Software Agents: A review. Dublin, Intelligent Agents Group, Trinity College Dublin, Broadcom Eireann Research Ltd.
99. Greenberg, S. (1996). "Teaching Human Computer Interaction to Programmers." ACM Interactions 3(4): 62-76.
100. Gregory, S. (1966). The Design Method. London, Butterworth & Co Ltd.
101. Gruber, T. R. (1991). The Role of Common Ontology in Achieving Shareable, Reusable Knowledge Bases. Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference, San Mateo, Morgan Kaufmann, 1991.
102. Gruber, T. R. (1993). "A Translation Approach to Portable Ontology Specification." Knowledge Acquisition 5(2): 199-220.

103. Gruber, T. R. (1995). "Toward Principles for the Design of Ontologies Used for Knowledge Sharing." International Journal of Human and Computer Studies 43(5/6): 907-928.
104. Gruninger, M. and M. S. Fox (1995). Methodology for the Design and Evaluation of Ontologies. IJCAI Workshop on Basic Ontological Issues in Knowledge Sharing, Montreal, Quebec, Canada.
105. Guarino, N. (1997). Semantic Matching:Formal Ontological Distinctions for Information Organization, Extraction, and Integration. Summer School on Information Extraction, Frascati, Italy, July 14-19.
106. Guarino, N. (1998). Formal Ontology and Information Systems. Formal Ontology in Information Systems. FOIS'98, 6-8 June 1998., Trento, IOS Press,.
107. Guarino, N., M. Carrara and P. Giaretta (1994). Formalizing Ontological Commitments. National Conference on Artificial Intelligence, AAAI 94, Seatle, Morgan Kaufmann.
108. Hales, C. (1987). Analysis of the Engineering Design Process in an Industrial Context. Department of Engineering. Cambridge, University of Cambridge.
109. Hartley, P. (1993). Interpersonal Communication, Routledge.
110. Harvey, C. M. and R. J. Koubek (1998). "Toward a Model of Distributed Engineering Collaboration." Computers & Industrial Engineering 35(1-2): 173-176.
111. Hayes-Roth, B. (1995). Agents on Stage: Advancing the State of the Art of AI. Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95).
112. Helin, H. (2003). Agent Architectures & Languages, http://www.cs.helsinki.fi/u/hhelin/opetus/oat/. 2003.
113. Henderson, R., J. Podd, M. Smith and H. Varela-Alvarez (1995). "An examination of four user-based software evaluation methods." Interacting with computers.
114. Hendler, J., T. Berners-Lee and E. Miller (2002). "Integrating Applications on the Semantic Web." Journal of the Institute of Electrical Engineers of Japan 122(10): 676-680.
115. Hirsch, B. (2000). Extended Products in Dynamic Enterprises", E-Business: Key Issues, Applications and Technologies,: 622-628.
116. Hnug, D. W. L. and C. Der-Thang (2001). "Situated Cognition, Vygotskian Thought and Learning from the Communities of Practice Perspective : Implications for the Design of Web-Based E-Learning." Education Media International.
117. Ho, J. and R. Tang (2001). "Towards an Optical Resolution to Information Overload : An Infomediary Approach." ACM.
118. Howden, N., R. Ronnquist, A. Hodgson and A. Lucas (2001). JACK Intelligent Agents - Summary of an Agent Infrastructure. 5th International Conference on Autonomous Agents.
119. http://jade.cselt.it, JADE, Last Accessed August 2005.
120. http://jakarta.apache.org/tomcat/ Tomcat, Last Accessed August 2005.
121. http://java.sun.com, Java, Last Accessed August 2005.
122. http://java.sun.com/products/servlet/ Java Servlets, Last Accessed August 2005.
123. http://protege.stanford.edu, Protege 2000, Last Accessed May 2005.
124. http://www.agentbuilder.com/, Last Accessed August 2005.
125. http://www.agent-software.com, Last Accessed August 2005.
126. http://www.ai.mit.edu/people/sodabot/sodabot.html, The SodaBot System, Last Accessed October 2004.
127. http://www.apache.org/, Apache, Last Accessed August 2005.
128. http://www.cognitiveagent.com, iGEN Overview, Last Accessed August 2005.
129. http://www.directia.com/, Last Accessed August 2005.

130. http://www.fipa.org, Foundation for Intelligent Physical Agents, Last Accessed August 2005.
131. http://www.hpl.hp.com/semweb/jena, JENA, Last Accessed August 2005.
132. http://www.iks.com/agentx.htm, Last Accessed September 2004.
133. http://www.lotus.com, Lotus Sametime, Last Accessed August 2005.
134. http://www.omg.org, Object Management Group, Last Accessed August 2005.
135. http://www.ptc.com, ProE, Last Accessed August 2005.
136. http://www.semanticweb.org, Semantic Web, Last Accessed August 2005.
137. http://www.trl.ibm.com/aglets, Aglets, Last Accessed August 2005.
138. http://www.tryllian.com, The Agent Development Kit (ADK), Last Accessed August 2005.
139. http://www.w3.org, RDF, Last Accessed August 2005.
140. Huang, J. (1999). "Knowledge sharing and innovation in distributed design: implications of internet-based media on design collaboration." International Journal of Design Computing: Special Issue on Design Computing on the Net (DCNet'99).
141. Hubka, V. and E. Eder (1987). "A Scientific Approach to Engineering Design." Design Studies 8(3): 123-137.
142. Hubka, V. and E. Eder (1996). Design Science, Springer-Verlag.
143. Huget, M.-P. (2002). Desiderata for Agent Oriented Programming Languages, University of Liverpool.
144. Huget, M.-P. (2002). Extending Agent UML Protocol Diagrams, University of Liverpool Department of Computer Science.
145. IEEE96 (1996). IEEE Standard for Developing Software Life Cycle Processes. New York (USA), IEEE Computer Society.
146. Iglesias, C. A., M. Garijo and J. C. Gonzalez (1999). A Survey of Agent-Oriented Methodologies. Proceedings of the 5th International Workshop on Intelligent Agents V : Agent Theories, Architectures, and Languages.
147. Iheagwara, C. and A. Blyth (2002). "Evaluation of the performance of ID systems in a switched and distributed environment the RealSecure case study." Computer Networks.
148. IKV++GmbH (2001). Grasshopper Basics And Concepts. http://www.grasshopper.de/.
149. Ingrand, F. F., M. P. Georgeff and A. S. Rao (1992). "An Architecture for Real-Time Reasoning and System Control." IEEE Expert 7(6): 33-44.
150. Jagdev, H. and J. Browne (1998). "The Extended Enterprise-A context for Manufacturing." Production Planning and Control 9(3): 326-339.
151. Jennings, N. R. (2000). "On agend-based software engineering." Artificial Intelligence.
152. Jennings, N. R., K. P. Sycara and M. Wooldridge (1998). "A Roadmap of Agent Research and Development." Journal of Autonomous Agents and Multi-Agent Systems 1(1): 7-36.
153. Jennings, N. R. and M. Wooldridge (1998). Applications of Agent Technology. Agent Technology: Foundations, Applications, and Markets. N. R. Jennings and M. Wooldridge, Springer-Verlag.
154. Jeon, H., C. Petrie and M. R. Cutkosky (2000). "JATLite: A Java Agent Infrastructure with Message Routing." IEEE Internet Computing.
155. Jiang, Y. C., Z. Y. Xia, Y. P. Zhong and S. Y. Zhang (2005). "Autonomous trust construction in multi-agent systems—a graph theory methodology." Advances in Engineering Software 36: 59-66.
156. Karuppan, C. M. (2001). "Web based teaching materials : a user's profile." Research : Electronic Networking Applications and Policy 11(2).

