Master Thesis in Information Systems

Faculty of Economics and Social Sciences
Agder University College - Autumn 2006

Positive and Negative Factors in Agent Oriented Software Development – A Case Study

Einar Jakob Hovland

Preface

This thesis is the concluding works of a two-year full time study at the Information Systems Master study program at Agder University College in Kristiansand, Norway. The research has been conducted throughout one semester running from August to December 2006, although preparations started during the spring semester that same year.

The main objective of my research was to identify differing factors in agent oriented software development in contrast to more traditional software engineering approaches. By building a prototype of a patient scheduling system using agent technologies, I have attempted to validate suggested factors from prior research using my own experiences. This has hopefully given some interesting results, contributing to a yet rather unexplored area within the agent oriented research communities.

My interest in the agent system field was initiated through a system engineering course conducted earlier in my master program. Already interested in software development, I saw this field as an exciting turning point in software engineering. Over this last semester my interest have only grown, and though regarding this evolution in software development in a more realistic light, I still see an exciting future with endless opportunities for this type of software engineering.

Finally I would like to thank my supervisor Even Åby Larsen for excellent guidance and invaluable constructive remarks throughout the project.

Agder University College, 14.12.2006

Einar Jakob Hovland

Abstract

This thesis attempts to pinpoint the main differences in agent oriented software engineering as opposed to more legacy approaches. The agent paradigm, still in its infancy, is believed to offer exciting new prospects to the process of modern complex information systems development, but although many tools, frameworks and applications are readily available, there is as of yet no specific research on how these technologies affects and differ the development process.

The research approach in finding these factors was two-folded. By conducting a comprehensive literature study, a set of implications was derived to represent suggested factors from earlier research. These terms could be related to well-established agent system characteristics and behaviours, and based on these results a framework model was devised to present a hypothesis of all terms and relationships discovered.

In parallel with this work, a prototype of an agent based patient scheduling system was developed using agent oriented methods and tools. Besides contributing to the validation of the agent approach in general, experiences from this work was used to evaluate the findings in the hypotheses model, and a final factor and relationship model was developed through comparison and discussion of both prior research and my own experiences.

The results clarify both positive and negative aspects of the agent oriented approach. I identified in particular three aspects of this paradigm that separates it from more traditional approaches:

- The high level of abstraction that an agent represents helps in defining components of complex systems during analysis and design, but poses problems in its lack of detail during implementation.

- Being autonomous entities, agent oriented development provides a highly modular development approach, helping to separate complex structures in an advantageous divide-and-conquer style, but consequently increase complexity in cooperation among these autonomous components.

- Agents' close relationship to their environment or domain provides great awareness and situatedness opportunities in highly modular systems, but this relationship comes with the price of increasingly complex interaction procedures.

Although these results hopefully can be regarded as a valuable contribution to the agent systems engineering research field, they are by no means exhaustive. Much work remains in clarifying the role of agent oriented approaches in the software development evolution, before the agent paradigm can be established as a validated commercial and industrial option.

Table of Contents

List of Figures

1 Introduction

From machine-code, via assembly, functional languages and object oriented approaches; software agents are seen as a further abstraction down the software engineering path, helping to overcome the ever-increasing complexity in designing modern information systems. Agent-oriented software engineering has by many researchers been dubbed the new paradigm in software development, and from its original concepts in the early 80s agents and agent systems are now active research areas in computer science.

Early promising theories and frameworks have contributed to this seemingly successful evolution, ensuring vast amounts of literature released on the subject. However, as there still only exists a minimal number of documented industrial and commercial test cases, many researchers (e.g. Brazier et al, 1997; Jennings et al, 1998) warns that this rather new technology is still immature, and needs to be rigorously tested by practitioners before the hype can begin.

Patient scheduling systems are seen as a typical challenge for tomorrow in such a context (Bartelt et al, 2002). Coordinating and processing a vast amount of complex variables, such a system should be designed to stock and schedule a wide range of resources based on the patients heath condition and availability, giving rise to complicated data control and optimization problems.

1.1 Thesis Objectives and Research Question

Taking these theories into account, this thesis will try to identify some key factors in agent systems development processes. By developing a prototype of a patient scheduling system, the thesis will both show how such a system might be designed, but more importantly collect evidence as to which elements of the agent system engineering process that differentiate this approach from more traditional attempts. The first research question is:

1. *Given the ever-increasing size and complexity of modern information systems, can agent technology contribute in making the build- and maintenance processes of such systems feasible?*

To answer this question, this thesis will attempt to model a patient scheduling system prototype, encapsulating agent system characteristics like coordination of resources, negotiation and optimization. This process will involve working through a methodology, documenting the experiences as the design evolves. These results should give evidence to answer the second research question:

2. *What are the significant positive and negative aspects when building such type of systems using the agent approach and how do they relate to agent and agent system characteristics?*

This second question will be evaluated by comparing the experiences found during development of the prototype with a set of relations found in a literature review. This latter process will result in a hypothesis model so that the results from the engineering process effectively can be compared to prior research.

1.2 Contributions to Research

As the two research questions pose a two-folded approach, contributing to research in different ways. Firstly, the before-mentioned hype, and the rather chaotic nature of theories within the research of agents and agent-systems poses a serious validity problem for the agent research branch (e.g. Brazier et al, 1997; Jennings et al, 1998). Still so early in its maturity, agent-system research is now in need of validation before this new software paradigm can be established.

This thesis will contribute in such by building and testing an agent-based prototype to effectively schedule patients at a medical institution. By validating operations like cooperation and optimization by and between agents, humans and data, the simulation should contribute to the growing list of agent-oriented applications, demonstrating that building and using such types of scheduling systems would be, in practice, feasible. Looking back at the research questions, this contribution will be made by answering question one.

Further, the simulation will hopefully not only contribute in the validation of the agent system paradigm, but also investigate some common terms and factors thought to influence and differentiate the approach. The identification and validation of such factors will contribute to the understanding of advantages and drawbacks of the approach, giving evidence to where and how the technology might prove useful. Though many frameworks and applications have been developed over the last decades, very few development processes have been properly documented, increasing the importance factor of this contribution.

Also, the key development factors found will be investigated in terms of relations with common agent and agent system properties. Establishing such relations should contribute to raising the understanding of agent technologies in contrast with other legacy approaches. If possible, a clear differentiation from traditional approaches is desirable to further clarify the possible roles of agent technologies.

1.3 Paper Structure

This chapter will describe the remaining chapters of this paper. Including this introduction, there are eight chapters in total:

- Chapter two presents some background terms and theories, providing a fundamental knowledge base on agent and agent systems. This chapter also briefly explains how these theories might be applied on a patient scheduling system. The practical impact of these theories is presented in the form of a recurring example throughout the chapter.
- Chapter three presents significant prior research on and around these subjects. This includes a general overview, an introduction to existing tools and applications, as well as an investigation of common factors in regards to the development process.
- Chapter four presents the research method with elaborative explanations on how the study was conducted and on which basis research methodologies were chosen. This chapter will also introduce the hypothesis model and give a brief overview of the design methodology used to develop the prototype.
- Chapter five presents the case study. In such, the mechanisms and cooperation structures will be presented in a detailed step-by-step manner, providing relevant diagrams and explanations for each phase.

- Chapter six presents a thorough explanation on the resulting experiences gathered during the development process. Each phase in the design methodology will be evaluated in terms of agent characteristics.
- Chapter seven discusses these results on terms of prior research, and thereby especially the hypothesis model. Each factor of this model is thoroughly evaluated and relationships to agent characteristics are challenged.
- Chapter eight concludes the thesis, elaborating on its main findings and shortcomings from the discussion. Furthermore, this chapter presents limitations of the study as well as possible future work.

2 Background

This chapter will present introductory background material needed in order to present and explain the work of this project. The first two sections will thoroughly define agent and agent systems, explaining their distinction from objects and their properties and abilities. The last section will look at agent and agent systems in the context of patient scheduling, investigating both positive and negative aspects. To provide a practical view of these concepts, all three sections are concluded with a recurring example, where an agent system is implemented to support an online book-club. Note that this example is solely invented and presented by the author and have origins from reality.

2.1 Agents

As its name implies, agent-based systems are based on agents and their ability to interact with each other and its environment. In its simplest form; an agent is an autonomous entity capable of performing actions and interactions based on beliefs and goals. Agent systems can consist of just one such agent or a collection of agents performing different tasks based on individual or common goals.

2.1.1 Definitions

A clear definition of agents and agent systems presents a full study by itself. This relatively young though academically popular technology suffers from poor standardization and lacks in consensus between researchers, definitions ranging from the rather pessimistic just-another-object interpretations, to the rather extreme opposite science-fiction tales (Jennings et al, 1998).

Russell and Norvig (2003) take a very broad view on agents, describing them as only acting entities within an environment:

> *"An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators."*

Although this definition aligns smoothly with most other agent definitions, it is very general in its approach. Some recurring definition problems, like distinguishing agents from objects and specification of dynamic environments, are not accounted for.

The organization AgentLink provides us with a more detailed definition. AgentLink presents itself as the biggest coordination portal within agent-oriented software development and defines agents in its continually updated agent systems roadmap paper:

> *"…a computer system that is capable of flexible autonomous action in dynamic, unpredictable… domains." (Luck et al, 2005)*

This definition recognizes agents as computer systems, underlining their independent nature. This is further elaborated by including their capability of performing autonomous actions in unpredictable environments, distinguishing to greater degree agents from objects and specifying the underlying dynamic environment.

However, according to many researchers, there are still more to it. Agent research have since its birth in the 80s been closely related to research within the field of Artificial Intelligence. In

fact, many of the terms now related to agents originated from concepts within this discipline (Jennings et al, 1998). This leads us to the most optimistic and futuristic definition:

> "*Intelligent agents are software entities that carry out some set of operations on behalf of a user or another program with some degree of independence or autonomy, and in so doing, employ some knowledge or representation of the user's goals or desires." (Knapik and Johnson, 1998)*

This definition not only realizes the potential collaboration with artificial intelligence, but also states that agents have adopted goals and desires, confirming the autonomous and anthropomorphic nature of such entities. By recognizing these possible intelligence aspects of agents, the definitions now cover the most common terms associated with agent characteristics.

2.1.2 Agents and Objects

Another approach in understanding agents is by comparison with earlier attempts at computer science abstractions. As mentioned above, earlier abstractions originated from machine code into more human friendly languages and frameworks. Examples are the early and highly mechanical approach of assembly, the mathematical, lambda calculus based methods of functional programming and the more recent, more human friendly, object abstractions.

Most agent researchers believe that agent technology is in some way the next in line in this computer science evolution. However, there is no clear consensus as to how big a step the agents have taken to distinct them from the now validated and well-used object oriented methods and technologies. Related to the hype factor described earlier, many researchers warn not to get too carried away in this distinction, fearing that agents might suffer the same backlash as that of the artificial intelligence research field in the 80s (e.g. Foner, 1993; Jennings et al, 1998). However, most agree on particularly three distinctions.

1. While objects rather passively perform actions based on input, agents are more *autonomous*, deciding for themselves whether to perform actions or change state through sets of goals and desires (e.g. Jennings and Wooldridge, 2000; Wooldridge and Ciancarini, 2001).
2. An agent is for this reason more *flexible*. By performing tasks based on goals and desires, there are intrinsically several ways those tasks can be done, often taking greater advantage of the potential resources available as a result (e.g. Wooldridge and Ciancarini, 2001; Zambonelli and Omicini, 2004).
3. While objects simply call on other objects requesting explicit tasks, agents are considered more *social* entities. They exhibit formal languages to perform complex conversation and discussions. This enables groups of agents to work on the same goals and allows for effective optimization strategies (e.g. DeLoach et al, 2001; Jennings and Woodridge, 2000).

To summarize these concepts, a comparison model is presented below. The model, originated from Zambonelli and Omicini (2004) illustrates the autonomous, flexible and social characteristics of agents, contrasted with traditional method-call architecture:

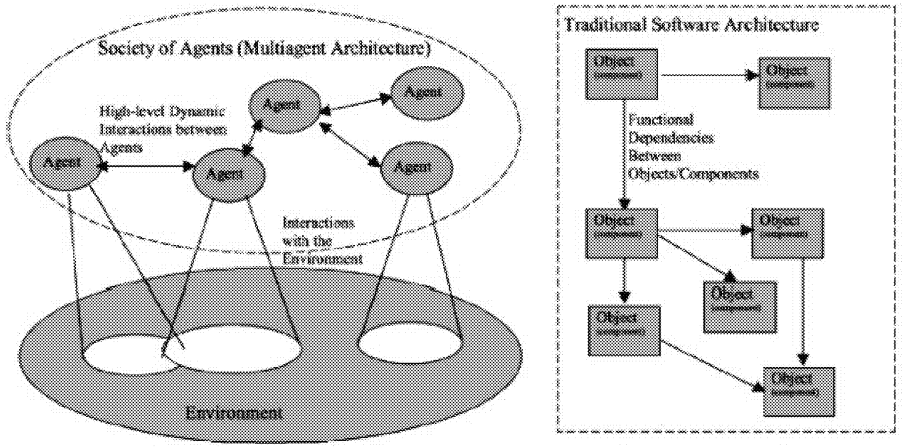


Figure 2.1: Agents and Objects Comparison (Zambonelli and Omicini, 2004)

Jennings et al (1998) argues that the main difference lies in the negotiation abilities of agents. While objects communicate passively, agents interact, socialize and reach agreements. Hence, the abovementioned authors summarize:

> *"Objects do it for free; agents do it for money."*

As a consequence agents not only have control over their own state, but also their own behaviour. For example: Whereas objects can only explicitly define access to public methods, the autonomous and social properties of agents allows for negotiating access terms and conditions to initiate behaviours and services.

2.1.3 Agent Characteristics

The definitions and comparisons above give us a general idea on what agents are and what they can do. This section will elaborate on this by giving a more detailed insight to agent characteristics. In this context, abilities denote agents' behavioural capabilities, while properties encapsulate structural and operational features. As we have seen, agent definitions vary in scope, consequently relating a vast amount of different characteristics to the agent term. The list below is therefore not a complete description of all interpreted characteristics, but encapsulates the most important recurring concepts from literature:

Anthropomorphic
Anthropomorphism denotes the attributing of human characteristics to non-human entities. As the agent abstraction often represent a human actor, adopting mental properties and complex behaviours, this property is thought to have more relevance and usefulness when designing agents than other computer science abstractions. Abilities like personalizability, learning and reasoning are often mentioned in this context (e.g. Foner, 1993; Wooldridge and Jennings, 1995).

Autonomous
Autonomy is a recurring and encapsulating term found in most agent related papers. In essence, the term refers to the property of independence, closely related to the ability of having grater control over own state and actions (e.g. Jennings et al, 1998, Woodridge and Ciancarini, 2001).

Pro-active
Pro-activeness denotes agents' ability of taking actions on their own account. An agent does not only react to its environment, but often exhibit behaviours to taking initiative, performing actions to achieve goals (e.g. Woodridge and Ciancarini, 2001; Zambonelli and Omicini, 2004).

Situated
Situatedness encapsulates the agents' close relationship to its environment. Agents optimally have awareness capabilities, supervising their domain through sensors and performing reactive actions through effectors accordingly. When all agents in a given environment inhibit such properties, they are in this context dynamically related to each other, and can to a greater degree work towards common goals (e.g. Jennings and Wooldridge, 2000; Weyns et al, 2004).

Social
This property denotes agents' ability to communicate and reach mutual decisions through conversation and negotiation. These behaviours are often encapsulates by the term discourse as discussion often is the foundation for making decisions within an agent environment (e.g. Foner, 1993; Wooldridge and Ciancarini, 2001).

2.1.4 Basic Structure and Functionality

This section will use the characteristics described above to give a simple overview of an agent structure and functionality. Note that there are many different approaches to designing agents, thus this suggestion must be regarded as a general example. The model below shows how an agent uses its properties, abilities and the environment to achieve its goals:

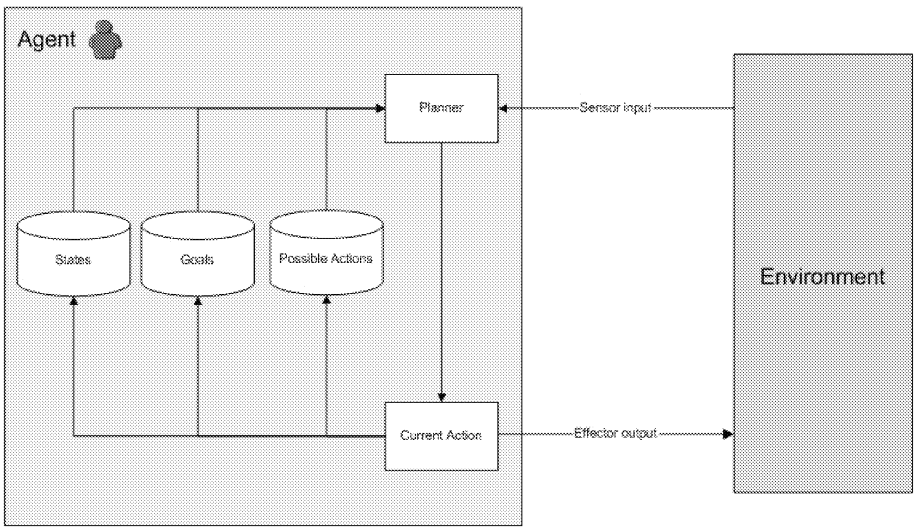


Figure 2.2: General Agent Structure

The agent planner can be considered the processing unit of the agent. As shown in the figure above, decisions are made by the planner on the basis of its current state, its set of goals, its possible actions and the surrounding environment or domain. More specifically, an action is scheduled with the intention of achieving a goal by performing one of the possible actions, taking both the current state and the environment into consideration.