157. Kimura, F. (1997). Inverse manufacturing: From Products to Services. Managing Enterprises - Stakeholders, Engineering, Logistics and Achievement First International Conference Proceedings, MEP Ltd, London,.
158. Kiniry, J. and D. Zimmerman (1997). "A Look at Mitsubishi's Concordia." IEEE Internet Computing online.
159. Kinny, D., M. Georgeff and A. Rao (1996). A Methodology and Modelling Technique for Systems of BDI Agents. Agents Breaking Away, 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, Springer.
160. Kolb, D. (1984). Experiential Learning: Experience as the Source of Learning and Development, Prentice-Hall.
161. Kruger, C. and N. Cross (2001). Modelling Cognitive Strategies in Creative Design. Computational and Cognitive Models of Creative Design V. J. Gero and M. Maher. University of Sydney, Australia.
162. Kumar, S., M. J. Huber, D. R. McGee, P. R. Cohen and H. J. Levesque (2000). Semantics of Agent Communication Languages for Group Interaction. The Seventeenth National Conference on Artificial Intelligence (AAAI 2000), Austin, Texas, AAIT Press/The MIT Press.
163. Labrou, Y., T. Finin and Y. Peng (1999). "Agent Communication Languages: The Current Landscape." IEEE Intelligent Systems.
164. Lang, S. Y. T., J. Dickinson and R. O. Buchal (2002). "Cognitive factors in distributed design." Computers in Industry 48: 89-98.
165. Lassila, O. and R. R. Swick (1999). Resource Description Framework (RDF) Model and Syntax Specification, W3C Recommendation 22 February 1999. 2003.
166. Laure, E. (2001). "OpusJava: A Java framework for distributed high performance computing." Future Generation Computer Systems 18: 235-251.
167. Lawson, B. (1990). How Designers Think 2nd Ed.
168. Lawson, B., M. Bassanino, M. Phiri and J. Worthington (2003). "Intentions, practices and aspirations: Understanding learning in design." Design Studies 24(4): 327-339.
169. Lazansky, J., O. Stepankova, V. Marik and M. Pechoucek (2001). "Application of the multi-agent approach in production planning and modelling." Engineering Applications of Artificial Intelligence 14(3): 369-376.
170. Lees, B., C. Branki and I. Aird (2001). "A framework for distributed agent-based engineering design support." Automation in Construction 10: 631-637.
171. Lesser, V. and D. Corkill (1981). "Functionally Accurate, Cooperative Distributed Systems." IEEE Transactions on Systems, Man, and Cybernetics SMC-11(1): 81-96.
172. Lesser, V. R. (1995). "Multiagent Systems: An Emerging Subdiscipline of AI." ACM Computing Surveys 27(3).
173. Lesser, V. R. (1999). "Cooperative Multiagent Systems: A Personal View of the State of the Art." IEEE Transactions on Knowledge and Data Engineering 11(1).
174. Liang, W.-Y. and C.-C. Huang (2002). "The agent-based collaboration information system of product development." International Journal of Information Management 22: 211-224.
175. Liu, H., M. Tang and J. H. Frazer (2002). "Supporting evolution in a multi-agent cooperative design environment." Advances in Engineering Software 33: 319-328.
176. Love, T. (2002). "Constructing a coherent cross-disciplinary body of theory about designing and designs: some philosophical issues." Design Studies 23(3): 345-361.
177. Lubart, T. (2005). "How can computers be partners in the creative process: Classification and commentary on the Special Issue." International Journal of Human-Computer Studies.

178. Luck, M., P. McBurney and C. Preist (2003). "Agent Technology: Enabling Next Generation Computing." AgentLink(ISBN 0854 327886).
179. Luckman, J. (1984). An Approach to the Management of Design. Developments in Design Methodolgy. N. Cross. London, John Wiley & Sons Ltd: 83-97.
180. MacGregor, S. P. (2002). "New Perspectives for Distributed Design Support." The Journal of Design Research 2(2).
181. MacGregor, S. P., A. L. Thomson and N. P. Juster (2001). Information sharing within a distributed, collaborative design process: a case study. Proceedings of Design Engineering Technical Conferences and Computers (DETC'01) and Information in Engineering Conference, Pittsburgh, Pennsylvania.
182. Maes, P. (1995). "Artificial Life meets Entertainment: Lifelike Autonomous Agents." Communications of the ACM, ACM Press 38(11): 108-114.
183. Man, E., J. E. Diez-Campo, C. Chira and T. Roche (2002). Product Life Cycle Design using the DFE Workbench. 5th IFIP International Conference on Information Technology for Balanced Automation Systems in Manufacturing and Services (BASYS), Cancun, Mexico.
184. Martin, F. J., E. Plaza, J. A. Rodriguez-Aguilar and J. Sabater (1998). Java Interagents for Multi-Agent Systems. Software Tools for Developing Agents.
185. McGuire, J. G., D. R. Kuokka, J. C. Weber, J. M. Tenenbaum, T. R. Gruber and G. R. Olsen (1993). "SHADE: Technology for knowledge-based collaborative engineering." Concurrent Engineering: Research and Applications 1(3).
186. Mena, E., Kashyap, V., Illarramendi, A., Sheth, A. (1998). Domain Specific Ontologies for Semantic Information Brokering on the Global Information Infrastructure. Formal Ontology in Information Systems. N. Guarino. Amsterdam, IOS Press.
187. Mori, T. and M. R. Cutkosky (1998). Agent-based collaborative design of parts in assembly. Proceedings of Design Engineering Technical Conference '98, Atlanta, Georgia, USA.
188. Muller, J. P. and M. Pischel (1993). The Agent Architecture InteRRaP: Concept and Application, DFKI Saarbrucken.
189. Nakakoji, K., Y. Yamamoto and M. Ohira (1999). "A Framework that Supports Collective Creativity in Design using Visual Images." Creativity and Cognition: 166-173.
190. Nakakoji, K., Y. Yamamoto, T. Suzuki, S. Takada and M. Gross (1998). "From Critiquing to Representational Talkback: Computer Support for Revealing Features in Design." Knowledge-Based Systems Journal 11(7-8): 457-468.
191. Ndumu, D. and H. Nwana (1996). "Research and Development Challenges for Agent-Based Systems." IEE/BCS Software Engineering Journal.
192. Neches, R. (1994). The Knowledge Sharing Effort, http://www-ksl.stanford.edu/knowledge-sharing/papers/kse-overview.html.
193. Neches, R., R. Fikes, T. Finin, T. Gruber, R. Patil, T. Senator and W. R. Swartout (1991). Enabling Technology For Knowledge Sharing. AI Magazine. 12: 36-56.
194. Nonaka, I. and N. Konno (1998). "The Concept of "Ba": Building a Foundation for Knowledge Creation." California Management Review 40(3): 40-54.
195. Nonaka, I. and H. Takeuchi (1995). The Knowledge Creating Company: How Japanese Companies Create the Dynasties of Innovation. New York, Oxford University Press.
196. Noy, N. F. and D. L. McGuinness (2001). Ontology Development 101: A Guide to Creating Your First Ontology. Stanford, CA, 94305, Stanford University.
197. Nwana, H., L. Lee and N. Jennings (1996). "Coordination in Software Agent Systems." BT Technology Journal 14(4): 79-88.

198. Nwana, H. and M. Wooldridge (1996). "Software Agent Technologies." BT Technology Journal 14(4): 68-78.
199. Nwana, H. S. (1996). "Software Agents: An Overview." Knowledge Engineering Review 11(3): 1-40.
200. Nwana, H. S. and D. T. Ndumu (1999). A Perspective on Software Agents Research. Ipswich, British Telecommunications Laboratories.
201. Nwana, H. S., D. T. Ndumu, L. C. Lee and J. C. Collis (1999). "ZEUS: A Tool-Kit for Building Distributed Multi-Agent Systems." Applied Artifical Intelligence Journal 13(1): 129-186.
202. Odell, J. (2000). Agent Technology - Green Paper, OMG - Agent Platform Special Interest Group.
203. Odell, J., M. Nodine and R. Levy (2005). A Metamodel for Agents, Roles, and Groups. Lecture Notes on Computer Science. J. Odell, P. Giorgini and J. Müller. Berlin, Springer. Agent-Oriented Software Engineering (AOSE) V.
204. Odell, J., H. V. D. Parunak and B. Bauer (2000). Extending UML for Agents. Proceedings of the Agent-Oriented Information Systems Workshop at the 17th National conference on Artificial Intelligence.
205. Oliveira, E., K. Fischer and O. Stepankova (1999). "Multi-agent systems: which research for which applications." Robotics and Autonomous Systems 27: 91-106.
206. OMG (2000). Mobile Agent Facility Formal Specification.
207. Pahl, G. and W. Beitz (1996). Engineering a Systematic Approach, Springer.
208. Pahng, F., N. Senin and D. Wallace (1997). Modelling and Evaluation of Product Design Problems in a Distributed Design Environment. DETC'97: 1997 ASME Design Engineering Technical Conferences, Sacramento, California.
209. Park, S. and V. Sugumaran (2005). "Designing multi-agent systems: a framework and application." Expert Systems with Applications 28: 259-271.
210. Patel, U., M. J. D'Cruz and C. Holtham (1997). "Collaborative Design for Virtual Team Collaboration : A Case Study of Jostling on the Web." ACM.
211. Pena-Mora, F., K. Hussein, S. Vadhavkar and K. Benjamin (2000). "CAIRO: a Concurrent Engineering Meeting Environment for Virtual Design Teams." Artificial Intelligence in Engineering 14: 202-219.
212. Petroski, H. (1996). Invention by Design: How Engineers Get from Thought to Thing, Harvard University Press.
213. Polanyi, M. (1966). The Tacit Dimension, Doubleday & Co.
214. Poslad, S., P. Buckle and R. Hadingham (2000). The FIPA-OS Agent Platform: Open Source for Open Standards. Proceedings of the 5th International Conference and Exhibition on the Practical Application of Intelligent Agents and Multi-Agents, UK.
215. Pugh, S. (1991). Total Design: Integrated Methods for Successful Product Engineering, Addison-Wesley Publishing UK.
216. Ramchurn, S. D., D. Huynh and N. R. Jennings (2004). "Trust in multi-agent systems." The Knowledge Engineering Review 19(1).
217. Ramsdell, J. D. (2000). A Foundation for a Semantic Web.
218. Rao, A. S. (1996). AgentSpeak(L): BDI Agents speak out in a logical computable language. Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World.
219. Rao, A. S. and M. P. Georgeff (1995). BDI Agents: From Theory to Practice. Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95), San Francisco, USA.

220. Roche, C. (2000). "Corporate ontologies and concurrent engineering." Journal of Materials Processing Technology 107: 187-193.
221. Roche, T. (1999). Development of a Design for the Environment Workbench. CIMRU, Industrial Engineering Dept. Galway, UCG.
222. Russell, S. and P. Norvig (2003). Artificial Intelligence: A Modern Approach, 2/E, Prentice Hall.
223. Salvendy, G. (1997). Handbook of Human Factors. New York, John Wiley & Sons.
224. Sclater, N., H. Grierson, W. J. Ion and S. MacGregor (2001). "Online Collaborative Design Projects: Overcoming Barriers to Communication." International Journal of Engineering Education 17(2): 189-196.
225. Sen, S. (1997). "Multiagent systems: milestones and new horizons." Trends in Cognitive Sciences 1(9).
226. Shaw, N. C. (2003). Knowledge Management Basics, ICASIT - International Center for Applied Studies in Information Technology. 2003.
227. Shen, W. and J.-P. A. Barthes (1996). "An experimental multi-agent environment for engineering design." International Journal of Cooperative Information Systems 5(2-3): 131-151.
228. Shintani, T., T. Ito and K. Sycara (2000). Multiple negotiations among agents for a distributed meeting scheduler. Proceedings of the Fourth International Conference on MultiAgent Systems.
229. Shneiderman, B. (1992). Designing the User Interface: Strategies for Effective Human-Computer Interaction, Addison-Wesley Publishing Co.
230. Shneiderman, B. and H. Hochheiser (2001). "Universal usability as a stimulus to advanced interface design." draft for Behaviour and Information Technology.
231. Shoham, Y. (1998). Agent-oriented programming. Readings in Agents, Elsevier Science. Artificial Intelligence 60 (1993).
232. Siemieniuch, C. E. and M. Sinclair (1999). "Real-time collaboration in design engineering: an expensive fantasy or affordable reality?" Behaviour & Information Technology 18(5): 361-371.
233. Simon, H. A. (1996). The Sciences of the Artificial. Cambridge Mass., MIT Press.
234. Snow, C. P. (1993). The Two Cultures. Cambridge, Cambridge University Press.
235. Sowa, J. F. (2000). Knowledge Representation: Logical, Philosophical, and Computational Foundations. Pacific Grove, CA, Brooks Cole Publishing Co.
236. Spyns, P., R. Meersman and M. Jarrar (2002). Data Modelling versus Ontology Engineering, ACM SIGMOD Record. 31.
237. Srinivas, H. (2003). Knowledge Management, THE GLOBAL DEVELOPMENT RESEARCH CENTER. 2003.
238. Studer, R., V. R. Benjamins and D. Fensel (1998). "Knowledge Engineering: Principles and Methods." Data and Knowledge Engineering 25(1-2): 161-197.
239. Swartz, A. and J. Hendler (2001). The Semantic Web: A Network of Content for the Digital City. Proceedings Second Annual Digital Cities Workshop, Kyoto, Japan.
240. Sycara, K. P. (1998). "Multiagent Systems." American Association for Artificial Intelligence.
241. Thoben, K.-D. (2002). Extended Products: Evolving Traditional Product Concepts. 7th International Conference on Concurrent Enterprising.
242. Thoben, K.-D., F. Weber and M. Wunram (2002). "Barriers in Knowledge Management and Pragmatic Approaches." Studies in Informatics and Control 11(1).
243. Tomiyama, T. (1994). The Technical Concept of Intelligent Manufacturing Systems (IMS). Tokyo, University of Tokyo.

244. Toye, G., M. R. Cutkosky, L. J. Leifer, J. M. Tenenbaum and J. Glicksman (1993). SHARE: A Methodology and Environment for Collaborative Product Development. Post-Proceedings of the IEEE Infrastructure for Collaborative Enterprises.
245. Tsvetovatyy, M., M. Gini, B. Mobasher and Z. Wieckowski (1997). "MAGMA: An agent-based virtual market for electronic commerce." Journal of Applied Artificial Intelligence.
246. Tuomi, I. (1999). Data Is More Than Knowledge: Implications of the Reversed Knowledge Hierarchy for Knowledge Management and Organizational Memory. The 32nd Hawaii International Conference on System Sciences, Maui, Hawaii.
247. Ullman, D. G. (1996). Mechanical Design Process, McGraw-Hill.
248. Uschold, M. (1998). "Knowledge level modelling : concepts and terminology." The Knowledge Engineering Review 13(1): 5-29.
249. Uschold, M. and M. Gruninger (1996). "Ontologies:Principles, Methods and Applications." The Knowledge Engineering Review 11(2): 93-136.
250. Uschold, M. and M. King (1995). Towards a Methodology for Building Ontologies. Workshop on Basic Ontological Issues in Knowledge Sharing" IJCAI-95.
251. Van de Riet, R., Burg, H., Dehne, F. (1998). Linguistic Issues in Information System Design. Formal Ontology in Information System. G. Nicola. Amsterdam, IOS Press.
252. VanCuilenburg, J. J., O. Scholten and G. Noomen (1991). Stiinta Comunicarii.
253. Viano, G. (2000). Adaptive User Interface for Process Control based on Multi-Agent approach. AVI 2000, Palermo, Italy.
254. Wang, L., W. Shen, H. Xie, J. Neelamkavil and A. Pardasani (2002). "Collaborative conceptual design - state of the art and future trends." Computer Aided Design 34: 981-996.
255. Wang, X. and Y. Lespérance (2001). Agent-Oriented Requirements Engineering Using ConGolog and i*. Proceedings of the 3rd International Bi-Conference Workshop AOIS-2001, Berlin, iCue Publishing.
256. Wang, Y. D., W. Shen and H. Ghenniwa (2003). "WebBlow: a Web/agent-based multidisciplinary design optimization environment." Computers in Industry 52: 17-28.
257. Weber, R. (1997). Ontological Foundations of Information Systems. Melbourne, Coopers and Lybrand.
258. Weiss, G. (1999). Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. London, MIT Press.
259. Werkman, K. J. (1990). Multiagent Cooperative Problem-Solving through Negotiation and Sharing of Perspectives. DAI-List, http://www-2.cs.cmu.edu/afs/cs.cmu.edu/project/ai-repository/ai/pubs/lists/dai-list/dailist/006.10may90.
260. Wong, H. C. and K. Sycara (1999). Adding Security and Trust to Multi-Agent Systems. Autonomous Agents '99 Workshop on Deception, Fraud, and Trust in Agent Societies.
261. Wooldridge, M. (1998). "Agent-based computing." Interoperable Communication Networks 1(1): 71-97.
262. Wooldridge, M. (1999). Intelligent Agents, The MIT Press.
263. Wooldridge, M. and P. Ciancarini (2001). Agent-Oriented Software Engineering: The State of the Art. Agent-Oriented Software Engineering. P. Ciancarini and M. Wooldridge, Springer-Verlag. AI Volume 1957.
264. Wooldridge, M. and N. R. Jennings (1995). "Intelligent Agents: Theory and Practice." Knowledge Engineering Review 10(2).

265. Wooldridge, M., N. R. Jennings and D. Kinny (2000). "The Gaia Methodology for Agent-Oriented Analysis and Design." Autonomous Agents and Multi-Agent Systems Kluwer Academic Publishers(3): 285-312.
266. Wooldridge, M. J. and N. R. Jennings (1995). "Agent Theories, Architectures, and Languages: A Survey." Lecture Notes in Artificial Intelligence, Springer-Verlag 890.
267. Zambonelli, F., N. R. Jennings and M. Wooldridge (2003). "Developing multiagent systems: the Gaia Methodology." ACM Transactions on Software Engineering and Methodology 12(3): 317-370.
268. Zhang, P. and N. Li (2004). "An assessment of human–computer interaction research in management information systems: topics and methods." Computers in Human Behavior 20(2): 125-147.
269. Zhao, G., J. Deng and W. Shen (2001). "CLOVER: an agent-based approach to systems interoperability in cooperative design systems." Computers in Industry 45: 261-276.
270. Zlotkin, G. and J. S. Rosenschein (1989). Negotiation and Task Sharing Among Autonomous Agents in Cooperative Domains. The Eleventh International Joint Conference on Artificial Intelligence, Detroit, Michigan.
271. Zlotkin, G. and J. S. Rosenschein (1996). "Mechanism Design for Automated Negotiation, and its Application to Task Oriented Domains." Journal of Artificial Intelligence 86(2): 195-244.