Further, the action is performed through effectors on the environment and the list of goals, possible actions and state are updated accordingly - possibly leaving the agent with a completely new agenda as a result. Note that the model does not account for additional agents in the domain as agent systems will be investigated in section 2.2.

2.1.5 Example Part One: Book-Club Agents

To conceptualize the model above, this section will present the first part of a recurring example. The example will introduce an online book-club to see how agents can help solve common tasks related to data transfer and interaction. Note that this chapter is considered introductory, and will therefore not provide any technical details or thorough concept explanations. Refer to the case study in chapter five for a more detailed approach.

Briefly, the online book-club in question must provide some services to its customers. A customer must have a personal member area where they can search for, reserve and buy books. Additionally, they should receive personalized book information and offers through stated preferences. A natural start when designing agent systems is to define the agents that are needed to complete the task. Considering the requirements above, our system will consist of only two agent types: Customer and Book.

Customer Agent
The customer agents will represent each member of the book-club and must therefore be able to store information about its user, provide this information to book agents and receive new personal information to update its preferences. These operations can be materialized through a set of goals with complimentary possible actions. For example, the agents' goals can be set to:

1. Store and provide updated information about the customer.
2. Provide search service for the user.
3. Display book offers and provide buy options for the user.

The two goals capture the service oriented nature of the agent. While the first goal ensures an updated profile servicing its user, the second provides search parameters, servicing the book agents. We will see how this service orientation is exploited in section 2.2.4. To continue, the possible actions to achieve these goals can be defined as follows:

1. Add and store information about a new customer.
2. Receive and update customer preferences.
3. Communicate with book agent(s) to provide customer information.
4. Interact with book agent(s) to find relevant book titles.
5. Communicate purchase details.

As we can see, the possible actions should be sufficient to achieve the abovementioned goals. The agent is summarized in our familiar model below, showing that the planner can use the suggested goals and possible actions to schedule appropriate tasks:

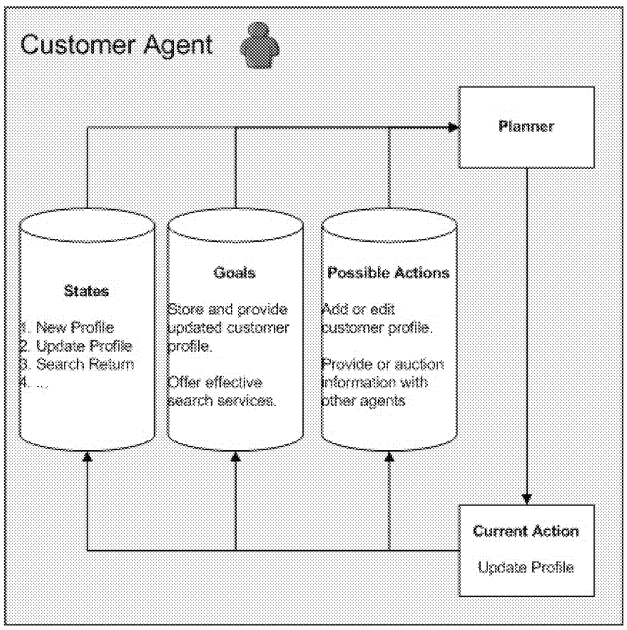


Figure 2.3: Customer Agent Example

Book Agent

The book agent will represent each book in the book-club and must, as such, store information about its book in addition to actively search for customers which might be interested in reading it. The search for interested buyers is based on the book information and the customer preferences mentioned above. Keeping this in mind, the goals can be defined as follows:

1. Store and provide updated information about a book.
2. Provide relevant book information to potential buying customer profiles.

The two goals are similar to the customer agent, but the book agent is not passive and responsive, but rather pro-active in its search for matching profiles. Considering these goals, the following possible actions can be devised:

1. Add and store information about a new book.
2. Actively interact with customer agents to find matching preferences.
3. Provide book details and offers to relevant customer profiles.
4. Interact with both customer agents and other book agent to find relevant search results.
5. Handle customer agent purchase order.

Again, the agent is summarized in the model below. In addition to actively searching for buyers, note also the similarities with the customer agent, especially in providing book information services to customer agents.

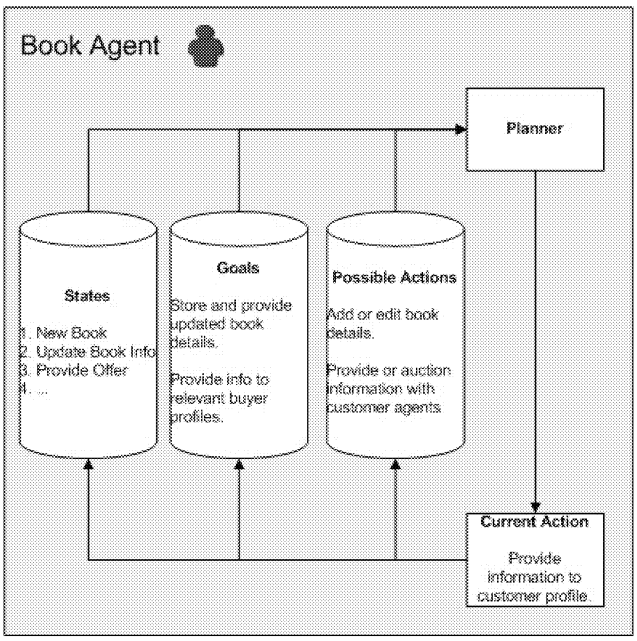


Figure 2.4: Book Agent Example

To summarize, we look back at the presented models with the list of agent properties and abilities in mind. Each time a new customer joins the online book-club, a new customer agent is made. Analogies can be drawn to its represented user, seeing the agent as the members' protégé in the system, thus giving evidence to the **anthropomorphism** property of agents.

Further, we have seen that the customer agent functions as a service provider, both to the customer and to the book agents. The customer can update his/her profile, including metadata like book genre preference, favourite authors, language and so on. Reacting to this change in environment, the book agent actively seek out the new information, eventually matching it with its own book details, thus giving rise to the ability of **pro-activeness** and **situatedness**. When the book agent finds a relevant customer profile, offers and book details are displayed, personalizing the private member area.

Further, if the customer wishes to perform a search, the resulting titles will be based on auction between relevant book agents. Based on the search criteria provided by the customer agent, the book agents argue their relevance, before returning an agreed list of titles. This process underlines the **social** ability and discourse property of agents.

Finally, despite the close communication and cooperation between the two agent types, it is evident that they are distinct and independent entities. For example, a book-agent pro-actively searches for potential customer agents to achieve its goals, denoting distinct entities exemplifying the **autonomous** nature of agents.

2.2 Agent Systems

Agent systems denote a society of agents, collectively functioning as a software system. The agents cooperate within such a system to achieve individual or common goals, often through complicated interrelated tasks and interaction. Some examples of well suited domains are online ticket ordering services, large manufacturing pipeline software or hospital information systems (Luck et al, 2005). As agents already have been defined above, this section will elaborate on the collective aspects like system structure, interaction and cooperation.

2.2.1 Definitions

As with agents, there is a wide selection of definitions available to capture the essence of agent systems. Vlassis (2003) takes a general, yet valid view, defining agent systems simply to be an interacting group of agents:

> *"…a group of agents that can potentially interact with each other…"*

This definition correctly captures the collective effort amongst agents, but is as we will see somewhat defensive in its approach. Note also, that this definition does not view interaction as a necessary property, but rather as a beneficial quality - hence, agent systems might consist of only one agent. Zambonelli et al (2003) further elaborate on the above definition by including how these interactions are done and for what purpose:

> *"In multi-agent systems, applications are designed and developed in terms of autonomous software entities (agents) that can flexibly achieve their objectives by interacting with one another in terms of high-level protocols and languages."*

By detailing some agent properties, this definition encapsulates the flexible nature of agent systems. It also states that the motivation for collaboration is a set of objectives, or as can be interpreted to the more common agent term; goals.

Though most of the common features of multi-agent systems are captured in the definitions above, there is still one very important point to make, namely the effectiveness of collaboration. Bernon et al (2005) captures this benefit in their definition:

> *"Most of the authors agree on viewing a MAS as a system composed of agents that communicate and collaborate to achieve specific personal or collective tasks."*

By using the potential abilities of agents in an interactive environment, tasks and goals can be reached in a collaborative manner, where specialized agents can be used to their fullest advantage, exploiting all competence available in the system.

2.2.2 Agent System Characteristics

To define collaborative operations including interaction procedures, decision making processes and data sharing, an agent information system must be built on the basis of a controlling structure, or in agent-terms; an underlying architecture. In addition to defining standards within data structures and communication language, this surrounding framework should also assist in coordinating mental states like intentions and goals, both between to the environment and amongst the participating agents. As we will see in the next chapter, agent architectures are readily available through implementation tools and communication protocols.

Having established the architecture, individual agents must be defined, organized and implemented to the system. As explained in the previous section, an agent is often an abstraction of some real-life entity. Deciding which abstractions to associate to which group of agents is a challenge developers must face early on in such software development projects. During this process, there are a number of properties associated with agent systems, which all must be considered:

Decentralization
Accounting for their autonomous nature, agent system architecture does not denote any form of centralized control, but rather the underlying foundation on which certain abilities may be materialized. In agent-oriented systems there is no centralized process to control and distribute agent operations. Instead, the agents are interdependent processes, communicating and solving tasks directly with each other or through the environment (Vlassis, 2003).

An example of this is two general structure models from some traditional software systems. The client-server model (Figure 4) is a typical centralized structure, where a central server provides services to its clients. An example of such a structure is web pages, where browsers are clients and web-servers provide the services. The other structure is the peer-to-peer model (Figure 5). Here, there is no central server, as all communication and processing is done directly between and amongst the peers. Example of the peer-to-peer structure is popular file-sharing programs like Napster or BitTorrent.

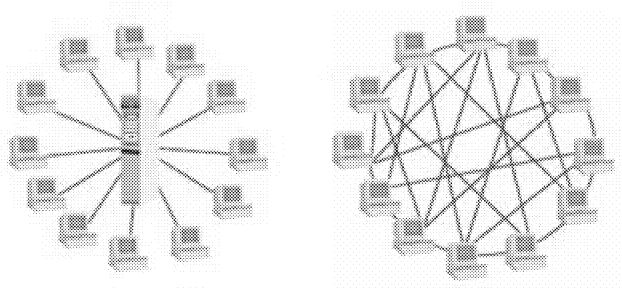


Figure 2.5: Client-Server and Peer-to-Peer Structres (Bellefemine et al, 2003)

Deterministic or stochastic states
Agents must in many situations evaluate future action outcomes in order to schedule optimal behaviour. These future predictions are affected by a number of factors in the environment, and vary accordingly in both complexity and operation. While a deterministic environment is fully predictable and holds no random properties, a stochastic environment is probabilistic, containing unpredictable factors which may affect action outcomes (Vlassis, 2003). As such, these terms are closely related to the observability problem and discrete data presented elsewhere in this section.

Discrete or continuous data
These terms denotes whether the agent system environment has a finite set of states or not. For example, when simulating a game like chess there will be a finite set of states. Computing in this context is based on a set of allowed moves and actions, hence a discrete number of possibilities. A contrasting example could be simulating coordinates in space, presenting a seemingly infinite number of possibilities, hence making the system continuous. Such a system is often harder to engineer and maintain as results within an infinite scope are inherently harder to predict (Weyns et al, 2004).

Dynamic or static environments
One of the widely referenced properties of agent systems is exploitation of the flexible nature of agents to create a dynamic system. While static systems does not account for components entering, leaving or changing at run-time, dynamic systems are engineered for handling such events (Woodridge and Ciancarini, 2001). Consequently, dynamic structures are inherently harder to implement and govern.

Heterogeneous or homogeneous agents
As agents in agent systems are decentralized and autonomous they provide greater opportunities for variation in structure and design. While cooperating to reach the same goals and intentions, agents can inhibit widely different capabilities and data structures. Despite being distinctly different entities they can still cooperate through the underlying environment. This in mind, a system comprising such a variety of agents is called a heterogeneous system, whereas a system containing agents of similar design and data structured can be labeled homogeneous (Vlassis, 2003).

Mobility
Because of the autonomous nature of agents and the dynamic property of agent systems, agents can more easily transfer to other systems with the same characteristics. As agents often represent a live entity, this is often a desired process, with the preservation of for example personal data as an example (Sycara, 1998).

Partial observability
Known as the partial observability problem, this term encapsulates agents' perceptions in regards to state and data of the environment. Due to inadequate sensors, equal or discrete entities or transparency of data, an agent may not be able to compute or observe parts of its domain. The partial observability problem is a direct result of three common and important properties of data in agent systems (Vlassis, 2003):

1. *Data is spatial - it is stored at different locations.*
2. *Data is temporal - it is distributed at different times.*
3. *Data is semantic – interpretation varies.*

Such properties obviously require the designer to account for undisclosed data and thus unpredictable behaviour, making the design process increasingly difficult.

2.2.3 Interaction and Cooperation

The social abilities of agents constitute one of the major differences between agent systems and legacy approaches. By complex interaction agents can negotiate to support cooperation in performing appropriate tasks and reaching shared goals.

This cooperation must be based on set standards and well founded optimalization structures to ensure that the agents can understand each others semantics and effectively make optimal decisions. Also, to make full use of all resources available, it is essential that the agents have full understanding of the environment as well as other agents' capabilities. When these parameters are established, the agents can make optimal decisions to ensure solutions based on common goals.

Communication Protocols
An explicit, yet essential criterion for such behavior is that agents understand each other, both in language and semantics. There have been numerous attempts at standardizing agent language, prominent examples being FIPA-ACL (FIPA, 2002) and KQML (Finin et al, 1994). These specifications allow for communication between agents based on regular protocol standards, tailored for agent interaction. Put simply, the protocols allow agents to communicate, regardless of location, architecture or platform.

Negotiation
Negotiation between agents is often used as means of optimal decision making. This term denotes the ability of an agent to perform an optimal individual action, which contributes to the goals of the system as a whole. This often involves consideration of numerous factors, including world state, personal goals, other agents' agenda and task priorities (Vlassis, 2003).

2.2.4 Example Part Two: Book-Club System

To continue our example from part one, we return to our online book-club. We assume our club has been expanded with several new customer and book agents to represent an agent system. Following the agent definitions from the previous example, the agents can be organized in the following way:

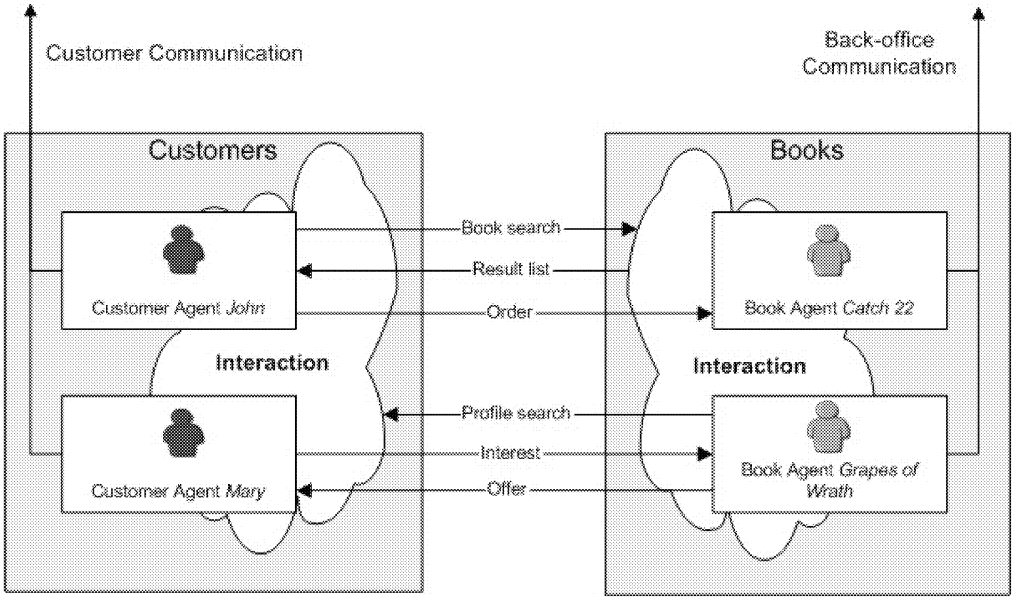


Figure 2.6: Book Club Agent System Example

Looking at figure six, there are two general scenarios. In scenario one, customer agent *John* search for relevant book titles based on some keywords. The book agents respond by collectively deciding on a relevance list which is in turn returned directly to agent *John*. Based on this information, *John* issues a purchase order directly to the book agent *Grapes of wrath* which handles the order accordingly.