Appendix 1

Protocol Analysis Test – Participant Introduction and Instructions

Participant Introduction and Instructions

MADIS Evaluation

Dear Participant,

Thank you for giving the time to this distributed design protocol analysis study. This activity will facilitate us in determining how the MADIS system, an ontological and agent based approach to distributed design support, will impact on the overall collaboration process between dispersed designers. The test will focus primarily on the following two key metrics:

- Information and knowledge retrieval times within a distributed design environment.
- Communication and collaboration efficiency between distributed designers.

You are part of a two-member team that will be assigned two tasks that will require you to use the Lotus Same Time Repository, MADIS agents, and the MADIS web portal. These task description documents will be given to you shortly. It is very important for you to comment on what you are doing when performing these tasks. In other words, you are being asked to "think aloud". One of the tasks will require you to collaborate with the second member of your team in completing the task using the communication technology available. You may use any of the following collaborative tools included to communicate: chat, audio or videoconference, whiteboard.

The time is approximately divided into four parts as follows:

- *30 minutes for introduction and presentation of the MADIS system*
- *30 minutes for the Time-Metric Test*
- *60 minutes for the Collaboration Test*
- *20 minutes for feedback*

Operation of the test

The test is based on a research method called protocol analysis. You will be videotaped while performing the task. Your actions as well as verbalisations will be recorded, so it is extremely important to remember to think aloud while solving the problem. During the session, the only role of the observer is to remind you to talk aloud in case that you forget to verbalize your thoughts.

Thank you for your time.

Appendix 2

Feedback Form for MADIS Evaluation

Feedback Form – MADIS Evaluation

1. **Did you feel constrained in any way by the video camera?**

2. **Did you find the IDIMS system (i.e. IDIMS agents and IDIMS web portal) useful in supporting your task? What is your general opinion about IDIMS?**

3. **Did you feel restricted in any way by the communication technology (i.e. Sametime) used?**

4. **Rate the collaboration process between you and the other member of your team on the following scale.**

very poor very good
1 ----- 2 ---- 3 ---- 4 ---- 5 ---- 6 ---- 7

5. **Rate the ease of use of the Sametime document repository on the following scale.**

very easy very difficult
1 ----- 2 ---- 3 ---- 4 ---- 5 ---- 6 ---- 7

6. **Rate the ease of <u>learning</u> of IDIMS *agents* on the following scale.**

very easy very difficult
1 ----- 2 ---- 3 ---- 4 ---- 5 ---- 6 ---- 7

7. **Rate the ease of <u>use</u> of the IDIMS *agents* on the following scale.**

very easy very difficult
1 ----- 2 ---- 3 ---- 4 ---- 5 ---- 6 ---- 7

8. **Rate the ease of <u>learning</u> of IDIMS *web portal* on the following scale.**

very easy very difficult
1 ----- 2 ---- 3 ---- 4 ---- 5 ---- 6 ---- 7

9. **Rate the ease of <u>use</u> of the IDIMS *web portal* on the following scale.**

very easy very difficult
1 ----- 2 ---- 3 ---- 4 ---- 5 ---- 6 ---- 7

10. **Any comments/suggestions on the *browse* service provided by the IDIMS *agents*:**

11. **Any comments/suggestions on the *search* service provided by the IDIMS *agents*:**

12. **Any comments/suggestions on the *browse* service provided by the IDIMS *web portal*:**

13. **Any comments/suggestions on the *search* service provided by the IDIMS *web portal*:**

14. **List some of the positive aspect(s) of the IDIMS system:**

15. **List some of the negative aspect(s) of the IDIMS system:**

16. **Any further comments or suggestions:**

Thank you for your time!

Appendix 3

The Time-Metric Test Description

Design Requirements Specification Task

Overview

The objective of this task is to quantifiably determine the impact that an ontological and agent based approach to the retrieval of design information and knowledge has over the retrieval of non-structured design information and knowledge. The comparative metric will be time.

Instructions

- For this task you are required to complete the design requirements specification for the Nortel media server product (MediaServerMS), shown in figure 1.
- You will be required to complete design specifications for 15 parts that make up the media server.
- You are required to use the Lotus Same Time document repository for the first five parts, MADIS Agents for the next five parts, and MADIS Semantic Web Portal for the last five parts to find information for each part in order to complete the design requirement specification document.
- This testing document is divided up into three main sections, representing the three methods of information extraction. At the beginning of each section, you are required to record your start time, and subsequently thereafter, you are asked to record the time after the completion of each design specification table.
- Please complete each part table before starting the next one.

Figure 1: Nortel Media Server

Section 1: Lotus Same-Time Repository

Instructions: Please complete the following component design specification tables using the information from the Lotus Same-Time Repository.

Start Time:	

Component 1: BracketMS

Mass	
Function	
Finishing	
Eco-Label	
Processes Used	
Parent Assembly (Is_component_of)	
Version	

Finish Time:	

Component 2: RearInterfaceMS

Mass	
Function	
Finishing	
Eco-Label	
Processes Used	
Parent Assembly (Is_component_of)	
Version	

Finish Time:	

Component 3: ChasisBaseMS

Mass	
Function	
Finishing	
Eco-Label	
Processes Used	
Parent Assembly (Is_component_of)	
Version	
Finish Time:	

Component 4: PowerSupplyCoverMS

Mass	
Function	
Finishing	
Eco-Label	
Processes Used	
Parent Assembly (Is_component_of)	
Version	
Finish Time:	

Component 5: PCB1MS

Mass	
Function	
Finishing	
Eco-Label	
Processes Used	
Parent Assembly (Is_component_of)	
Version	

Finish Time:	

End of Section 1

Section 2: MADIS Agents

Instructions: Please complete the following component design specification tables using the information that the MADIS Agents present.

Start Time:	

Component 1: BezelMS.

Mass	
Function	
Finishing	
Eco-Label	
Processes Used	
Parent Assembly (Is_component_of)	
Version	

Finish Time:	

Component 2: MechHardwareMS.

Mass	
Function	
Finishing	
Eco-Label	
Processes Used	
Parent Assembly (Is_component_of)	
Version	

Finish Time:	

Component 3: Plugs01MS

Mass	
Function	
Finishing	
Eco-Label	
Processes Used	
Parent Assembly (Is_component_of)	
Version	
Finish Time:	

Component 4: PCB2MS

Mass	
Function	
Finishing	
Eco-Label	
Processes Used	
Parent Assembly (Is_component_of)	
Version	
Finish Time:	

Component 5: LedsMS

Mass	
Function	
Finishing	
Eco-Label	
Processes Used	
Parent Assembly (Is component of)	
Version	

Finish Time:	

End of Section 2

Section 3: MADIS Semantic Web Portal

Instructions: Please complete the following component design specification tables using the information that is contained in the MADIS Semantic Web Portal.

Start Time:	

Component 1: MetalSheet 1MS

Mass	
Function	
Finishing	
Eco-Label	
Processes Used	
Parent Assembly (Is_component_of)	
Version	

Finish Time:	

Component 2: MetalSheet 2MS

Mass	
Function	
Finishing	
Eco-Label	
Processes Used	
Parent Assembly (Is_component_of)	
Version	

Finish Time:	

Component 3: NetworkSocketSupportMS

Mass	
Function	
Finishing	
Eco-Label	
Processes Used	
Parent Assembly (Is_component_of)	
Version	
Finish Time:	

Component 4: PinsMS

Mass	
Function	
Finishing	
Eco-Label	
Processes Used	
Parent Assembly (Is_component_of)	
Version	
Finish Time:	

Component 5: LabelMS

Mass	
Function	
Finishing	
Eco-Label	
Processes Used	
Parent Assembly (Is_component_of)	
Version	

Finish Time:	

End of Section 3

THANK YOU FOR YOUR TIME

Appendix 4

Protocol Analysis Transcripts of the Time-Metric Test

Time Metric Test
Sametime Document Repository

Observer's notes:

- No problem with verbalization.
- Users generally found the Groupware Document Repository difficult to use
- Frustration was observed (e.g. "too many clicks")
- Parent Assembly information was very difficult to obtain (few users needed some suggestions)
- As time progresses users worked quicker as they learned from experience

No	Time Start	Time End	Notes
1	00:25	00:40	Identify part file for 'BracketMS'
2	00:40	01:30	Extract part information i.e. mass, function, finishing, eco-label and version
3	01:30	02:40	Get the process used for the part; *"How am I supposed to find this part"*
4	02:40	05:54	Get the parent assembly; Confusion; *"That's not a proper way to look for information"*
Time Duration = 5:29			
1	06:10	06:50	Identify part file for 'RearInterface'
2	06:50	07:40	Extract part information i.e. mass, function, finishing, eco-label and version
3	07:40	09:56	Get the process used for the part; *"This is annoying"*
4	09:56	12:30	Get the parent assembly; Confusion; *"Very cumbersome"; "This is very bad"; Sighs; Observer at 11:01*
Time Duration = 6:20			
1	13:00	13:30	Get the parent assembly;
2	13:30	13:48	Get the process used for the part;
3	13:48	14:00	Identify part file for 'ChasisBaseMS'
4	14:00	15:06	Extract part information i.e. mass, function, finishing, eco-label and version
Time Duration = 2:06			
1	15:10	15:50	Identify part file for 'PowerSupplyCoverMS'
2	15:50	16:28	Extract part information i.e. mass, function, finishing, eco-label and version
3	16:28	16:55	Get the process used for the part;
4	16:55	18:40	Get the parent assembly; *Frustration; Sighs*
Time Duration = 3:30			
1	18:45	19:03	Identify part file for 'PCB1MS'
2	19:03	19:55	Extract part information i.e. mass, function, finishing, eco-label and version
3	19:55	20:27	Get the parent assembly
4	20:27	21:02	Get the process used for the part
Time Duration = 2:17			

Table 1. Time Metric Test – Sametime Document Repository – Subject 1

Time Metric Test
Sametime Document Repository

Observer's notes:

- No problem with verbalization.
- Users generally found the Groupware Document Repository difficult to use
- Frustration was observed (e.g. "too many clicks")
- Parent Assembly information was very difficult to obtain (few users needed some suggestions)
- As time progresses users worked quicker as they learned from experience

No	Time Start	Time End	Notes
1	00:20	00:40	Identify part file for 'BracketMS'
2	00:40	01:22	Extract part information i.e. mass, function, finishing, eco-label and version
3	01:22	05:18	Get the parent assembly; Confusion; *"Pretty annoying"; Observer at 2:30; Sighs; "Loosing my patience"*
4	05:18	06:15	Get the process used for the part
Time Duration = 5:55			
1	06:30	07:10	Identify part file for 'RearInterface'
2	07:10	07:58	Extract part information i.e. mass, function, finishing, eco-label and version; Difficult to navigate though the Document Repository
3	07:58	08:41	Get the process used for the part
4	08:41	09:55	Get the parent assembly; *"This is the most annoying part"*
Time Duration = 3:25			
1	10:00	10:30	Identify part file for 'ChasisBaseMS'
2	10:30	11:10	Extract part information i.e. mass, function, finishing, eco-label and version
3	11:10	11:42	Get the process used for the part
4	11:42	12:20	Get the parent assembly
Time Duration = 2:20			
1	12:25	12:55	Identify part file for 'PowerSupplyCoverMS'
2	12:55	13:30	Extract part information i.e. mass, function, finishing, eco-label and version
3	13:30	14:10	Get the process used for the part
4	14:10	15:15	Get the parent assembly; Uses the function of the part as a hint
Time Duration = 2:50			
1	15:20	15:45	Identify part file for 'PCB1MS'
2	15:45	16:29	Extract part information i.e. mass, function, finishing, eco-label and version
3	16:29	19:29	Get the process used for the part; Confusion
4	19:29	20:43	Get the parent assembly
Time Duration = 5:23			

Table 2. Time Metric Test – Sametime Document Repository – Subject 2

Time Metric Test *Sametime Document Repository*			
Observer's notes: • No problem with verbalization. • Users generally found the Groupware Document Repository difficult to use • Frustration was observed (e.g. "too many clicks") • Parent Assembly information was very difficult to obtain (few users needed some suggestions) • As time progresses users worked quicker as they learned from experience			
No	**Time Start**	**Time End**	**Notes**
1	00:30	00:30	Identify part file for 'BracketMS'
2	01:30	01:30	Extract part information i.e. mass, function, finishing, eco-label and version
3	02:10	05:02	Get the process used for the part; *Observer at 03:20;* Difficulties finding the process used
4	05:02	18:20	Get the parent assembly; *Non-verbal codes* Confusion; *"This isn't easy at all!"; Observer at 10:00; Observer at 17:10*
Time Duration = 17:50			
1	18:30	19:00	Identify part file for 'RearInterface'
2	19:00	19:40	Extract part information i.e. mass, function, finishing and eco-label;
3	19:40	20:41	Get the process used for the part
4	20:41	26:00	Get the parent assembly; High concentration
5	26:00	26:50	Get the part version
Time Duration = 8:20			
1	27:00	27:15	Identify part file for 'ChasisBaseMS'
2	27:15	28:15	Extract part information i.e. mass, function, finishing, eco-label and version
3	28:15	30:27	Get the process used for the part
4	30:27	31:22	Get the parent assembly
Time Duration = 4:22			
1	31:27	31:50	Identify part file for 'PowerSupplyCoverMS'
2	31:50	32:24	Extract part information i.e. mass, function, finishing, eco-label and version
3	32:34	33:20	Get the process used for the part
4	33:20	33:57	Get the parent assembly; Uses the function of the part as a hint
Time Duration = 2:30			
1	34:08	34:30	Identify part file for 'PCB1MS'
2	34:30	35:28	Extract part information i.e. mass, function, finishing, eco-label and version
3	35:28	37:24	Get the process used for the part; Confusion
4	37:24	38:25	Get the parent assembly; The subject works faster learning from experience
Time Duration = 4:17			

Table 3. Time Metric Test – Sametime Document Repository – Subject 3

Time Metric Test ***Sametime Document Repository***			
Observer's notes: • No problem with verbalization. • Users generally found the Groupware Document Repository difficult to use • Frustration was observed (e.g. "too many clicks") • Parent Assembly information was very difficult to obtain (few users needed some suggestions) • As time progresses users worked quicker as they learned from experience			
No	**Time Start**	**Time End**	**Notes**
1	00:40	01:40	Identify part file for 'BracketMS'
2	01:40	03:40	Extract part information i.e. mass, function, finishing, eco-label and version
3	03:40	04:50	Get the process used for the part; *Observer at 03:55;* Difficulties finding the process used
4	04:50	06:15	Get the parent assembly; *Non-verbal codes* Confusion; *Observer at 06:04*
Time Duration = 5:35			
1	06:17	06:43	Identify part file for 'RearInterface'
2	06:43	07:38	Extract part information i.e. mass, function, finishing and eco-label; *"There is quite a lot of clicks"*
3	07:38	08:30	Get the process used for the part
4	08:30	09:14	Get the parent assembly; High concentration
Time Duration = 2:57			
1	09:26	10:01	Identify part file for 'ChasisBaseMS'
2	10:01	10:43	Extract part information i.e. mass, function, finishing, eco-label and version
3	10:43	11:35	Get the process used for the part; High concentration
4	11:35	12:26	Get the parent assembly
Time Duration = 3:00			
1	12:30	12:47	Identify part file for 'PowerSupplyCoverMS'
2	12:47	13:28	Extract part information i.e. mass, function, finishing, eco-label and version
3	13:28	15:31	Get the process used for the part
4	15:31	16:20	Get the parent assembly
Time Duration = 3:50			
1	16:20	15:53	Identify part file for 'PCB1MS'
2	15:53	18:09	Extract part information i.e. mass, function, finishing, eco-label and version
3	18:09	20:14	Get the process used for the part; Confusion
4	20:14	22:00	Get the parent assembly; Difficulties locating information
Time Duration = 5:40			

Table 4. Time Metric Test – Sametime Document Repository – Subject 4

Time Metric Test ***MADIS Agents***			
Observer's notes: • No problem with verbalization. • Agents were found easy to use and intuitive • The search service provided by user agents was preferred • A more relaxed approach to performing the task as observed			
No	**Time Start**	**Time End**	**Notes**
1	21:38	22:54	Uses the search service to locate the required part i.e. name like "BezelMS"
2	22:54	24:09	Extracts part information from Agent Query Results i.e. mass, function, finishing, eco-label, process, parent assembly and version; *"This is much better"*
Time Duration = 2:27			
No	**Time Start**	**Time End**	**Notes**
1	26:43	27:14	Uses the search service to locate the required part i.e. name like "MechHardwareMS"
2	27:14	28:14	Extracts part information from Agent Query Results i.e. mass, function, finishing, eco-label, process, parent assembly and version; *Relaxation*
Time Duration = 1:31			
1	28:14	28:31	Uses the search service to locate the required part i.e. name like "Plugs01MS"
2	28:31	29:23	Extracts part information from Agent Query Results i.e. mass, function, finishing, eco-label, process, parent assembly and version; *"There it is"*
Time Duration = 1:09			
1	29:30	29:45	Uses the search service to locate the required part i.e. name like "PCB2MS"
2	29:45	32:20	Extracts part information from Agent Query Results i.e. mass, function, finishing, eco-label, process, parent assembly and version; *"This is only routine"*
Time Duration = 2:50			
1	32:20	32:35	Uses the search service to locate the required part i.e. name like "LedsMS"
2	32:35	33:30	Extracts part information from Agent Query Results i.e. mass, function, finishing, eco-label, process, parent assembly and version; *Relaxation*
Time Duration = 1:10			