In scenario two, book agent *Catch 22* actively search for potential buyers based on their book preferences. After interaction between the available customer agents, agent *Mary* issues a message of interest directly to the book agent. *Catch 22* then sends its book information with corresponding offers and prices back to *Mary* which may choose to issue a purchase order.

To conceptualize our definitions of agent system characteristics presented above we can relate them to our example: The system is solely made of autonomous agents and is thereby **decentralized**. This is not so evident looking at the figure where both customer and book information are provided to a centralized yellow page unit, however, this kind of service are provided by a single specialized agent, thus adding another autonomous component to the system. Further, the example is clearly **discrete** as there are a finite number of known variable

types in the system and they all range within finite structures. Examples are price, book title, customer name or book name.

When entering data, whether being a customer or member of staff, there is always a chance of syntax or semantic errors in for example book titles or customer names. Also, there may be technical difficulties disclosing certain parts of the book or customer agents' information fields. Both issues are fairly common in any computer system and can in our case lead to problems in interpretation, thus according for the **partial observability problem**. Further, all actions within this system is **deterministic**. In our example we know what will happen if a customer buys a book for a certain price or if a new book title is added to the system. None of these events could be related to computational probabilities or uncertainties.

Our agents in the online book-club are fairly **homogeneous**, though there is no clear line to confirm this. While there are two distinct types of agents (i.e. **heterogeneous**) there are but small variations when comparing agents of the same type. In the same manner, the agent system above can be considered both **dynamic** and **static**. While the system allows for new agents to represent new books and customer, it is not dynamic enough to accept new types of agents or variations of the existing ones.

These last properties allow for the final agent system characteristic. Since the system can accept new agents, representing customers or books, such entities can be moved across similar system structures, thus accounting for the **mobility** behaviour.

2.3 Patient Scheduling Using Agents

Having defined the concepts of agents and agent systems, we need to find an appropriate environment for our case study. The prototype developed during the course of this thesis will simulate the activities at a medical institution, more specifically focusing on the scheduling of patients undergoing a number of examinations and treatment methods. This section will investigate the basic characteristics of such systems and suggest how an agent based solution might be devised.

2.3.1 Hospital Information Systems Characteristics

First, we will introduce some typical characteristics of hospital information systems in general. This is both to justify the choice for the case study and to identify the focus areas and typical challenges involved the engineering of such systems.

In his 1984 paper "*Health Information Systems – Past, Present and Future*", late Peter Reichertz (1984) presented seven typical characteristics for a modern hospital information system. The characteristics were reviewed and validated in Haux (2005) paper of the same title, thus providing a solid foundation for investigating such characteristics. Below we have presented a selected four of these properties and related them to the agent and agent system characteristics discussed throughout this chapter.

Towards computer-based information processing tools
There has been a great shift from paper-based to computer-based data processing within hospital systems over the last decades. This shift is thought to continue and agent orientation being a new paradigm suitable for handling complex systems can be considered in this context:

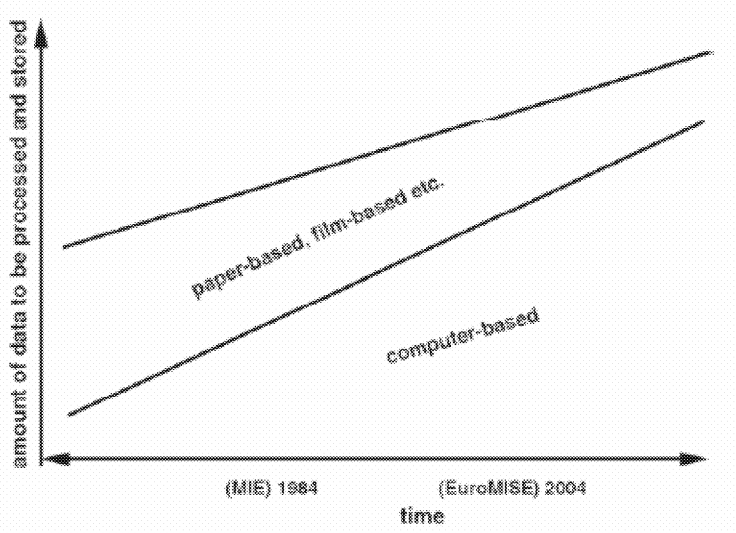


Figure 2.7: Towards Computer-Based Hospital Systems (Haux, 2005)

From local to global information system architectures

Most current health care information systems are in some sense regional, while the sharing of information and expertise over several institutions, and preferably on a global level, is a natural goal of the future. Autonomous and flexible in nature, the agent technology can be seen as a suitable candidate to handle this challenge:

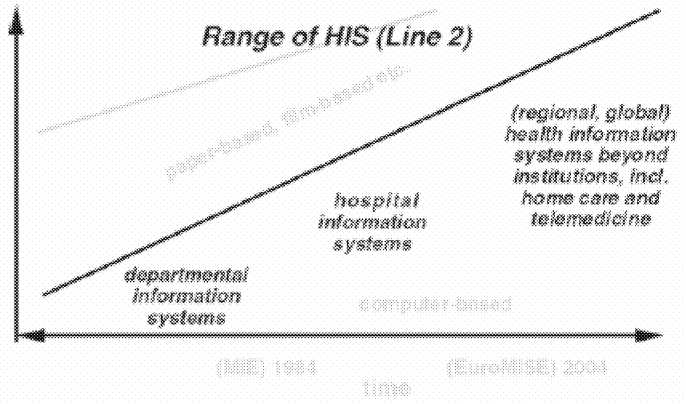


Figure 2.8: Local to Global Architecture (Haux, 2005)

From health care professionals to patients and consumers

There is an ongoing trend in making heath care systems more public, opening up for input and information sharing also from and between patients and citizens. Agent systems often represent stakeholders through the agent abstraction, allowing for personalized and tailored components, highly suitable for such tasks:

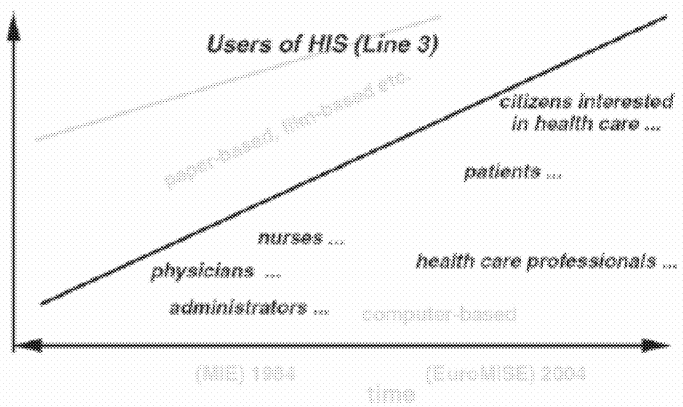


Figure 2.9: Patient Centered Systems (Haux, 2005)

Inclusion of new technologies
Among several, the most apparent change in technologies at this stage of the evolution is the introduction of ubiquitous mobile devices. These devices are integrated in the environment, for example in clothes, computerized watches etc., and is often used to monitor the health condition of patients. By drawing on the dynamic and mobile abilities of agent systems, ubiquitous devices can communicate seamlessly, making agent technologies highly relevant in this context:

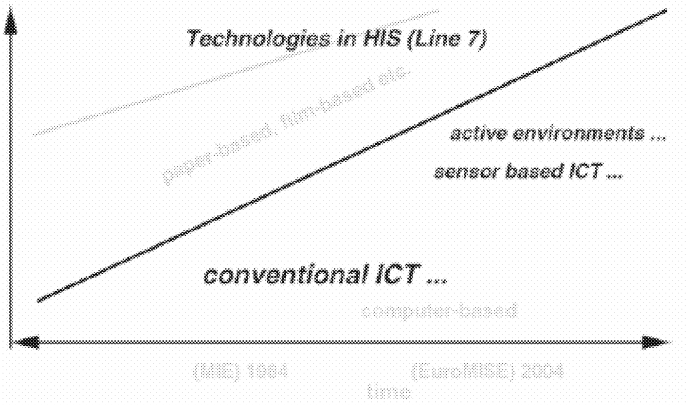


Figure 2.10: New Techologies Emerging (Haux, 2005)

2.3.2 Patient Scheduling Systems Characteristics

In accordance with the evolution described above, medical decision support systems, and especially patient scheduling systems, have become an increasingly important factor in many hospitals and medical institutions (Manansang and Helm, 1996). This view is supported by the renowned marketing consultancy company Frost & Sullivan whom in a study in 2004 identified the need for innovative patient-scheduling systems as a reaction to the privatizing, and thus increasingly competitive tendency of health care institutions in the U.S.

As patient scheduling inherently deals with a distribution problem (Decker and Li, 1998), we can look at some general characteristics for such systems:

- Patient scheduling systems have one primary goal: Treating as many patients as possible in the shortest possible time (e.g. Bartelt et al, 2002; Decker and Li, 1998).
- Dealing with examination and treatment processes for patients involves a high degree of uncertainty in regards of time spans and resulting diagnosis, thus patient scheduling systems have been deemed complex (Bartelt et al, 2002).
- Modern patient scheduling system design focuses on patients rather than specific tasks or resources (Guo et al, 2004).

We can see that patient scheduling systems exhibits many of the same characteristics as we identified in our introduction to agents and agent systems. Characteristics like goal-oriented design, and high complexity and abstraction levels are well-founded identifiers in agent-oriented literature. This last section will merge these theories, exploring how a patient scheduling system might be constructed from an agent-oriented perspective.

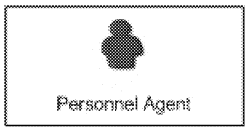
2.3.3 Example Part Three: Agent Based Patient Scheduling

To conceptualize the ideas presented above, this section will continue our ongoing example by presenting a simple agent system designed to schedule patients. As the general structures and communication patterns for such systems is described in part two, this part will only show how to tailor such a system to handle the scheduling tasks described above.

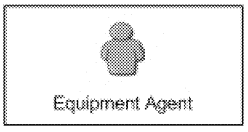
To start off, we need again to define proper agents for our system. As mentioned above, modern patient scheduling systems focus on the patient rather than functions or tasks. Hence, our first agent is the patient:

Goals: *Get examined/treated as fast as possible*
Possible actions: *Apply for appointment*

But there are of course more stakeholders. As our patients must be examined and treated, we also need to include agents for doctors/nurses/surgeons etc. and machines/treatment rooms/medications etc. For simplicity we generalize the terms to personnel and equipment:



Goals: *Treat/examine patients as fast as possible*
Possible actions: *Grant/deny appointment, examine/treat patients.*



Goals: *Support treat/examination process effectively*
Possible actions: *Grant/deny participation, support treat/examine process of patient.*

In the same manner as in our online book-club example, the agents involved use their available actions to achieve their goals. Our patient agents use their possible action "*Apply for appointment*" to try to get personnel and equipment for their examination or treatment. It is then up to the personnel and equipment agents to accept or deny this request based on which patient agents that needs treatment the most. Such optimization calculations often involve several variables – for example, in this case; availability of personnel and equipment, the condition of the patient and the state of other patients.

After the scheduling calculations are done, the personnel and equipment agents use their possible actions "*Grant/deny appointment/participation*" and "*Examine/treat patients*" to notify and process the relevant patient agents accordingly. When the scheduling is finished, the patient agents are notified about their place in the queue, and can apply for examination/treatment at a later stage if their application for a time slot was denied. The following model shows how the agents might cooperate during scheduling:

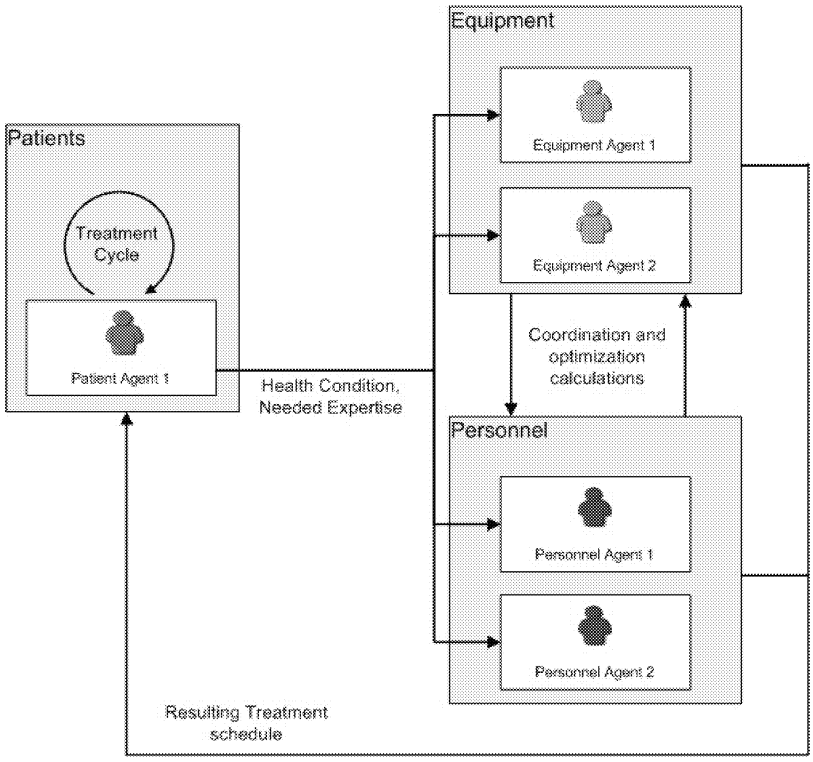


Figure 2.11: Patient Scheduling Cooperation Example

A prototype for a scheduling system was designed as a case study for this thesis. These theories are therefore reintroduced and thoroughly explained throughout chapter five.

3 Significant Prior Research

This chapter will present prior research within the agent and agent system field. The first section will present a brief history as well as available frameworks, tools and techniques. We will also present some commercial and industrial applications to see how these theories have been put to work. The second section will focus on the agent oriented system engineering process itself, discussing some advantages and challenges, as well as looking at some examples from industry.

3.1 Agents and Agent Systems

This section will introduce available tools, frameworks and applications in a historical context. The agent paradigm being relatively new, this chronological approach clarifies the challenges offered by its immaturity as well as presenting a rough state-of-the-art. After a quick historic glance, each tool and framework genre will be investigated individually before a presentation of some commercial application examples.

3.1.1 Historical Evolution and Today's Main Challenges

The term agent can be traced back to the *Actor Model* first presented by its authors Carl Hewitt, Peter Bishop and Richard Steiger in 1973 (Hewitt et al., 1973). The paper presented a mathematical solution on concurrent computation between several autonomous entities. Hewitt and his colleagues describe the entities to be "*computational agents who have a mail address and a behavior*". Thus the first primitive agent concepts were defined.

Jennings et al (1998), recognize a wide array of disciplines contributing to the early definitions of agents and agent systems. They define artificial intelligence to be the main driving force in its early attempts at creating intelligent entities acting in an environment. As presented in chapter two, this was to be a common definition of an agent. These attempts however, came as late as the early 80s, due to the artificial intelligence communities focus on rather specific aspects of intelligence (i.e. learning, vision, understanding etc.).

Through his paper in the mid 80s (Brooks, 1986), robotics specialist Rodney Allan Brooks presented a new way to reason about planning and learning in artificial intelligence. Instead of the traditional symbolic reasoning, Brooks proposed interaction as the main challenge and focus for intelligent entities. His proposed frameworks (e.g. Brooks, 1990) have many similarities to today's definitions of agent systems. The entities reasoned on the basis of other entities and the environment, and the entities were somewhat autonomous in nature. However, the theories still suffered from having no tailored engineering methods, and their applicability was highly questionable.

Another research area contributing to early agent research was human-computer interaction; the idea of simulating human qualities, not only on the intelligence level but also on behavior and representation. In the late 80s the Belief-Desire-Intention (*BDI*) model was proposed (Bratman et al, 1988). The model (fig. 3.1) represented a novel approach of giving human properties to digital agents. Through available information about the environment (beliefs), the agents are given a set of certain possible actions (desires) which are activated based on agent goals (intentions). Many more recent tools and techniques are based on this framework and propose modeling techniques and implementation tools using these concepts (e.g. Shoham, 1993; Zambonelli et al, 2003; Kinny et al, 1996). We will look at some of these approaches more closely in section 3.1.2.