Table5. Time Metric Test – MADIS Agents – Subject 1

Time Metric Test *MADIS Agents*			
Observer's notes: • No problem with verbalization. • Agents were found easy to use and intuitive • The search service provided by user agents was preferred • A more relaxed approach to performing the task as observed			
No	**Time Start**	**Time End**	**Notes**
1	22:00	23:30	Uses the search service to locate the required part i.e. name like "BezelMS"
2	23:30	24:25	Extracts part information from Agent Query Results i.e. mass, function, finishing, eco-label, process, parent assembly and version
Time Duration = 2:25			
1	24:29	24:55	Uses the search service to locate the required part i.e. name like "MechHardwareMS"
2	24:55	25:45	Extracts part information from Agent Query Results i.e. mass, function, finishing, eco-label, process, parent assembly and version
Time Duration = 1:16			
1	25:50	26:14	Uses the search service to locate the required part i.e. name like "Plugs01MS"
2	26:14	27:00	Extracts part information from Agent Query Results i.e. mass, function, finishing, eco-label, process, parent assembly and version; *"This is much faster"*
Time Duration = 1:10			
1	27:05	27:30	Uses the search service to locate the required part i.e. name like "PCB2MS"
2	27:30	29:28	Extracts part information from Agent Query Results i.e. mass, function, finishing, eco-label, process, parent assembly and version; *Relaxation*
Time Duration = 2:23			
1	29:33	29:56	Uses the search service to locate the required part i.e. name like "LedsMS"; Not frustrated; The agent is trusted to do its job
2	29:56	30:49	Extracts part information from Agent Query Results i.e. mass, function, finishing, eco-label, process, parent assembly and version; *Smile*
Time Duration = 1:16			

Table 6. Time Metric Test – MADIS Agents – Subject 2

Time Metric Test *MADIS Agents*			
Observer's notes: • No problem with verbalization. • Agents were found easy to use and intuitive • The search service provided by user agents was preferred • A more relaxed approach to performing the task as observed			
No	**Time Start**	**Time End**	**Notes**
1	39:00	41:25	Uses the search service to locate the required part i.e. name like "BezelMS"
2	41:25	42:41	Extracts part information from Agent Query Results i.e. mass, function, finishing, eco-label, process, parent assembly and version; *"This is very easy"*
Time Duration = 3:41			
1	42:45	43:10	Uses the search service to locate the required part i.e. name like "MechHardwareMS"
2	43:10	44:00	Extracts part information from Agent Query Results i.e. mass, function, finishing, eco-label, process, parent assembly and version
Time Duration = 1:15			
1	44:05	44:50	Uses the search service to locate the required part i.e. name like "Plugs01MS"
2	44:50	45:47	Extracts part information from Agent Query Results i.e. mass, function, finishing, eco-label, process, parent assembly and version;
Time Duration = 1:42			
1	45:52	46:09	Uses the search service to locate the required part i.e. name like "PCB2MS"
2	46:09	47:15	Extracts part information from Agent Query Results i.e. mass, function, finishing, eco-label, process, parent assembly and version
Time Duration = 1:23			
1	47:20	47:30	Uses the search service to locate the required part i.e. name like "LedsMS"; Not frustrated; The agent is trusted to do its job
2	47:30	49:00	Extracts part information from Agent Query Results i.e. mass, function, finishing, eco-label, process, parent assembly and version; *Smile; Relaxation*
Time Duration = 1:40			

Table 7. Time Metric Test – MADIS Agents – Subject 3

Time Metric Test ***MADIS Agents***			
Observer's notes: • No problem with verbalization. • Agents were found easy to use and intuitive • The search service provided by user agents was preferred • A more relaxed approach to performing the task as observed			
No	**Time Start**	**Time End**	**Notes**
1	22:30	24:02	Uses the search service to locate the required part i.e. name like "BezelMS"
2	24:02	25:09	Extracts part information from Agent Query Results i.e. mass, function, finishing, eco-label, process, parent assembly and version; *"This is a lot more informative"; "This is easier: with a simple search you get what you want"*
Time Duration = 2:39			
1	25:38	25:45	Uses the search service to locate the required part i.e. name like "MechHardwareMS"
2	25:45	26:30	Extracts part information from Agent Query Results i.e. mass, function, finishing, eco-label, process, parent assembly and version
Time Duration = 0:56			
1	26:35	27:00	Uses the search service to locate the required part i.e. name like "Plugs01MS"; *"Learning curve is a lot quicker"*
2	27:00	27:40	Extracts part information from Agent Query Results i.e. mass, function, finishing, eco-label, process, parent assembly and version; *"This is a lot better!"*
Time Duration = 1:25			
1	27:44	28:09	Uses the search service to locate the required part i.e. name like "PCB2MS"; *"Search is accurate"*
2	28:09	28:57	Extracts part information from Agent Query Results i.e. mass, function, finishing, eco-label, process, parent assembly and version; Subject very happy with the agent performance
Time Duration = 1:13			
1	29:04	29:24	Uses the search service to locate the required part i.e. name like "LedsMS"; Not frustrated;
2	29:24	29:58	Extracts part information from Agent Query Results i.e. mass, function, finishing, eco-label, process, parent assembly and version; The interface is easy to use, very intuitive.
Time Duration = 0:54			

Table 8. Time Metric Test – MADIS Agents – Subject 4

Time Metric Test *MADIS Web Portal*			
Observer's notes: • No problem with verbalization. • The Web Portal was easy to navigate • The interface was considered friendly • Most subjects experienced both Browse and Search services provided			
No	**Time Start**	**Time End**	**Notes**
1	34:00	34:28	Identify the part "MetalSheet1MS" using Browse service
2	34:28	35:55	Extracts part information i.e. mass, function, finishing, eco-label, process, parent assembly and version; *Relaxation; "This is routine"*
Time Duration = 1:55			
1	36:00	36:20	Identify the part "MetalSheet2MS" using Browse service
2	36:20	37:00	Extracts part information i.e. mass, function, finishing, eco-label, process, parent assembly and version
Time Duration = 1:00			
1	37:05	38:52	Identify the part "NetworkSocketSupportMS" using Search service but prefers Browse
2	38:52	39:50	Extracts part information i.e. mass, function, finishing, eco-label, process, parent assembly and version
Time Duration = 2:45			
1	39:55	40:20	Identify the part "PinsMS" using Search service
2	40:20	41:10	Extracts part information from Search Results i.e. mass, function, finishing, eco-label, process, parent assembly and version;
Time Duration = 1:15			
1	41:15	41:39	Identify the part "LabelMS" using Search service
2	41:39	42:40	Extracts part information from Search Results i.e. mass, function, finishing, eco-label, process, parent assembly and version
Time Duration = 1:25			

Table 9. Time Metric Test – MADIS Web Portal – Subject 1

Time Metric Test *MADIS Web Portal*			
Observer's notes: • No problem with verbalization. • The Web Portal was easy to navigate • The interface was considered friendly • Most subjects experienced both Browse and Search services provided			
No	**Time Start**	**Time End**	**Notes**
1	31:14	31:59	Identify the part "MetalSheet1MS" using Search service
2	31:59	33:00	Extracts part information i.e. mass, function, finishing, eco-label, process, parent assembly and version
Time Duration = 1:46			
1	33:05	34:24	Identify the part "MetalSheet2MS" using Browse service, but switch to Search service *"I prefer the search"*
2	34:24	35:40	Extracts part information i.e. mass, function, finishing, eco-label, process, parent assembly and version; *"Yes, I have the information"*
Time Duration = 2:35			
1	35:45	36:15	Identify the part "NetworkSocketSupportMS" ; *"Much faster"*
2	36:15	37:00	Extracts part information i.e. mass, function, finishing, eco-label, process, parent assembly and version; *"Very good"*
Time Duration = 1:45			
1	37:10	37:30	Identify the part "PinsMS" using Search service
2	37:30	38:25	Extracts part information from Search Results i.e. mass, function, finishing, eco-label, process, parent assembly and version
Time Duration = 1:15			
1	38:30	39:00	Identify the part "LabelMS" using Search service
2	39:00	39:50	Extracts part information from Search Results i.e. mass, function, finishing, eco-label, process, parent assembly and version
Time Duration = 1:20			