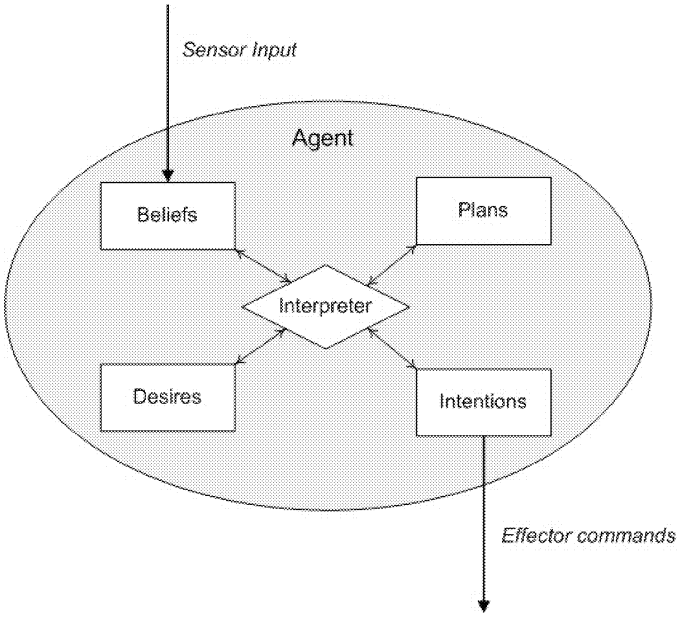


Figure 3.1: The Belief-Desire-Intention Model (Wooldridge and Parsons, 2003)

With the development of the agent notion, there were also ongoing attempts at solving the communication and cooperation challenge. Knowing the potential of single agents, the multi-agent system research looked for optimal ways of cooperation and task coordination (Jennings et al, 1998). One of the most prominent attempts was the *ContractNet* protocol (Smith, 1980) which introduced negotiation as a means of reaching optimal cooperation. The framework allows agents to divide up given tasks before delegating them on auction. Other agents' bids are based on how well suited they are to solve the auctioneered sub-tasks. The initial protocol is however limited in communication options, concentrating on delegation of tasks rather than more robust discourse abilities. It also lacks flexibility on conflict solving, due to information restrictiveness in the domain environment (Jennings et al, 1998). *ContractNet* has however been extended and adapted to many models and frameworks. The terms negotiation, bids and auctions have later been an integral part of agents and agent systems research, featuring heavily in many methods and applications (e.g. Van Dyke Parunak, 1987; Jennings, 1993; Jennings et al, 2001; FIPA, 2002).

From these early concepts, agent systems are now recognized as a software paradigm. Although somewhat lacking in validating its usefulness and applicability in industrial and commercial applications, this new paradigm is by many researchers deemed to cause big changes in the software industry (e.g. Bernon et al, 2005; Zambonelli and Omicini, 2004). The hype has however been met with some skepticism, mainly due to the before-mentioned lack of verification (e.g. Jennings et al, 1998; Wooldridge and Jennings, 1998).

Based on this, Luck et al (2005) argue that agents and agent systems are still far from reaching its deployment period. Based on Perez' life of technologies model (fig 3.1), the authors argue that agent technologies are still at the irruption stage. In this phase, a technology has not yet reached the maturity which is required for industrial, commercial or even serious experimental application. The phase is also often related to hype and romanticism. Ibid. (2005) argues that this immaturity in agent and agent systems is mainly due to the lack of standardization, making industrial deployment attempts somewhat of a gamble as of yet. Thus, the main challenge for the agent and agent systems research communities today is

verification and validation of existing models, frameworks and methods, and finally, the rigorous process of establishing standards.

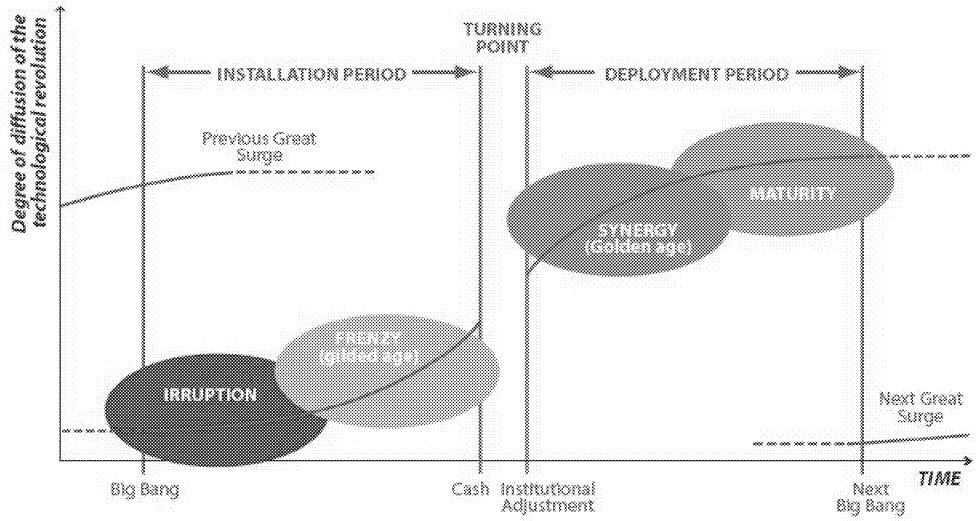


Figure 3.2: The life and phases of a technology (Perez, 2002)

In his famous book, *Diffusion of Innovations* (Rogers, 1995), late Everett Rogers introduced theories on the lifetime of innovative trends. A major contribution in Rogers' work is the now well known notion of a *tipping point*. Rogers suggests that that after a 10-25% adoption of a technology, adoption will rapidly increase, confirming the S-shaped adoption rate proposed by Perez. This in mind, we see another confirmation of the immaturity of agent and agent system technologies, as such adoption numbers are still far from reality (e.g. Luck et al, 2004).

3.1.2 Tools and Techniques

There have been many contributing tools and techniques to aid the development process of agents and agent systems. Though these contributions have added valuable input for many aspects of the agent research community, it can be argued that the great variety of approaches poses the technology with a twofold challenge. While the wide array of techniques presents a solid foundation to solve problems, the many ontologies, methods and frameworks can be seen to cause even further chaos to the agent verification process (Jennings et al, 1998). Nevertheless, this section will present some of the approaches presented over the last decades, highlighting both differences and similarities.

Methodologies

Amongst the most quoted tools and a key factor in applying agent systems in industry are the methodologies, aimed to help the developer through the whole engineering process (Iglesias et al, 1998). Using tools and techniques for modeling and implementation, either adopted from others or defined from scratch, a methodology should be a high-level, comprehensive effort to assist a project from start to finish. Looking at the most significant proposals for agent-oriented methodologies helps broaden the understanding of and the context in which the various tools are applied. In addition we will outline the strengths and weaknesses of the three presented methodologies, generalizing some key issues in agent systems development.

AAII (Australian Artificial Intelligence Institute) presented in 1996 is a methodology based on the before-mentioned *BDI* (Belief-Desire-Intention) concept (Kinny et al, 1996). The methodology provided a foundation for many of the methods to come. Building on the

established formal specification framework *DESIRE* - DEsign and Specification of Interacting REasoning (Brazier et al, 1995), the methodology consists of two viewpoints; an internal and an external. The internal viewpoint defines classes of agents through various models, while the external viewpoint specifies the interaction and cooperation between these classes.

Presented relatively early in the agent systems history, the methodology have some obvious constraints (Kumar, 2002). First, the external viewpoint does not provide sufficient details on cooperation and coordination issues, deeming the methodology somewhat incomplete. Second, being founded on existing object oriented techniques, communication is purely based on hierarchal inheritance and method calls, thus not exploiting the negotiating and discourse abilities of agent systems.

Gaia (Generic Architecture for Information Availability) (Zambonelli et al, 2003) was first presented in 2000 and offered a more detailed and complete approach. Centered on the concept of roles, the methodology defines models for individual agents as well as interaction and communication in a multi-agent system. These characteristics are quite similar to the *AAII* methodology mentioned above, but there are some important differences (Kumar, 2002). Gaia represented a step forward in being designed around and about agent properties and concepts. This is a contrast to previous attempts, mostly extending object oriented or knowledge engineering methods (Iglesias et al, 1998). This new level of focus also ensured that the interaction and cooperation abilities of agents represent a central part of the methodology. While being in this sense more complete than the *AAII* methodology, *Gaia* still lacks sufficient guiding on the implementation phase, both in respect to communication and agent specification. While being designed around abstract agent concepts can help the developer to more easily map agent classes to real-life entities in the analysis and design phase, the concepts can prove too abstract to handle during implementation. This gap is a reoccurring and well-documented challenge within the agent systems research field (Sycara, 1998).

Lastly, *Message/UML* (Multiagent Systems Engineering) by Caire et al (2001) borrows features from several existing agent methods and is, as a result of this, one of the most complete methodologies around (Bernon et al, 2005). The process is described from five different viewpoints, ranging from internal agent buildup to interaction among agent organizations. All phases include the use of an extended version of *UML*; *AUML* – Agent Unified Modeling Language (Odell et al, 2001), offering a balance between the advantageous agent abstraction design and easy implementation. But even this methodology has some limitations (Goméz-Sanz and Pavon, 2002). *Message/UML* does not model the interaction with the environment sufficiently. As this interaction is an integral part of an agent system (labeled *situatedness* in chapter two), the methodology has shortcomings in encapsulating the multi-agent system concept as a whole. Further, all the methodologies lacks in providing sufficient technical tools to assist developers in design, implementation and testing.

The above three methodologies are just a few of many approaches presented over the last decade. Most of them have contributed in some way to the AOSE research, but have either problems like the ones described above or is promising, but too novel to yet be confirmed as a functional methodology. Examples include *Tropos* (Giunchigilia et al, 2001), *MaSE* (DeLoach et al, 2001), *INGENIAS* (Gómez-Sanz and Fuentes, 2002) and *Prometheus* (Padgham and Winikoff, 2002).

Communication and Language
As mentioned, a good agent system is characterized as social and should inhibit discourse abilities. This means that communication protocols and language semantics within agent technologies are somewhat more ambitious than that of earlier software engineering paradigms, specifying to a higher degree the meaning and intention of communicative acts. Labrou et al (1999) puts this consequence in relation to the autonomous and anthropomorphic nature of agent systems. Human language being a superior communication tool, it is only natural to try to simulate its complexity when designing within the agent abstraction.

There are two communication protocols (commonly labeled ACL – Agent Communication Language) which have been particularly influential in the agent system research community; the *FIPA* (*Foundation for Intelligent Physical Agents*) specification (FIPA, 2002) and the *KQML* (Knowledge Query and Manipulation Language) language (Finin et al, 1994).

Developed in the early 90s, *KQML* was the first serious attempt at specifying a unifying language protocol for software agents. *KQML* was based on the already established linguistic and philosophical notion of speech-acts, where messages can be semantically interpreted in terms of the senders intended actions (ibid.). Though *KQML* has been extended in many varieties and successfully tested in different settings by a large research community, it was criticized early on for lacking backing tools for API specifications, debuggers, interaction tracers and the likes (Mayfield et al, 1996; Labrou et al, 1999).

The other specification mentioned, *FIPA* ACL is a product of a standardization effort initiated by the well established IEEE association and is in essence very similar to *KQML*. The specification did however present an extended framework for interaction semantics as it could build on some concepts already implemented in the *KQML* specification. Also, both approaches are now gaining considerable amount of attention from development and implantation tools, taking advantage of the potential standardization process (Chaib-Draa and Dignum, 2002).

Scheduling and Planning
With sophisticated communication, agents can interact to cooperatively achieve global tasks and goals. But this coordination needs more than sufficient shared semantics. It also requires planning and scheduling techniques to govern the order and partition of tasks. With desired characteristics like autonomy, pro-activeness and flexibility these techniques pose quite a challenge in the context of agent systems. Roughly, there are two general frameworks developed over the last decades to deal with these challenges; namely the partial global planning (PGP) algorithms and the joint intentions framework.

First presented in the late 80s (Durfee and Lesser, 1991), PGP was an early attempt at planning in a distributed dynamic environment. By sharing and communicating intentions globally, the framework allowed agents to make optimal decisions locally. Decker and Lesser (1993) further developed this framework by presenting a specified algorithm for agents and agent systems. The framework pinpointed the partial observability problem (described in chapter 2 in this paper) by basing solutions on the information available, discarding hidden agents or disclosed data. PGP is very much active today, mainly through the *TAEMS* framework (Horling et al, 2005). *TAEMS* is a further development from the earlier frameworks, allowing for better controlled scheduling of tasks and easier transfer of semantics (Jennings et al, 1998). The TAEMS framework is used in many applications and, among other implementation options, have its own Java API.

Joint intentions present another approach in coordinating and planning node actions. Instead of passively collecting information to decide on optimal actions, the joint intentions frameworks is mainly focused on reaching common goals through agreement (ibid.). The model is that of a team's intention rather than the individual agents' goals. As the main focus of communication in these frameworks is reaching agreements, a natural consequence is *negotiation* which we look at more closely in the next section.

The notion of joint intentions has appeared in many models and frameworks. In chronological appearance, the most notable have been *Commitments and Conventions* (Jennings, 1993), *SharedPlans* (Grosz and Kraus, 1995) and *Shell TEAM* (Tambe, 1997)

Negotiation
As presented above, in some situations, agents need to negotiate to reach agreements. In fact, the desirable flexible and social behavior of agents can not be effectively exploited without negotiating task delegation and scheduling. Naturally, there have been various attempts at modeling effective negotiating techniques.

Almost all negotiation in agent systems is based on some notion of auction (e.g. Luck et al, 2004). An auction involves some artifact, in this context often a task, which is being auctioneered at the highest price. Agent bids are in this situation often a numerical representation of its suitability of handling the task in question. As mentioned above, a framework for such an auction negotiation was presented in the *ContractNet* protocol as early as 1980 (Smith, 1980).

The bids used in auctions are often based on game theory mechanics and utility functions. Basically, game theory is the study of decisions in environments where several players interact (Vlassis, 2003). The main principles of game theory are recognized in a wide array of methods and frameworks. As the early frameworks originally branched from mathematics and economy research, the agent research communities have customized many theories to suit agents and agent systems. Such original game theory frameworks include the Nash Equilibrium solution concept (Nash, 1950) and Operations Research (Phillips et al, 1976) amongst others.

As for agent system research, Rosenschein (1985) presented a framework for using game theory to find matrices which in turn derives actions that maximize utility. In this framework, the agents share utilities tied to certain actions and the matrices are used to maximize output from the choice of several actions among several agents. Agent can in this way choose an optimal action based on all variables in the environment. Jennings et al (1998) suggests that although the techniques used were indeed fully operable, the framework did not present flexibility enough to handle real-world applications.

Their proposed solution was the ADEPT (Advanced Decision Environment for Process Tasks) project (Jennings et al, 1996). The project involved development of a system that was to optimize business processes. Designed to be flexible and robust, the system ensured that it was applicable on many process levels, as well as in many branches of businesses. The project was tested in partnership with BT (British Telecom) and yielded some optimistic results.

Elaborating on this, Zeng and Sycara (1997) introduced *Bazaar* which included Bayesian probability to include unknown factors in utility functions. Bayesian theory provides a set of

rules establishing a degree of belief based on probability. The formulae can also be used to increase or decrease this degree based on new information. Perfectly suited for autonomous agents, the *Bazaar* framework uses these theorems to further account for the learning aspects of agent systems.

To handle the growing complexity of negotiation, Decker et al (1997) introduced the now common term *yellow page agents*. From these early theories, these types of agents are now a central part of most auction-based agent systems. As the name implies, these new types of agents did not introduce any novel negotiation techniques, but rather suggested a new way to organize the auction interaction. Functioning as brokers or blackboards, the yellow-page agents holds and provides updated lists of the services available which whomever other agents may wish to use.

Not only did this novel approach help the designer keep a tidy overview of services available, but it also provided some opportunities for better security. By having a central commanding agent governing the dynamics of the system, potentially bad agents or data may easier be detected and removed from the system (Jennings et al, 1998). This however, also implies that the before-mentioned profits of decentralization are somewhat undermined.

Programming and Implementation

Putting all these frameworks and theories to work requires a fluent transition between the design and the implementation phase. As we will study in greater detail later, for agent oriented systems development this is a particularly wide gap to fill. Because of the high level of abstractions agents usually represent, the process of implementation often involves working with large system components (Bernon et al, 2005). As a consequence, the agent systems development process needs tools to help the developer fill the gap.

The first serious implementation tool came in the early 90s (Shoham, 1993) through the language Agent-0 and its supporting framework Agent Oriented Programming (AOP). AOP was an extension of existing object oriented programming models, introducing mental state and agent related terms like commitments and capabilities. The language was largely influenced by the before-mentioned BDI framework, adopting human mental properties. Problems with this early attempt included a lack of possible long-term planning and goal-oriented behavior. Practically, the agents could only be told to perform rather primitive actions, either by other agents or the user of the system.

Agent-K (Davies and Edwards, 1994) extended this framework by integrating the before-mentioned KQML communication protocol. This enabled the Agent-0 structures to communicate in a network based on well-established methods. Expanding the scope of implementation tools also enabled some new-found restrictions. In particular, there was still a lack in semantics and communication control, resulting in unpredictable interaction and behavior.

More recent additions to implementation tools include extensions of the Java programming language. Both Jade (Bellifemine et al, 2000) and JACK (e.g. Howden et al, 2001) supplies Java API (Application Language Interface) with appropriate classes and methods to implement and govern agents and agent systems. While JACK was commercially developed and strongly builds on the BDI framework, Jade is an academic effort, incorporating the FIPA ACL language specification.

3.1.3 Applications

As mentioned above, gaining considerable attention in research, the agent system field has, as of yet, not been applied to much extent in industrial and commercial applications (e.g. Bernon et al, 2005; Zambonelli and Omicini, 2004). Furthermore, this view was confirmed in the Luck et al (2005) study, defining the technology still to be in its adoption stage. There have however been many theoretic predictions as to which commercial genres the paradigm might be best applicable, and some applications have been tested and applied in these areas. This section will first examine the theories before looking at some existing applications.

Application Fields

One major reason for the high research activity on agent systems today is its potential in dealing with complexity among autonomous components (Weiβ, 2002). A keyword in handling this cooperation is interaction. When dealing with modulus components using different platforms, architectures and languages, there is a need for "intelligent" middleware to interpret and transfer communication. As agent systems focus on both autonomy and complex interaction they should in theory be suitable for these tasks.

This merging of modular component scenario is highly relevant today, as connecting branches, businesses and partners through software modules often is a criteria for effective operations. Examples include various internet portals, international law enforcement cooperation or national academic collaborations. Naturally, this evolution has especially been sparked from the increase in bandwidth and available networking over the last decade. We will investigate efforts dealing with these issues in the next section.

Another related potential factor of agent systems lies in their ability of negotiation and pro-active behavior. These abilities are especially useful in solving optimization and coordination problems. In industry this might be applicable to optimizing production processes or distribution of products to customers. Many frameworks have been made to handle these problems (e.g. Davidsson and Wernstedt, 2002), most based on some of the negotiation techniques presented in the negotiation section above.

Luck et al (2005) assess a wide array of agent related predictions in their deliberative Delphi study. The study's participants consisted of a panel of agent system researchers and one of the predictions identified the most relevant industry sectors related to the potential factors described above by rank from 1 to 5 (fig. 3.3). In accordance with the theories mentioned, the industry sectors identified typically involves either distributed environments with complex interaction or production and coordination through negotiation and optimization. While telecommunications and transport were identified as examples of the former, manufacturing and trade represented the latter:

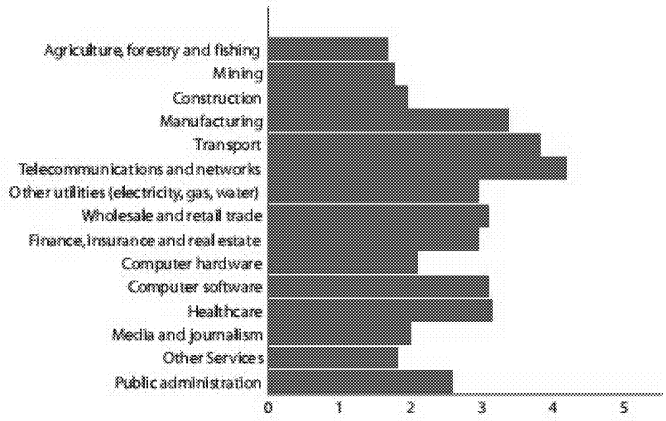


Figure 3.3: Potential Application Fields for Agent Technologies (Luck et al, 2005)

The same study also pointed to healthcare as a significant actor within the agent system field over the next five to ten years. As referenced in chapter two in this thesis, Heeks (2005) described healthcare constantly evolving towards global cooperative systems. Furthermore, Reed et al (2005) pointed out the need of more patient-oriented approaches including efficient coordination of patient examinations and treatment. These theories provided the foundation for deciding on building a patient scheduling system as a case study during this thesis.

Existing Applications

Based on the findings in Luck et al (2005) Delphi study, this section will present some examples of existing applications within the five most relevant sectors: Transport, telecommunications, manufacturing, trade and healthcare.

In transportation there is often challenges related to coordination of cargo or passengers. As an example Rebello et al (2000) presented architecture for the handling and distribution tasks in container terminals. By having agents representing ships, unloading routines, terminals and cargo types, different negotiation and collaboration techniques could be applied to distribute the cargo effectively. The system is currently being tested at an operational port.

As an example of agents in telecommunications, Hughes Research Libraries developed a system called Collaborator Discovery, where the objective was to dynamically develop collaborative groups. These online communities naturally emerged from the participants' usage of an internet browser. By recording the trail of these browsing actions, multiple user agents could collectively create suitable personalized user groups through negotiation of these trail results (Payton et al, 1999)

In manufacturing the focus of agent systems is on resource coordination and process optimization. For example, car manufacturer DaimlerChrysler exploits agent capabilities by using the agent system KoWest to coordinate materials used for car engines (Van Dyke Parunak, 2000). These agents use interaction and auction abilities to optimize work along the production line.

Building on earlier suggestions (e.g. Cliff, 1997; Gjerstad and Dickhaut, 1998), Tesauro and Das (2001) created an application for high-performance real-time bidding in an automatic auction domain. Seeing such auction being more and more common in the trade industries, the effectiveness of the algorithms is essential to obtain profitable results. By modifying the

algorithm presented by Gjerstad and Dickhaut, MGD-based agents claims to be the most effective bidders in such environments today.

As for healthcare, a good example is the Guardian Angel project (Szolovitz et al, 1994). This project was initiated in 1994 and aimed to "*construct information systems centered on the individual patient instead of the provider, in which a set of* guardian angel *software agents integrates all health-related concerns…*" Guardian Angel was originally funded by the U.S. Department of Defence, is still an ongoing attempt, but has lost funding interest mainly due to a change of strategy from the above-mentioned sponsors. Amongst several other applications, Guardian Angel resulted in a World Wide Web based medical record system which was implemented at several medical institutions in the Boston area with mixed results. Although the system was used with success in many of the institutions, the application was only operated on an internal basis due to privacy issues somewhat undermining the collaboration efforts of the project.

3.2 The Agent Oriented Software Engineering Process

The focus of this thesis is to gather some experience on the process of building and maintaining agent systems. Though relatively immature, both in a technological and industrial context, the agent research communities have already presented a wide variety of theorized positive and negative aspects of the building and maintenance processes. This section will look at some of these aspects, in addition to some examples of practical experiences.

3.2.1 Theoretical Challenges

Amongst the many theories presented there are some are some aspect of the development process that is recurring more than others. Especially three broad terms encapsulates most of these aspects: *Agent abstraction*, *autonomy* and *situatedness*. In this section these three aspects are described in detail and make the foundation for the hypothesis presented in the next chapter.

Agent Abstraction

When designing complex systems, the vast number of variables, functions and entities enforce the developer to make some abstractions. As we have seen, object oriented approaches base this abstraction on relatively passive entities, in general more reactive than proactive. But many researchers believe it is more beneficial to approach increasingly complex problems in a more anthromorphistic manner and thereby adopting the agent oriented approach. Bresciani et al (2004) describes this approach:

> "*While developing agent oriented specifications and programs, one uses the same notions and abstractions used to describe the behavior of human or social agents, and the processes involving them.*"

Furthermore, the authors believe that this higher level of abstraction can help the developers design systems more closely related to the requirements of the eventual users of the system. By narrowing this gap it is theorized that the complexity is reduced:

> "*The conceptual gap from what the system must do and why, and what the users interacting with it must do and why, is reduced to a minimum.*"

However, Wooldridge and Jennings (1998) warns not to get dogmatic about the agent abstraction, labeling the over-use of agents as a common pitfall when designing agent

oriented software. They emphasize that selecting agent types in a system is very much a balancing act between complexity and efficiency:

> *"…there is a tendency to view everything as an agent. This is perceived to be in some way conceptually clean – after all, an object-oriented language is considered "pure" if everything in the language is an object – isn't the situation the same for multi-agent systems? If one adopts this viewpoint, then one ends up with agents for everything – including agents for addition and subtraction."*

Clearly, such a system would have massive communication overhead and be very inefficient indeed. Bernon et al (2005) also warns about getting too carried away with such abstract concepts. While pointing out that the agent abstraction is a great tool during analysis and design, the authors argue that the produced concepts and models might prove to be too abstract when reaching implementation and maintenance:

> "*This can facilitate analysis and design activities but the gap to implementation is greater than with other paradigms*".

This gap is a recurring problem in many papers. There is a clear balance between keeping the complexity at a minimum through abstract concepts while producing sufficient details to be able to manage implementation and maintenance.

Autonomy

Autonomy is also a recurring term in research papers. As described above, agents are defined as proactive and independent entities. Sycara (1998) confirms this, emphasizing the advantage of modularity:

> "*The most powerful tools for handling complexity are modularity and abstraction. Multiagent systems offer modularity."*

The modular nature of agent system development is often tied to terms like flexibility, mobility and robustness. These terms suggests advantageous dynamics both during development and maintenance. As the components are independent, governed by interaction, the theory is that new components can be added and existing components removed more smoothly. Jennings (2000) point to this aspect in a development process perspective:

> *"…individual agents or organizational groupings can be developed in relative isolation and then added into the system in an incremental manner."*

Furthermore, Sycara (1998) points to the advantage of concurrently building specialized components during large and complex software development projects. As developers often have different fields of expertise and modern systems usually is presented with a wide array of problems, the authors argue that such divide-and-conquer approach is especially beneficial for today's complex systems:

> *"If a problem domain is particularly complex, large and unpredictable, then the only way it can reasonably be addressed is to develop a number of functionally specific and (nearly) modular components (agents) that are specialized at solving a particular problem aspect."*

However, as the agent system paradigm is trying to support the need for interdependent autonomous components, there is a parallel increase in complexity of interaction. Wooldridge and Ciancarini (2001) describe this shift in complexity:

> *"Over the past three decades, software engineers have derived a progressively better understanding of the characteristics of complexity in software. It is now widely recognized that interaction is probably the most important single characteristic of complex software."*

The authors hereby establish a rather important point; the complexity of the software is not removed by the autonomous approach, but rather moved. This is also gives some problems related proactive behavior. As components get more autonomous and interaction more complex, proactive behavior is increasingly necessary, but harder to govern and predict. Jennings and Wooldridge (2000) points to the behavioral uncertainty of autonomous agents:

> "*As agents are autonomous, the patterns and the effects of their interactions are uncertain.*"

Also, on a final note, some authors argue that effective autonomy is harder to achieve than most research suggest. Considering the current tools and technologies available, Dastani et al (2004) argues that flexibility is not supported by current methodologies:

> "*The methodologies implicitly suppose that agents are purposely designed to enact roles in a system. But as soon as agents from the outside may enter, the analysis, design and implementation need to treat agents as given entities.*"

Hence, the authors suggest that current developer tools do not sufficiently model the potential flexibility options of agent systems. As in most other systems, when new entities are added, there are still requirements for compliance in, for example, data structures and communication protocols.

Situatedness

Adding somewhat to the complexity of interaction, most agent definitions suggests a close reactive relationship between agents and the underlying environment or domain. This relationship helps agents maintain an updated knowledge base about the state of the system. Using this knowledge, agents can more efficiently make decisions based on their own state and goals. Zambonelli and Omicini (2004) define this relationship:

> *"…an agent performs its actions while situated in a particular environment… and it is able to sense and affect (portions of) such an environment."*

This type of relationship allows for the modularity described above. This is confirmed by Luck et al (2004) whom points out the importance of well designed environments in order to preserve other agent system dynamics:

> *"When designing agent systems, it is impossible to foresee all the potential situations an agent may encounter and specify behavior optimally in advance. Agents must therefore learn from, and adapt to, their environment."*

Hence, the authors argue that designing this symbiosis is a major challenge, and a crucial operation if wanting to preserve the before mentioned agent abilities and characteristics. Jennings et al (1998) elaborates on this challenge, pointing out that misinterpretations of domain variables can lead to deceitful actions in poor designed environments. This problem, known as *tragedy of commons*, denotes self-interest agents performing seemingly advantageous actions, but harming the system as a whole:

> *"There are, however, many problems facing such a society of self-interested agents…agents might overuse and hence congest a shared resource, such as a communications network."*

3.2.2 Documented Experience

As mentioned before, the agent and agent system technologies are for now dominated by theoretical work. Following this, there is next to none documented agent based development projects, and thus the implications of using the paradigm are not well known. This is also supported in the literature – for example Jennings (2000) concludes that *"…nobody has systematically analysed precisely what makes the paradigm effective*". There are however a selected few papers available, and some of the experiences are presented through the examples below:

Software Agents in International Traffic Insurance
In this case study (Belecheanu, 2005a) the Dutch insurance company Interpolis requested a software solution to handle transnational car insurance claims together with partners in Germany and Belgium. With data being highly heterogeneous and Interpolis requesting a robust system, the situation was suitable for an agent-oriented approach and agent technology specialists Acklin B.V. was given the assignment.

The solution was simple non-proactive agents, exchanging the information necessary across platforms and languages. This communication was anthropomorphically mapped to the behavior of employees and co-workers based on Interpolis' prior experiences.

Interpolis' choice of agent system was stated to be purely on business grounds. They evaluated Acklin B.V.s system to be far cheaper than implementing a traditional centralized system, but thus took the risk of implementing a rather novel solution. The project result was very positive, in some cases reducing the processing of some insurance claims from six months to two minutes.

During development both involved parties found the agent abstraction easy to work with, not only when analyzing and designing the system, but also when discussing more general strategies with top leaders at Interpolis. The only main problem reported was a lack of belief and enthusiasm around the introduction of a completely new technology, as many of Interpolis' divisions were operating on well-established legacy systems.

Agent-Based Factory Modeling
Our second example concerns the corrugated-box manufacturer SCA Packaging (Belecheanu, 2005b). SCA Packaging requested a software solution to reduce their stock levels while keeping their deliveries on time. Presented with this rather complicated optimization problem, Eurobios provided SCA Packaging with an agent system to solve balance production with customer needs.

Although the project was a success in reducing the stock levels at the factory, it did encounter some interesting problems. Requiring more sophisticated agents, the project ran into the problem of agents behaving in an unexpected manner. As mentioned above, pro-active, autonomous agents are intrinsically hard to govern. This problem was further amplified by the high level of user input, causing data to be inconsistent, thereby adding to interaction complexity.

Another interesting observation was made in regards of balancing the time spent on design and the time spent on implementation and validation. After a considerable time at the design phase, it was realized that the constant analysis and evaluation no longer returned effective results. It turned out that the complexity offered by such a system required a considerably longer testing period than other legacy systems might have required. By re-balancing the phase time spans, the development process again yielded satisfactory results. As we have seen, both these challenges denote some general problem areas of the agent system technology.

4 Research Method

This chapter will explain the research process of this thesis. As the project develops a design, namely a prototype for agents managing patient scheduling, one can argue that the research conducted should fall under the design research umbrella. However, as the main agenda for creating the prototype was not to present a new design idea, but rather to document experience on the process itself, the research can also borrow some elements from action research.

This poses a challenge when choosing a research approach. While design research focus on establishing the rules and guidelines to ease the building process of a given design (Reich, 1995), action research focus on evaluating a process of change, whether involving a design or not (Dick, 1997). This is a common problem in research projects. One research process or methodology rarely covers all aspects needed to produce the satisfactory results (Reich, 1995).

Therefore, in this chapter we will look at both these aspects, but use Remenyi et al (2003) more general research process as a foundation for discussion. This widely quoted proposal roughly consists of five steps which the researcher can follow through the course of a project: (1) Reviewing the literature, (2) formalizing a research question, (3) establishing the methodology, (4) collecting and analyzing evidence and (5) developing conclusions.

Throughout the following sections, these phases will be explained one by one in detail. It is worth mentioning however, that even though the steps are presented chronologically, a research process like this is rarely conducted in a clear step-by-step manner. This is also the case in this project, but iteration or back-tracing is clearly documented in the appropriate sections. Note that the theory presented at each stage is from ibid. unless stated otherwise.

4.1 Reviewing the Literature

Before defining research questions and project methodology, the researcher needs to have a solid foundation of past work and theories within the chosen subject. The literature review phase should start building this foundation by reviewing a broad range of papers and articles to get a general understanding of the subject. This should assist the researcher in narrowing the scope of a project in a sensible manner in order to define the focus the study.

As a solid theoretical foundation is essential in all academic work, this phase should occupy a significant amount of time. In fact, the literature review often represents a continuing process throughout the project, as it is impossible to foresee all needed theories at the start of a project. Also, there needs to be a strong critical emphasis during this phase. While at the start of the phase one needs not worry too much about the origin and validation of papers, it is suggested that when the focus is narrow enough, all papers evaluated should be from published academically reviewed journals.

As agents and agent systems is a relatively new field of research, this phase of the project posed somewhat of a challenge. First, although many concepts and theories exist, only few of them are evaluated and reviewed by other researchers. Also, with the exception of some (e.g. Jennings et al, 1998; Weiss, 2002; Zambonelli and Omicini, 2004), the field being so new, there seems to be a lack of existing literature reviews evaluating the flora of agent subjects. This required some extra effort from the researcher, paying close attention to the validity of proposed papers.

Furthermore, as argued in the previous chapter, the agent technologies are characterized as still being rather immature. This is evident in the nature of the published papers. While there are a good number of papers published, most are purely theoretical approaches suggesting new tools and techniques or evaluating old ones. Being at this early stage, the literature reveals shortcomings on applying and testing both the technology itself and its proposed various tools in a commercial or industrial context.