Table 10. Time Metric Test – MADIS Web Portal – Subject 2

Time Metric Test *MADIS Web Portal*			
Observer's notes: • No problem with verbalization. • The Web Portal was easy to navigate • The interface was considered friendly • Most subjects experienced both Browse and Search services provided			
No	**Time Start**	**Time End**	**Notes**
1	49:25	51:10	Play with the system
2	51:10	51:56	Identify the part "MetalSheet1MS" using Browse service; *"There we are!"*
3	51:56	53:10	Extracts part information i.e. mass, function, finishing, eco-label, process, parent assembly and version
Time Duration = 3:45			
1	53:20	53:25	Identify the part "MetalSheet2MS" using Browse service
2	53:25	54:20	Extracts part information i.e. mass, function, finishing, eco-label, process, parent assembly and version
Time Duration = 1:00			
1	54:25	54:30	Identify the part "NetworkSocketSupportMS" using Browse service
2	54:30	55:28	Extracts part information i.e. mass, function, finishing, eco-label, process, parent assembly and version
Time Duration = 1:03			
1	55:30	55:45	Identify the part "PinsMS" using Browse service
2	55:45	56:41	Extracts part information i.e. mass, function, finishing, eco-label, process, parent assembly and version
Time Duration = 1:21			
1	56:46	56:59	Identify the part "LabelMS" using Browse service
2	56:59	57:53	Extracts part information i.e. mass, function, finishing, eco-label, process, parent assembly and version
Time Duration = 1:07			

Table 11. Time Metric Test – MADIS Web Portal – Subject 3

Time Metric Test *MADIS Web Portal*			
Observer's notes: • No problem with verbalization. • The Web Portal was easy to navigate • The interface was considered friendly • Most subjects experienced both Browse and Search services provided			
No	**Time Start**	**Time End**	**Notes**
1	30:45	31:40	Identify the part "MetalSheet1MS" using Browse service
2	31:40	32:42	Extracts part information i.e. mass, function, finishing, eco-label, process, parent assembly and version; *"This is very good"*
Time Duration = 1:57			
1	32:46	33:30	Identify the part "MetalSheet2MS" using Browse service
2	33:30	34:10	Extracts part information i.e. mass, function, finishing, eco-label, process, parent assembly and version; *"This is easy"*
Time Duration = 1:24			
1	34:15	34:25	Identify the part "NetworkSocketSupportMS" using Browse service
2	34:25	35:10	Extracts part information i.e. mass, function, finishing, eco-label, process, parent assembly and version; *"The interface is nice, easy to navigate"*
Time Duration = 0:55			
1	35:10	35:40	Identify the part "PinsMS" using Browse service
2	35:40	36:15	Extracts part information i.e. mass, function, finishing, eco-label, process, parent assembly and version
Time Duration = 1:05			
1	36:20	36:30	Identify the part "LabelMS" using Browse service
2	36:30	37:20	Extracts part information i.e. mass, function, finishing, eco-label, process, parent assembly and version
Time Duration = 1:00			

Table 12. Time Metric Test – MADIS Web Portal – Subject 4

Appendix 5

The Collaboration Test Description

Measuring Collaboration Efficiency: Obtain Product Mass Task

Overview

The objective of the second testing phase is to determine what impact will the application of ontologies and software agents have on a distributed collaboration process in terms of overall communication efficiency between two distributed designers (Designer A and Designer B). The design task is centred on determining the mass of the smoke alarm product from the design information relating to it. The smoke alarm has 9 components and 4 subassemblies. There will be two instances of the task. To complete the first instance of the task the distributed designers will be required to virtually collaborate with each other, by communicating design information and knowledge about the smoke alarm, using firstly just the Lotus Sametime groupware technologies. The second instance will also require the two distributed designers to complete the same task, but in this instance they will use MADIS agents and semantic web portal. Both sessions will be video recorded for protocol analysis, whereby the impact of the MADIS ontologies and software agents had on the collaboration process can be evaluated. A scenario for the design task will be presented for designer A and designer B for each instance of the test.

Scenario for test using Sametime groupware technology

Designer A is required to calculate the total mass of the smoke alarm product, but will need the assistance of Designer B in order to complete the task. Designer A is expecting Designer B to calculate the mass of a particular subassembly of the smoke alarm (PCBAssembly). Designer B will also assist Designer A with any problems that he/she encounters during the task. As Designer A starts his/her task by searching the Sametime product repository, Designer B will be working on his/her own task of computing the mass of the 'PCBAssembly'. The product repository will be divided up into two sub-repositories, *assemblies/subassemblies*, and *parts*. The assemblies/subassemblies repository will contain excel files that are named after various product assemblies/subassemblies, where as the parts repository will contain excel files that are named after various parts. The assemblies/subassemblies excel files contain information relating to the part names that make up the specific assemblies. The part excel files

contain various information about the parts. This information also includes the mass of the part. In order for Designer A to calculate the total mass for the smoke alarm, she/he must find the mass of each part. However, Designer A will discover that there is no information relating to one of the subassemblies *('CoverAssembly')* of the smoke alarm. Consequently, Designer A will not be able to obtain information about the parts that make up the 'CoverAssembly', and will therefore be unable to calculate the total mass for the smoke alarm. However, Designer A knows that Designer B was responsible for the design of the 'CoverAssembly'. Designer A communicates his/her problem to Designer B using the collaborative tools that are available from the groupware system. Designer B, (who is working on his/her own task relating to the PCBAssembly), realises that he/she did not upload the excel files relating to the 'CoverAssembly' and related parts to the Sametime repository. Designer B then has to look through all his/her assembly/subassembly and part excel files, which are located in file directories on his/her local system. From his/her own local file directories, Designer B will be able to identify the related parts that make up the 'CoverAssembly', and consequently upload the appropriate excel files to the Same Time repository. Designer B then communicates to Designer A that the information he/she requires is now contained within the Sametime repository. Designer A continues with his/her task of obtaining the mass of the rest of the parts, after which he/she then communicates a request to designer B for the PCBAssembly mass, thus enabling Designer A to complete the task of calculating the total mass of the smoke alarm.

Scenario for test incorporating MADIS components

The task for the subjects in this scenario also relates to calculating the total mass for the smoke alarm. In this instance of the test the subjects change roles. Designer A for the first test instance becomes Designer B for this instance, and Designer B for the first test instance becomes Designer A for this instance. Both subjects can use the collaborative tools from MADIS web portal (e.g. Sametime instant messaging, whiteboard, videoconference and meeting room), in conjunction with the MADIS components. Again, as in the first instance Designer A is required to calculate the total mass of the smoke alarm product, and will again need the assistance of Designer B in order to complete the task. Designer A is expecting Designer B to calculate the mass of a particular

subassembly of the smoke alarm (PCBAssembly). Designer B will also assist Designer A with any problems that he/she encounters during the task. In this instance Designer A starts the task by using either of the MADIS components for extracting information from the product ontology. Designer B will start on his/her own task of computing the mass of the 'PCBAssembly' and will also use either of the MADIS components. Again in this instance Designer A will discover that is no information relating to one of the subassemblies ('CoverAssembly') that make up the Smoke Alarm. Consequently, Designer A will not be able to obtain information amount the components of the 'CoverAssembly', and will therefore be unable to calculate the total mass for the smoke alarm. As in the initial scenario, Designer A knows that Designer B is responsible for the design of the 'CoverAssembly', and communicates his/her problem to Designer B. Designer B (who is working on his/her own task relating to the PCBAssembly) realises that he/she did not save any information relating to the design of the parts associated with the 'CoverAssembly' to the product ontology. In this instance Designer B starts up Pro/Engineer and finds the appropriate CAD files that are associated with the smoke alarm. Amongst the associated smoke alarm CAD files, Designer B will find assembly and part Pro/E files for the CoverAssembly, thus enabling him/her to identify the parts that make up the CoverAssembly. Designer B then opens the CAD model, for each of the parts associated with the CoverAssembly, within the Pro/E environment. He/she then uses the MADIS application agent (that has been integrated within the Pro/E environment) to extract the part name and part mass from the CAD model and append them to the product ontology. Designer B then communicates to Designer A that the required information should be available in MADIS now. Designer A uses the MADIS components to extract the mass information for the parts that he/she requires from the product ontology. Designer A continues with his/her task of obtaining the mass of the rest of the parts, after which he/she then communicates a request to designer B for the PCBAssembly mass, thus enabling Designer A to complete the task of calculating the total mass of the smoke alarm.

Instructions for Groupware test scenario for Designer A

- You are required to calculate the total mass of a smoke alarm product ('SmokeAlarm').
- Designer B will assist you in calculating the mass of the smoke alarm, as he/she will be responsible for calculating the mass of a particular subassembly (PCBAssembly) of the smoke alarm. (Note: Do not calculate the mass of the PCBAssembly).
- You should note that specific information you require has been deliberately omitted for this scenario.
- When you encounter any information omission you must collaborate with Designer B, as he/she is responsible for it. (Note: For any other problems you encounter in this scenario, you must collaborate with Designer B).