Looking back, these shortcomings are evident in the previous chapter of this thesis, where it would have been desirable with a more extensive review of previous agent oriented software development experiences. On a positive note, this particular lack of literature increases the purpose and novelty of the documented findings of the case study in this thesis.

To summarize the literature review of this project, figure 4.1 presents an overview of all references categorized by subjects:

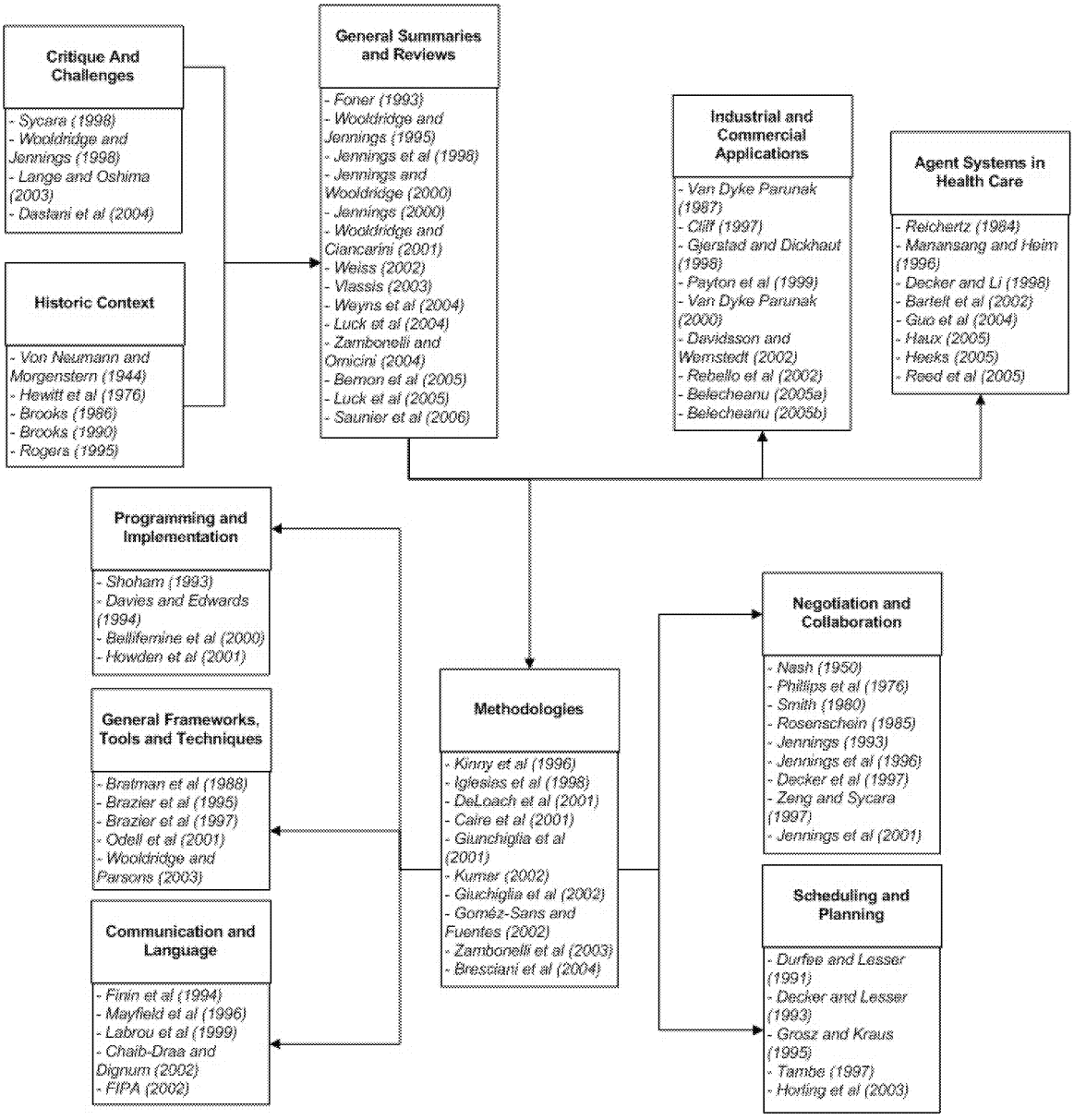


Figure 4.1: Overview of the Thesis Literature Review

4.2 Formalizing a research question

Following the literature review, one should discover some unanswered questions that can appropriately be tested and answered. Often, the research questions will be altered during the research process, as a result of new discoveries or limitations. Again, this evolutionary process should enable the researcher to narrow the scope and focus of the questions, such as to make them testable. It is also suggested to keep the number of research questions between three and five to ensure enough focus while at the same time explain scope of the problem to a satisfactory extent.

The research questions for this thesis were changed a number of times to fit the ongoing research process. Generally, the research shifted from a development project planned as design research, to the eventual mix with focus on the study of this mentioned building process. The reasons for this are explained below, while the research process is described in detail in the next section.

First, the literature review showed a lack of documentation on agent oriented development projects. This was contrasted by numerous papers on existing prototypes used as test cases for the many tools and methodologies available. As this balance was revealed it was natural to shift the research into more unexplored territories.

Also, the project was originally a collaborative effort with a local industry. Naturally, developing a system makes more sense if it will be applied for something useful. As this collaboration regretfully broke down, the system in development no longer had any specific purpose, and the research shift described above made even more sense.

To summarize, as presented in chapter one, the change of focus eventually gave the following research questions:

1. *Given the ever-increasing size and complexity of modern information systems, can agent technology contribute in making the build- and maintenance processes of such systems feasible?*

2. *What are the significant positive and negative aspects when building such type of systems using the agent approach and how do they relate to agent and agent system characteristics?*

While this challenge is twofold, the focus is on the second question. To identify these positive and negative aspects however, there is a clear necessity to also answer the first question. How both questions will be answered are explained in the following sections.

4.3 Establishing the Methodology

When deciding on methodology one should again consider the literature review. Preferably, the methodology should be well established within the field of study and validated by earlier attempts. Also, the methodology must account for limitations like time and money to ensure a realistic approach to finding results. Often, workload plans defined at the start of projects has to be shortened for this very reason, leaving the researchers with dilemmas on which elements to drop and why.

Landing on a joint effort of both participating in and observing a development process, the research methodology had to be comprised of a mixture of established methodologies. While the literature review should produce a hypothesis on key development factors, the case study should reveal both observational and measurable results to challenge these theories. The process is summarized in the figure below:

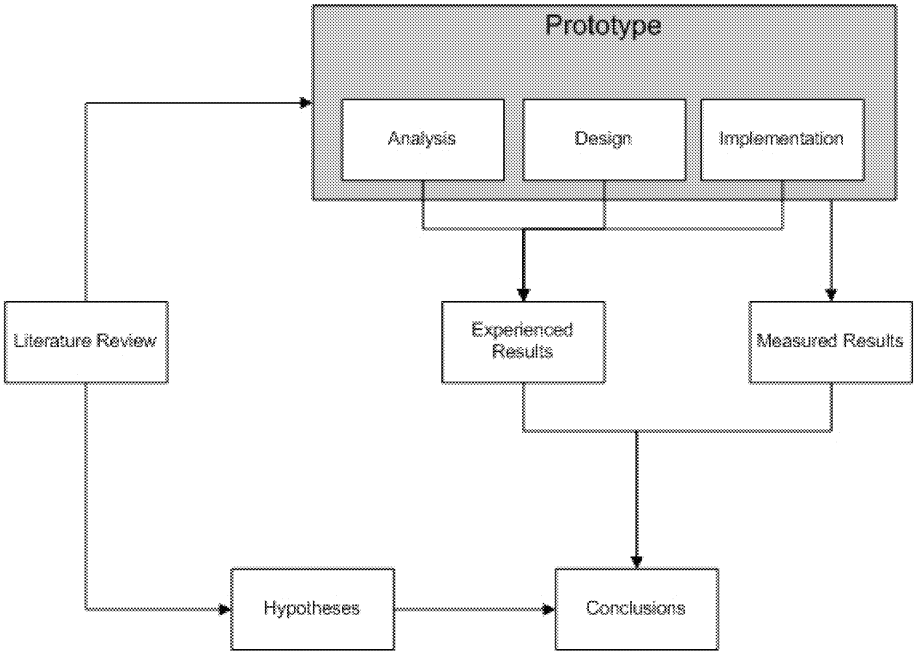


Figure 4.2: Thesis Research Methodology

Throughout the research process, the ongoing literature review process, the development of the prototype, and the hypotheses for agent oriented software development was produced in parallel. The software development part is backed by the established agent oriented software development methodology MESSAGE/UML and results in a functional agent-based patient scheduling prototype. The mechanisms of this system are described in chapter five, while reflections on the development process are presented in chapter six. The resulting hypothesis, drawn from the literature review is presented through a model in the next section, while the resulting discussions and comparisons are done in chapter seven.

4.4 Collecting and Analyzing Evidence

The evidence produced during the last parts of the research methodology should both be sufficiently validated and cover all elements of the posed research questions. It is also suggested that the evidence in general should answer questions related to *why* and *how* instead the more measurably related *how much* and *when*.

As this project in large evaluate qualitative results using the case study approach, it is suggested to develop a set of questions working as reminders to what evidence that needs to be collected and evaluated. For this thesis, the question list is represented by the hypothesis model. This model (figure 4.1) contains the suggested key factors when developing agent systems as found through work on the literature review.

In such, this thesis collects evidence both from background literature, as well as from self-experience. As mentioned above, the theoretical research focused on recurring elements in the agent oriented engineering process. This work revealed some interesting relationships

between development terms, agent system characteristics and typical agent behavior abilities. A justification of the terms found and their relationships with characteristics and behaviors can be found in section 3.2.

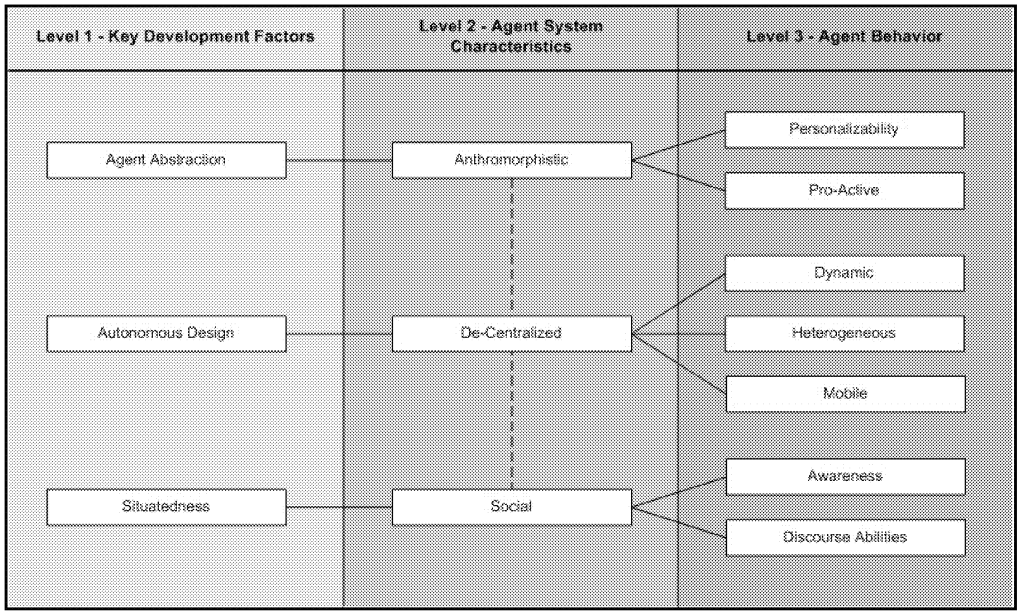


Figure 4.3: Hypothesis on Key Development Factors and Relations

As we can see, prior research tied some recurring agent oriented development terms (labeled key development factors) to specific agent system characteristics and agent behaviors. In essence, the model should describe common factors that differentiate the agent system development process from other more traditional system engineering approaches. The model is reintroduced and properly explained, before discussed and revised in chapter seven.

While developing the prototype, I sought these terms and characteristics in order to later validate whether the hypotheses could be justified. Some terms were more present than others, and they all had both positive and negative aspects. Also, while most of the model terms are purely based on experience, some of the behaviors could also be measured by testing functionalities in the prototype. More specifically, this was applicable for mobile and dynamic behaviors. During these tests, an identical prototype platform was set up at a remote host and patient, personnel and equipment agents were transported across the network and added to the new system. The results of both the experiences and the measured tests are gathered in chapter six.

4.5 Developing Conclusions

The conclusions should be both valid and convincing as this is the final saying of the research performed. As this is a master thesis, the arguments used and the conclusions drawn should convince the supervisor and external examiner that the research had purpose and was useful. Also, the conclusions must remain neutral in that the conjectures which the research was at first founded upon can be questioned and rejected. Such a rejection is as good a contribution as other standpoints.

As mentioned, the conclusions in this project will be drawn from testing the hypothesis presented in the previous section by a comparison of the experienced and measured results from the case study. Through the discussion in chapter seven, each of the factors from the

model will be tested against the experienced results and discussed with reference to the literature. This process will hopefully result in a revised model, representing a mixture of interpreted factors from literature and own experiences. The final conclusions from this work are presented in chapter eight.

5 Case Study Specifications

By conceptualizing many of the theories presented earlier in this thesis, a prototype for an agent-based patient scheduling system has been developed during the course of this project. The prototype, named *AgentMedic*, is based on well-established optimization and scheduling models from various fields of science, including optimal decision processing and game theory. In addition, agent structures and communication protocols is built on concepts from the MESSAGE/UML methodology and the Jade implementation tool.

The process will hopefully give some valuable knowledge about building and maintaining processes of such agent systems and hopefully also be a small addition to the validation process of agent-based concepts and technologies. While this chapter explains the operations and specifications of AgentMedic, the methodology workings and related experiences are described in chapter six, before discussed in chapter seven.

The chapter is divided into three main sections. In the first section agents are defined, and communication patterns and general structure is presented. The second and third section will investigate the negotiation and scheduling by detailing the workings of patient agents and personnel/equipment agents respectively.

5.1 General Structure and Organization

Before embarking on the coordination and scheduling methods, which is the core operations of this system, we need to define the agents and get a rough idea on how they will communicate during the scheduling tasks. The first section will define the agent types, before the communication patterns are sketched out in the second section. Lastly, there will be a necessary small note on the references which is used throughout the rest of the chapter.

5.1.1 Agent Types

When choosing the types of agents needed in a system it is convenient to remember the typical agent properties discussed in chapter two of this thesis. Autonomous entities requiring both social and flexible behavior must be considered (Wooldridge and Ciancarini, 2001). Further, the design should allow for the agents to make use of their anthropomorphic and pro-active nature, so as to represent a live entity in the best possible way (Foner, 1993).

This in mind, AgentMedic contains three distinct types of agents, each representing different roles in the context of a treatment process and the actions performed in medical institution. In addition to a description, we will reintroduce the general goals and possible actions defined in the patient scheduling example in chapter two:

- A **patient agent** is responsible for coordinating the full examination and treatment process of the patient it is representing. The agent must at all times keep track of the progress of the patient, his/her health condition and the management of upcoming examination or treatment tasks.

  Goals: *Get examined/treated as fast as possible*
  Possible actions: *Apply for appointment*

- A **personnel agent** represents a member of the hospital staff. Its tasks involve keeping track of the personnel competence as well as availability. Further, a personnel agent

must use these properties to optimize and coordinate examination and treatment tasks with the patient agents.

Goals: *Treat/examine patients as fast as possible*
Possible actions: *Grant/deny appointment, examine/treat patients.*

- An **equipment agent** is in the same manner as the personnel agents responsible for keeping track and coordinating the usefulness and availability of hospital equipment. The agent is involved in optimizing and coordinating the assignment of equipment to treatment tasks, taking patient and personnel into account.

  Goals: *Support treat/examination process effectively*
  Possible actions: *Grant/deny participation, support treat/examine process of patient.*

To summarize, a patient agent monitors the progress and condition of the patient, while the personnel and equipment agents keeps track of the availability and relevance of personnel and equipment. The actual optimization calculations will be done by the personnel agents. These operations could have been fulfilled by any of the agents, thus as further explained in chapter six, choosing the personnel agents for this task was simply down to a random choice.

5.1.2 Agent Communication

The figure below show the general communication patterns for the agents defined above. A patient requests an examination or treatment within some field of expertise. The information from all patients is then dealt with by the equipment and personnel agents to reach the optimal scheduling for a particular timeslot cycle. The results are sent back to the patient agents who at the next timeslot can move on to next step in the treatment cycle:

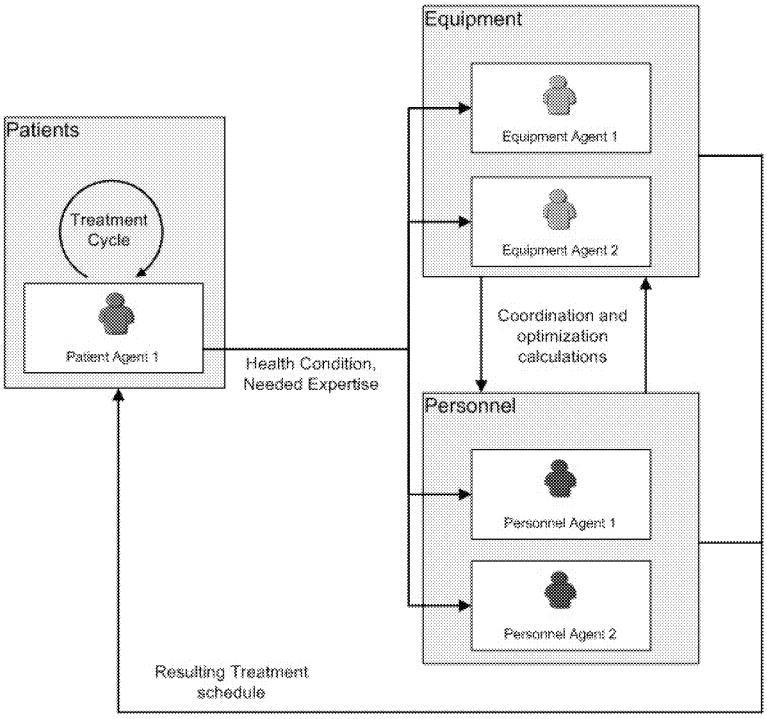


Figure 5.1: Communication Patterns Amongst Agents

In accordance to many of the scheduling and negotiation techniques mentioned in chapter three, to fulfill the optimization potential of an agent system, every agent whether it represents a patient, a representative from the personnel or a piece of equipment, constantly needs to argue their position and usage in the system. For example, a patient argues for urgent treatment or examination based on his or her health condition, a doctor argues for using his or her competence and equipment argues for being used in suitable tasks.

As a consequence, some of these arguments are superfluous as they are already covered by other agents' bidding. For example, there is no need for equipment agents to constantly argue for their usage when doctor and patient agents are ahead in scheduling and checking its availability when needed. Also, there will arise some conflicting requests. For example, a doctor might be requested for examination of two different patients, or one piece of equipment is needed in two different treatment processes.

Such issues are handled by the before-mentioned *utility* functions which is the basis of decisions during negotiation between agents. The next sections will study this in detail, presenting the basis of decisions and cooperation in the AgentMedic system.

5.1.3 On References

To sidetrack a bit, this section will present the references used throughout this chapter. As there are so many mathematical formulas and specific usage of known theorems, I found it difficult to place references throughout the process without disturbing the flow of the presentation. This section will therefore give a brief background to where the theorems and medical data originated from.

All the formulae and calculations presented below have roots in Game Theory which can be traced back as far as mathematician John Von Neumann's The Theory of Games and Economic Behavior (1944). These theories have been the foundation of many important

auction and optimization techniques, especially within economy and mathematics. More specifically, for this thesis I have used Richard Bellman's Bellman-Equations to find patient utilities and John Forbes Nash's Nash Equilibrium matrices (e.g. Nash, 1950) to handle cooperation and optimization among the agents. Nikos Vlassis' introductory work, Multiagent Systems and Distributed AI (Vlassis, 2003) has also been an invaluable guide during this process.

The various medical conditions and symptoms are examples only and should not be seen in any way as a serious approach to medical evaluation or diagnose results. The numbers used for utilities and probabilities are solely my own, arbitrary and only roughly related to the reality of the conditions and treatment methods. Examples are gathered from the Yale Medical Group which provides excellent reference on various conditions and symptoms on their website.

5.2 Patient Agents

To make sensible calculations about the scheduling of patients, we need to know something about the world state, whether it be past, present or future. Basically, we need to know the importance of a state and how to reach the next one. This section will deal with this, applying the above-mentioned Bellman-Equations to our path-finding patient agents.

5.2.1 Assumptions

To simplify the treatment process, AgentMedic evaluate and process patients during *timeslots*. A timeslot is a one minute period where patients first are evaluated, before prioritized patients are examined or treated based on their condition. In AgentMedic, the timeslots are continuous cycles that run until all patients are treated or new ones have arrived.

For further simplification, there are a set number of three treatment cycles per patient. The preliminary examination finds the possible general conditions for the patient based on his or her symptoms. The secondary examination is more thorough and might involve a physical examination, looking at the patient's medical record and so on. The last cycle is the actual treatment, ranging from surgery to diet programs. While having a set number of treatment cycles must be considered a simplification, two steps of examinations before treatment generally complies with a real life hospital visit.

5.2.2 Probability and Utility Introduction

Consider past observations *o* and actions α at time *t*. These two measures can be denoted α_t and o_t. An agent may use this information to decide which action to do next in the following form:

$$\pi = (o_1, \alpha_1, o_2, \alpha_2, o_3, \alpha_3 ...) = \alpha_t \qquad \text{(i)}$$

In plain English, the function π is used by an agent to decide which action to do next. The action is in this function a direct result of the agents' previous actions α and observations *o* at time intervals *1*, *2*, *3* and so on.

This function however, runs into the before-mentioned partial observability problem, stating that observations usually are incomplete or inaccurate. In agent systems, observations should preferably reveal as much as possible of the state of the world *S*. But this is often problematic. For example, there are always uncertainties related to human observations and conclusions. In

our case, this includes the unknown outcomes of a patient diagnose and the results of treatment. To overcome this problem, we need to introduce some statistical measures.

$$P = (s_{t+1} \mid s_t, \alpha_t) \qquad \text{(ii)}$$

In words, this simply states that there is a probability *P* of state s_{t+1} to incur when action α_t is performed at state s_t at time *t*. By using probabilities to account for inaccuracies in the system, the mechanisms deal with the partial observability problem, helping agents to decide which states can be reached at which probabilities.

But how do we differ between several obtainable states? Clearly, there will be some states which are more desirable than others, based on the goals of the agents. Keeping this in mind, we can introduce the notion of utility. Utility *U* is a measure used to capture the desirability of an achievable state s and is calculated by the user based on real applicable values.

The next two sections will cover this in detail, presenting an example of two patients going through all three examinations and treatment stages. Our two patients, Mary and John have just arrived at the hospital; Mary is suffering from frequent headaches while John is complaining about heartburns. The next section will explain how their respective patient agents calculate utilities and probabilities in this particular situation.

5.2.3 Utility and Probability Measures for Patient Agents

Utility in our context is the diagnosed health condition of our patient. As mentioned above, each patient undergoes three distinct treatment stages. In our scenario each new stage obtainable by one of the patient agents are considered a new obtainable state in the agent world. Thus the main challenge for the agents is to decide which obtainable state that takes precedence under which conditions and how valuable its present stage is.

Stage 1: Preliminary Examination
At the preliminary examination stage, we can assume that there already exists some basic knowledge about the health state and/or condition of the patient. Usually this originates from the patients' general practitioner or through some nurse/registration procedure when arriving at the medical institution. This leads to the mapping of probabilities to what category of disease or illness the person in question might have.

For example, when John arrived at the hospital, his symptoms were heartburn and some nausea. Early assessment of these symptoms points to two possible medical conditions:

- *Gastric/duodenal ulcer*: An open sore on the inside lining of the small intestine/bowel.
- *Gastroesophegeal reflux disease (GERD)*: An inflammation in the esophagus (the tube leading from throat to stomach).

While an ulcer is quite serious and in many situations lead to surgery, GERD (usually referred to as just *heartburn*) is a more common condition, usually caused by and cured by a change in lifestyle. Given knowledge like this, the utilities *U* at this stage correspond to the seriousness of the two possible outcomes.

Further, while John's symptoms were not specific enough to reach unambiguous conclusions about his condition, we can still assume a difference between the probabilities *P* of the two possible outcomes. While nausea is a symptom of ulcer, it is still a very rare disease and nausea can be caused by many other factors. Also, John shows no other symptoms related to ulcer. GERD however, being far more common, is more probable in this case. Taking both utilities and probabilities into account, a representation of the first and second stage of John's hospital visit is shown in the diagram below:

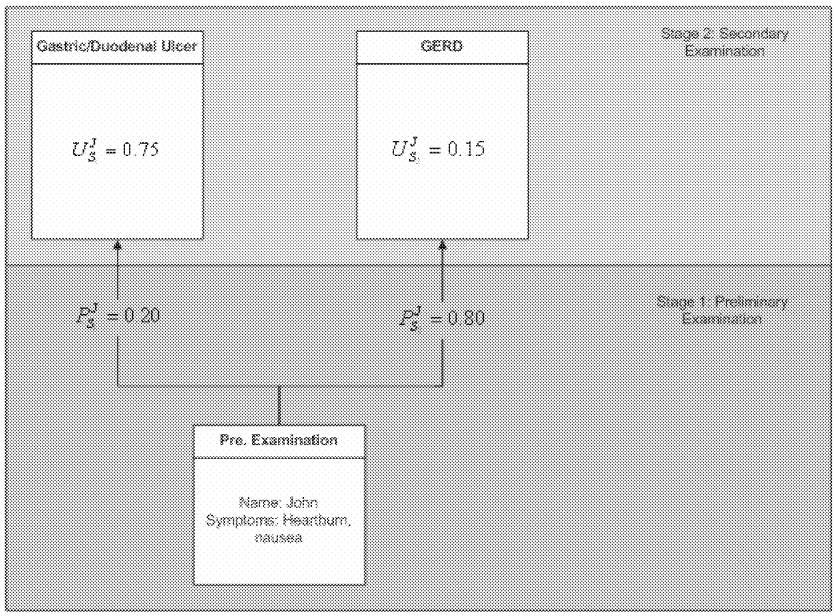


Figure 5.2: John's Preliminary Examination

We can see from the model above that *before* the preliminary examination is done, we know there is a $P_{s_1}^J = 20\%$ chance that John has a gastric/duodenal ulcer condition, and a $P_{s_2}^J = 80\%$ chance he has GERD. Also, the two conditions are rated in terms of seriousness by a utility factor (ranging from 0 to 1). As mentioned above, ulcer is considered a much more dangerous condition than GERD thus the utility value for the former is considerably higher than that for the latter.

After the preliminary examination, one of the diagnoses is determined and the patient moves on for further tests at the secondary examination stage.

Stage 2: Secondary Examination

As in stage one, there are some prerequisite knowledge about the outcomes. As treatment methods are well known amongst the different conditions, the probabilities and utilities are set accordingly. In this case, the probabilities refer to what type of treatment the patient is likely to be scheduled for, while the utilities indicate the seriousness/importance of the treatment process.

Returning to our example, assuming John was diagnosed with ulcer in the preliminary examination there are a number of treatment methods available:

- *Change of lifestyle*: For example cutting down on unhealthy food, alcohol and cigarettes. This requires no treatment at the hospital.

- *Medications*: Helps to stabilize acid levels.
- *Antibiotics*: The bacterium Helicobacter Pylori is often the main cause of an ulcer, thus antibiotics can in such cases be an effective cure.
- *Surgery*: Includes various procedures to decrease and stabilize acid levels.

All these treatment methods have different levels of importance. For example, surgery is a much more complicated and serious procedure than medication. Also, there is often urgency constraints related to surgical procedures, while setting up and monitoring medication programs often allows for some "slack" time if there are other more important tasks at hand. Figure 3 illustrates John's secondary examination with a suggested set of utilities and probabilities involved:

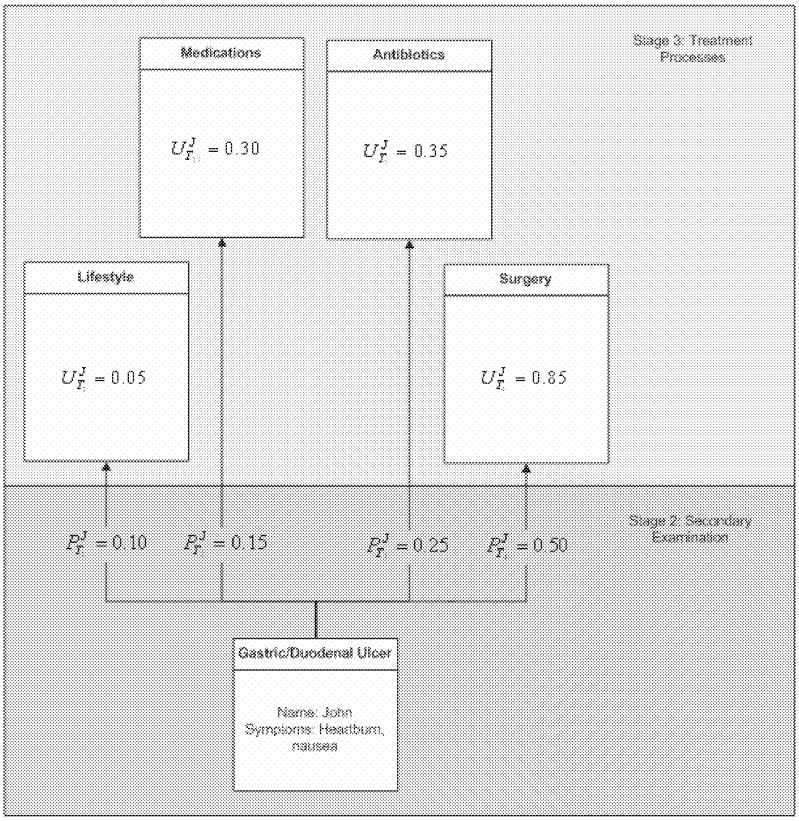


Figure 5.3: John's Secondary Examination

Again conferring with the model above, we see that *before* secondary examination, there is a $P_{T_4}^J = 50\%$ chance he will have to undergo surgery. The probabilities for needing one of the remaining treatment methods namely change of lifestyle, medication program and antibiotics, are $P_{T_1}^J = 10\%$, $P_{T_2}^J = 15\%$ and $P_{T_3}^J = 25\%$ respectively. Further, the utilities indicate that surgery, $U_{T_4}^J = 0.85$, is by far the most serious procedure, while a change in lifestyle, $U_{T_1}^J = 0.05$, is not a serious/important treatment task.

After the secondary examination, the patient is scheduled for treatment and is ready to move into the last stage of the medical institution visit.

Stage 3: Treatment Processes
The third stage is the actual treatment process. At this stage, the patient's health condition should haven been diagnosed fairly precisely and the utility measure from the treatment processes found at stage 2 can be re-used and directly applied to represent the importance of this last stage.

5.2.4 Optimal Decision Process

Knowing the general seriousness of the various stages and the probabilities of reaching them is only the first step in a patient agent's evaluation of its patient. As mentioned in the introduction of this chapter, an agent needs to know how to argue its case, trying to get its patient prioritized over other patients. To illustrate this, we need to bring in another patient agent to our example.

To continue from the last section we include Mary in our system. In the same manner as John, Mary, complaining about frequent headaches, is diagnosed at an early stage, either by her general practitioner or when arriving at the medical institution. She is at this stage thought to have either migraines (vascular headache) or muscular contractions. Different treatment methods are relevant to the two conditions, but muscular contractions are in general a more serious health threat.

Combining all three stages and including Mary's visit to the hospital, our two example patients may be presented in the following way:

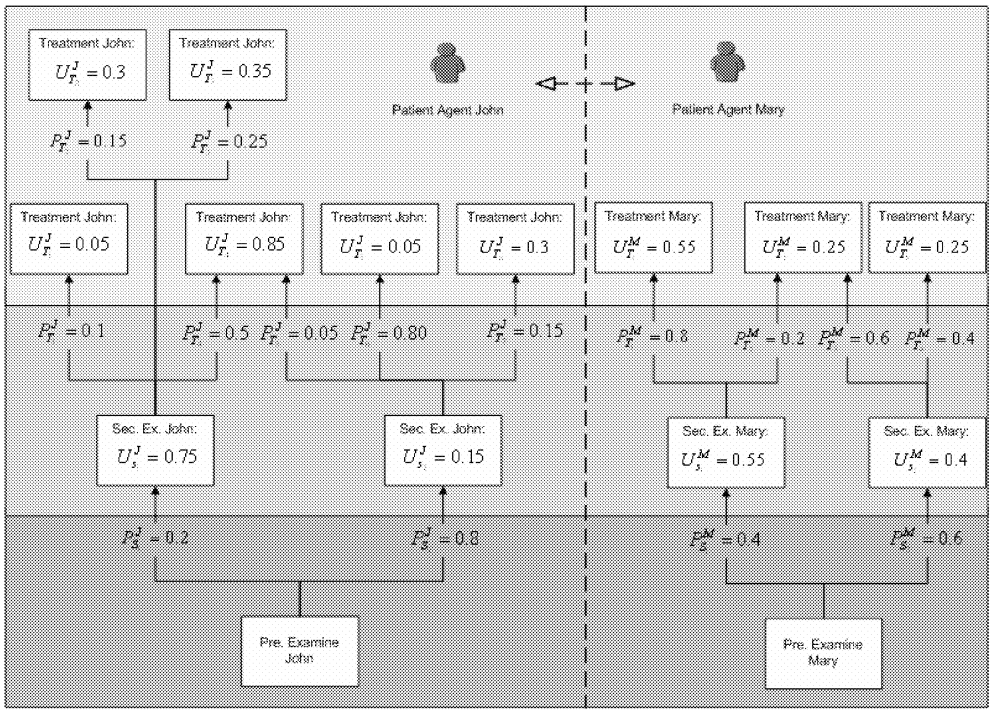


Figure 5.4: John and Mary Parallel Patient Agent Treatment Processes

Technically, figure 4 shows the process path tree of two patient agents, one representing John and one representing Mary. Assuming there is only one doctor available, the two agents should cooperate and bargain to obtain effective scheduling for the different stages of the two

patients. Before stage one, both patients must be scheduled for their preliminary examination, so which one goes first?