- *For this task you must use the product information that is contained within the Lotus Same-Time repository.*
- *You will be required to collaborate with Designer B, using the available groupware tools, e.g. Instant Messaging, Voice/Video Conferencing, Whiteboard etc, for any problems you encounter during completion of your task.*

Instructions for Groupware test scenario for Designer B

- You are required to calculate the mass of one of the subassemblies of the smoke alarm (PCBAssembly).
- Designer A will communicate with you at various stages during this scenario regarding problems relating to missing repository information that you forgot to upload. Based on your communication with Designer A you are requested to identify and find the missing information that Designer A requires, from your local product file directory: C:\Product Repository.
- You are then requested to upload this information to the same-time repository and inform designer A that the information he/she requires is now contained within the same-time repository.
- Complete your task of calculating the mass for the PCBAssembly, as designer A needs this information in order to complete his/her task.

- *For this task you must use the product information that is contained within the Lotus Same-Time repository.*
- *You will be required to collaborate with Designer A, using the available groupware tools, e.g. Instant Messaging, Voice/Video Conferencing, and Whiteboard etc, to assist him/her during any problems that he/she encounters during this scenario.*

Instructions for MADIS components test scenario for Designer A

- You are required to calculate the total mass of a smoke alarm product ('SmokeAlarm')
- Designer B will assist you in calculating the mass of the smoke alarm, as he/she will be responsible for calculating the mass of a particular subassembly (entitled PCBAssembly) of the smoke alarm. (Note: Do not calculate the mass of the PCBAssembly).
- You should note that specific information you require has been deliberately omitted for this scenario
- When you encounter this information omission you must collaborate with Designer B, as he/she is responsible for it. (Note: For any problems you encounter or queries you have with this scenario, you must collaborate with Designer B).

- *For this task you must use the product information that is contained within the product ontology.*
- *You must use either the MADIS web portal or MADIS agents to extract the required information from the product ontology.*
- *You will be required to collaborate with Designer B, using the available groupware tools, e.g. Instant Messaging, Voice/Video Conferencing, Whiteboard etc, for any problems you encounter during completion of your task.*

Instructions for MADIS components test scenario for Designer B

- You are required to calculate the mass of one of the subassemblies of the smoke (PCBAssembly).
- Designer A will communicate with you at various stages during this scenario regarding problems relating to information you were responsible for, but subsequently forgot to append to the product ontology. Based on your communication with Designer A, you are requested to identify and find the missing information relating to the specific information that Designer A requires from the Pro/E files contained on you local system. These files are located at C:\CAD Files\smoke alarm
- Once you have identified these files, you must open them within the Pro/E environment and save component information from the CAD model to the product ontology.
- Communicate to Designer A that the information he/she requires has now been appended to the product ontology.
- Complete your task of calculating the mass for the PCB Assembly, as designer A needs this information in order to complete his/her task.

- *For this task you must use the product information that is contained within the product ontology.*
- *You must use either the MADIS web portal or MADIS agents to extract the required information from the product ontology.*
- *You must use the application agent to save component information from the CAD file to the product ontology.*
- *You will be required to collaborate with Designer A, using the available groupware tools, e.g. Instant Messaging, Voice/Video Conferencing, Whiteboard etc, for any problems you encounter during completion of your task.*

Observer's notes:

- The two video cameras used in the testing were completely ignored by subjects
- Confusion and irritation was observed throughout the scenario
- Audio, instant messaging and white board were used
- In some cases, subjects did not follow the exact instructions given e.g. delegation of work by Designer A, Designer B computing the total mass for the Smoke Alarm instead of Designer A

No	Time Start	Duration	Give suggestion/opinion/orientation (GS) Ask for suggestion/opinion/orientation (AS) Agree (A) Disagree (D) Show solidarity (S) Show tension (T) Show tension release (TR)		Notes
			Designer A	Designer B	
1	0:00	7:15	5GS; 2AS; 4S; 2A; 2T	4GS; 3AS; 1A; 1S	A wants to establish a collaboration process from the beginning delegating tasks for the scenario; A and B work together to identify Smoke Alarm subassemblies; Confusion; A – *"This isn't going very well"; "I'm lost here"* B – Observer at 3:47 (clarification of tasks) A – Observer at 7:02 (clarification of tasks)
2	7:15	4:40	2AS; 2A; 2S	5GS; 2AS; 1S	A informs B about missing CoverAssembly file B uploads CoverAssembly file but creates some confusion by uploading the wrong file (i.e. ChassisAssembly)
3	11:55	3:10	1A; 1S; 1T; 1TR	3GS; 1A; 1S; 1AS; 1T	A computes mass for BaseAssembly B computes mass for PCBAssembly B – Observer at 12:33
4	15:05	5:36	6AS; 2GS; 1S	2GS; 4AS; 2S	A asks B to upload CoverAssembly components i.e. Button and Cover B uploads the Button and Cover files under Product Parts A – confusion : *"Where did you upload the Button?"* B finishes the task of computing the mass for PCBAssembly
5	20:41	4:29	2GS; 3AS; 2A;2S; 1T; 1TR	2GS; 1AS; 1A; 1S; 1T; 1TR	Using the whiteboard, A tells B the mass for CoverAssembly and BaseAssembly B computes the total mass of the SmokeAlarm by adding the PCBAssembly mass to the value received from A A – Observer at 23:10 (to remind A that he is responsible for calculating the total mass)

Table 2. Episodes for the second Sametime Document Repository session

Observer's notes:

- The two video cameras used in the testing were completely ignored by subjects
- Audio technology was mainly used for communication
- Subjects were generally relaxed
- Both the Web Portal and the Agents were engaged to support the task performance; Some subjects relied more on their Personal Agent while others preferred the interface of the Web Portal.
- The search service was preferred to the browse
- The tasks were easier to complete using the MADIS system

No	Time Start	Duration	Give suggestion/opinion/orientation (GS) Ask for suggestion/opinion/orientation (AS) Agree (A) Disagree (D) Show solidarity (S) Show tension (T) Show tension release (TR)		Notes
			Designer A	Designer B	
1	0:00	3:01			A identifies Smoke Alarm components using Search Agent first but then Web Portal – browse B uses Search Agent to identify PCBAssembly components
2	3:01	1:28	1AS	1A	A asks B for PCBAssembly mass A calculates mass for BaseAssembly B calculates mass for PCBAssembly
3	4:29	4:33	2GS; 2AS;2S	5GS; 2AS; 2S	A asks for CoverAssembly information using audio & chat A waits for B to reply B saves Button & Cover parts from ProE using Application Agent
4	9:02	2:30	1AS	1S	A calculates the CoverAssembly mass A – smile, happy face, relaxation (body posture) B continues mass calculation for PCBAssembly using Web Portal – search A reminds B about the previously requested PCBAssembly mass
5	11:32	0:35	1S	1GS	B communicates the PCBAssembly mass A finishes the task of calculating the SmokeAlarm mass

Table 3. Episodes for the first MADIS session

Observer's notes:

- The two video cameras used in the testing were completely ignored by subjects
- Audio technology was mainly used for communication
- Subjects were generally relaxed
- Both the Web Portal and the Agents were engaged to support the task performance; Some subjects relied more on their Personal Agent while others preferred the interface of the Web Portal.
- The search service was preferred to the browse
- The tasks were easier to complete using the MADIS system

No	Time Start	Duration	Give suggestion/opinion/orientation (GS) Ask for suggestion/opinion/orientation (AS) Agree (A) Disagree (D) Show solidarity (S) Show tension (T) Show tension release (TR)		Notes
			Designer A	Designer B	
1	0:00	5:49	2GS; 1AS; 1A; 2S	2GS; 2AS; 1S	A identifies Smoke Alarm components using Search Agent B uses the Web Portal to familiarize himself with the Smoke Alarm structure Needless communication B – smile; This episode exists mainly because subjects took longer to adapt to the system.
2	5:49	4:16	1AS; 1S	1GS	A asks B for PCBAssembly mass A calculates mass for BaseAssembly B calculates mass for PCBAssembly B – observer at 8:41 to clarify tasks B informs A the mass for the PCBAssembly
3	10:05	4:25	2GS; 3AS; 1A; 1S	4GS; 2AS; 1S	A asks for CoverAssembly information using audio: *"Ok, so if I do a search now I should see them"* (i.e. the Button and Cover parts of the CoverAssembly) A continues mass calculation for BaseAssembly B saves Button & Cover parts from ProE using Application Agent: *"That's it now"*
4	14:30	4:33	2GS; 1AS; 1S	3GS; 1A; 1S	A calculates the CoverAssembly mass: *"Ok, I see them now!"* A – smile B supports A in finishing the task
5	19:03	0:40	1GS; 1S	1S	A completes the mass calculation for SmokeAlarm B supports A in finishing the task

Table 4. Episodes for the second MADIS session