This is done by using the relevant probability *P* and utility *U* values at each stage. Generally, knowing the utility value *U* and the probability *P* of reaching all achievable states, an optimal action α_t^* can be calculated the following way:

$$\pi^*(s) = \arg\max_{\alpha} \sum_{s^1} \mathrm{P}(s^1 \mid s, \alpha) U^*(s^1) \qquad \text{(iii)}$$

In words, for each state a related utility is calculated by finding the max return value of all possible actions obtainable in this state. The max return value is the utility multiplied by the probability for each new obtainable state. In our hospital situation we need to find these temporary utilities to be able to prioritize one task over another so that resources like personnel and equipment can be used effectively.

Using our example from figure 4, the two agents need to agree on which patient goes first. Using the formula in (iii), we can calculate the temporary utility values for John (iv) and Mary (v).

Hence, the results indicate (0.27 < 0.46) that preliminary examination of patient Mary should be prioritized before preliminary examination of John. Again, this is based on the possible outcomes of the examination. After the preliminary examination, we assume Mary was diagnosed with migraines and her agent now has to calculate the new temporary utility (vii) and argue for being prioritized at the next stage.

This time, Mary must wait for further examination, as John's preliminary examination is calculated to be more important and must be prioritized. The agents continue to cooperate in this way until all patients are treated at the treatment stage at cycle three. We assume John was diagnosed with ulcer (viii), but after further tests was relieved only to be put on a supervised change of lifestyle treatment (xiv). Further, we assume Mary was put on a medication treatment (xiii) after her migraine diagnose. We see that the agents always schedule the next action based on the importance of reaching the next state. To

(iv) $\alpha_{pre.ex.John} = (0.2*0.75)+(0.8*0.15) = 0.27$

(v) $\alpha_{pre.ex.Mary} = (0.4*0.55)+(0.6*0.4) = 0.46$

$\downarrow (\alpha_{pre.ex.Mary} \rightarrow \alpha_{sec.ex.Mary2})$

(vi) $\alpha_{pre.ex.John} = 0.27$

(vii) $\alpha_{sec.ex.Mary2} = (0.6*0.25)+(0.4*0.25) = 0.25$

$\downarrow (\alpha_{pre.ex.John} \rightarrow \alpha_{sec.ex.John.1})$

(viii) $\alpha_{sec.ex.John.1} = (0.1*0.05)+(0.15*0.3)+(0.25*035)+(0.5*0.85) = 0.5625$

(ix) $\alpha_{sec.ex.Mary2} = 0.25$

$\downarrow (\alpha_{sec.ex.John.1} \rightarrow \alpha_{treatment.John.1})$

(x) $\alpha_{treatment.John.1} = 0.05$

(xi) $\alpha_{sec.ex.Mary2} = 0.25$

$\downarrow (\alpha_{sec.ex.Mary2} \rightarrow \alpha_{treatmentMary2})$

(xii) $\alpha_{treatment.John.1} = 0.05$

(xiii) $\alpha_{treatmentMary.2} = 0.25$

$\downarrow (\alpha_{treatmentMary2})$

(xiv) $\alpha_{tratment.John.1} = 0.05$

$\downarrow (\alpha_{tratment.John.1})$

get a clear view of our example, we re-introduce figure 4 with the chosen paths calculated above marked in red:

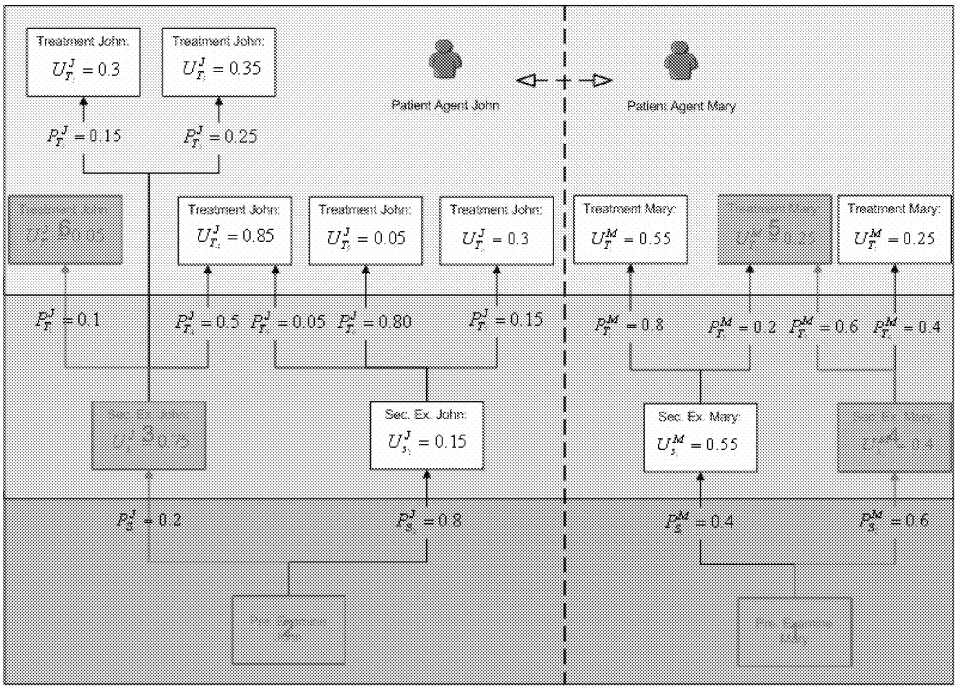


Figure 5.5: John and Mary Resulting Agent Tree Paths

The path clearly shows the connection between the agents' chosen path and the related probabilities and utilities: Due to the very slim chance of John having an ulcer, Mary gets to be examined first, as both vascular abnormalities and muscular contractions often pose quite serious health risks. When Mary is examined however, she is diagnosed to have a type of migraine associated with relatively risk-free treatment methods. The agents therefore decide to examine John for his possible ulcer before diagnosing Mary any further.

As it turns out, John has a potentially dangerous ulcer and further tests are done to find a suitable treatment method. After blood and endoscope tests, John is relieved to hear he need not undergo surgery or any medical treatment. Instead he will participate in a supervised diet program to treat his ulcer. As this treatment method is considered rather lightweight in a hospital context, it is Mary's turn to get her second examination.

After a long question/answer session and a blood test, Mary is scheduled for a medication program to ease her migraine attacks. The treatment, consisting of an informative conversation with a doctor, before a hand-out of the prescription, is prioritized before John's meeting with his doctor. Finally, John gets his diet and lifestyle program and both patients have been through the full treatment process and can return home.

As explained in the communication section above, the patient agents distribute the utilities to personnel and equipment agents for them to evaluate. When a utility is distributed, we may consider this action as a task generation, proposing a new task for the personnel and equipment agents. The next section will take a closer look at this process.

5.3 Personnel and Equipment Agents

We have shown that the patient agents are capable valuing tasks based on patient information, but we also need to bring personnel and equipment variables into the equation.

5.3.1 Assumptions

As with the patient agents, we assume that some utility variables already have been calculated. For personnel, this includes competence levels within different medical fields, as well as an availability factor. The utilities are simply represented as a number between 0 and 1. For example, in medical competence, 1 represents full expertise within a field, whereas 0 represent no knowledge at all.

The availability of personnel denotes the position and status of personnel in respect to the patient in question. This involves actual position (personnel might be at home or at another institution) or the personnel agent already having a long list of tasks to do. For simplicity, availability is indicated by a 1 for available or a 0 for not available.

For equipment, there are measures for relative position and usefulness in respect to the task at hand. Whereas usefulness is similar to competence, denoting the relevance of the equipment within a medical category, position differs slightly from the personnel agent's availability factor. As equipment often is hard to move due to physical constraints, equipment agents is often the deciding factor of where the task is going to be performed.

The following sections will elaborate on personnel and equipment agents, and explain how they can best be assigned to the most appropriate patient tasks.

5.3.2 Assigning Roles

First off, to decrease the number of possible personnel and equipment alternatives we need to assign a role to each task. To simplify this concept, AgentMedic defines rather broad categories to encapsulate these roles, and relate them to the competence and usefulness utilities for personnel and equipment agents.

For example, the symptom *heartburn* and the condition *ulcer* are both assigned to the more general role of *digestive disorder*. Further, digestive disorder is part of the competence field for some of the personnel agents and has a usefulness factor in some of the equipment agents. Hence, the tasks involving the symptom heartburn or the condition ulcer are linked with personnel having competence in-, and equipment with a usefulness factor for digestive disorders.

The roles are assigned to tasks ad hoc by the patient agents as patients are processed. This ensures that no unnecessary tasks are being evaluated and assigned when in fact no personnel or equipment is needed within that field.

When a role is assigned to a task, the patient agent can start to look for available personnel and equipment agents within the given competence and usefulness fields. Suitable personnel and equipment agents are then added to a list and sorted by potential. Thus, the result of this process is a set of tasks, represented by the corresponding patient agents, each holding a "wish list" for most suitable personnel and equipment agents.

For example, returning to our patient John from section 5.2.3 at stage two in the patient process – his agent finds two suitable personnel agents and one suitable equipment agent to help at the second examination:

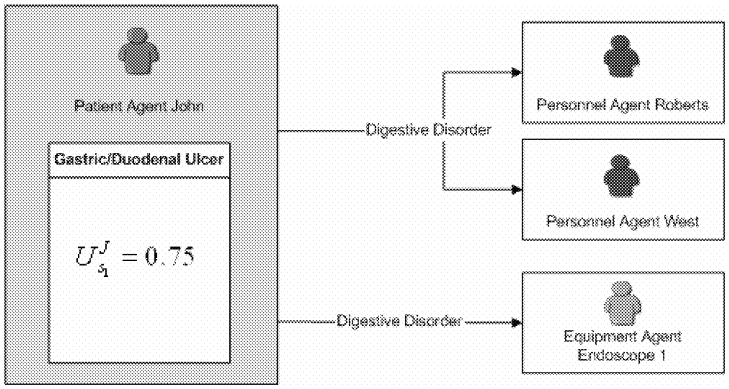


Figure 5.6: John's Role Assignment

Thus, this is patient agent John's wish list, but how do we choose between the different available agents? And what if other tasks are scheduled at the same timeslot and need the same resources? The next section will present how personnel and equipment agents coordinate to assign resources to the most appropriate tasks.

5.3.3 Distribution Using Nash Equilibrium

We now have a set of patient agents with tasks containing requests for personnel and equipment. The collaboration between patient, personnel and equipment agents must consist of a method to involve all variables of all agents to produce an optimal solution for the current ongoing and scheduled tasks. To solve this, AgentMedic use theories on game theory and more specifically the concept of Nash Equilibrium (NE).

Consider a possible action α for a personnel agent *i*: α_i. First, we define an action for a personnel agent to be the assignment to a patient task, and the action of a patient agent to be the acquisition of a personnel agent. We can then introduce the concept of a joint action $(\alpha_1,...,\alpha_n)$. A joint action denotes the collaborative result of two or more agents performing actions simultaneously. Further, the utility function *u* for agent *i*; u_i can be used to measure the value of and compare such joint actions.

Drawing from the definitions above, in our context, a joint action is the assignment of a personnel or equipment agent to a patient task. Using the utility function, we can compare these joint actions. For example:

$$u_i(assign_{patient.John}, assign_{personnel,Roberts}) \qquad \text{(xv)}$$

…can be compared to:

$$u_i(assign_{patient.John}, assign_{personnel,West}) \qquad \text{(xvi)}$$

The comparison should assign the most competent and available of the two members of the personnel; Roberts and West, to our patient John. To do this, and to also include all the other agents' actions α_{-i}, we introduce the concept of NE:

$$u_i(\alpha^*_{-i}, \alpha^*_i) \geq u_i(\alpha^*_{-i}, \alpha_i) \qquad \text{(xvii)}$$

In words, this formula state that a joint action is NE if no agent can improve utility by choosing a different action. Thus, NE represents optimal distribution for the two agents involved.

For example, consider a timeslot generating four tasks by four different patient agents; John, Mary, Chris and Sarah. The four agents define roles for their tasks and add personnel to their list in the following manner:

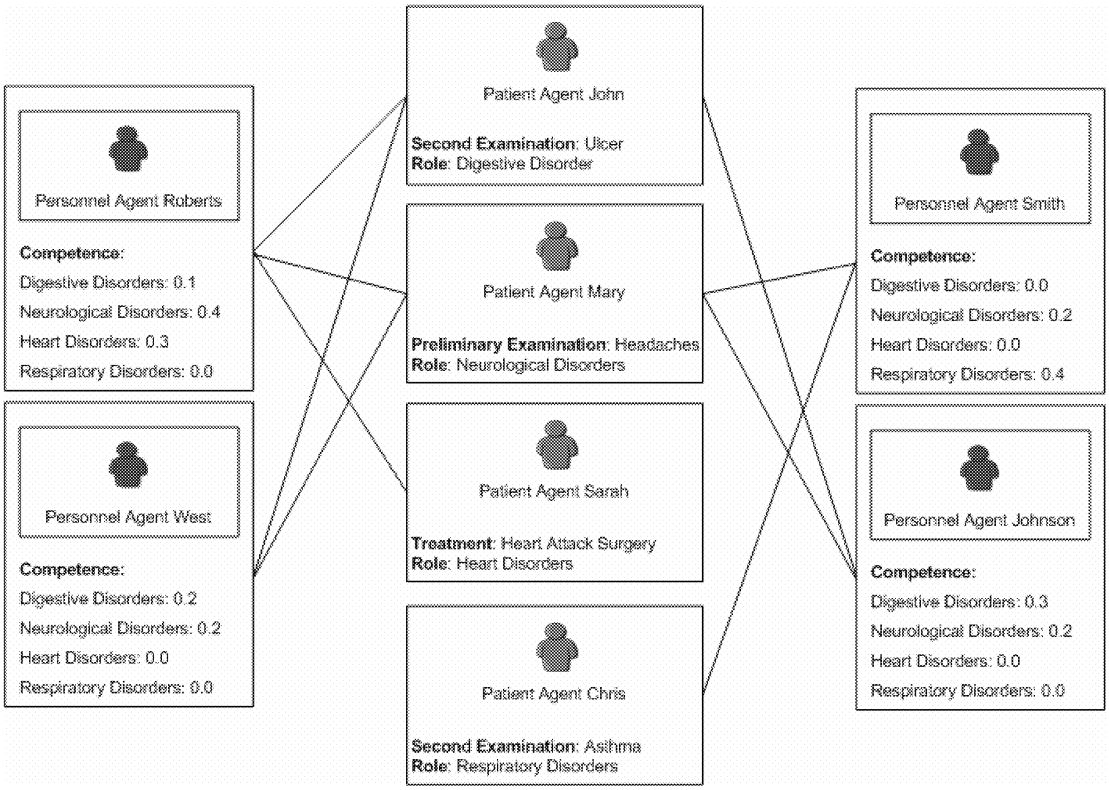


Figure 5.7: General Role Assignment Example

The diagram above shows that each patient agent finds all personnel agents that have some competence within the role that is assigned to their next task. For example, John would in this example make a sorted list with the following personnel agents: {Johnson, West, Roberts}.

Now that related tasks and personnel have been tied together, we can use a matrix to find NE. The matrix shows both the competence utility and a relative availability factor:

	Roberts	**West**	**Smith**	**Johnson**
John	*0.1 , 0.0*	*0.2 , 0.85*	*0.0 , 0.85*	*0.3 , 0.85*
Mary	*0.4 , 0.3*	*0.2 , 0.3*	*0.2 , 0.3*	*0.2 , 0.0*
Sarah	*0.3 , 0.15*	*0.0 , 0.15*	*0.1 , 0.15*	*0.0 , 0.0*
Chris	*0.0 , 0.5*	*0.0 , 0.5*	*0.4 , 0.5*	*0.0 , 0.0*

Note that the availability factor (1 or 0) has been multiplied with the patient utility. The three factors (utility, competence and availability) are now part of the same matrix and can be calculated using the NE formula. To calculate optimal distribution, we apply (xvii) to each matrix cell. This produce the following set of NE joint actions:

{(John, Johnson), (Chris, Smith), (Mary, Roberts)}

With the correct personnel assigned, the patient agents should connect to the appropriate equipment agents in the exact same manner. Only patients that have been assigned a personnel representative may look for a piece of equipment however, as with no personnel, there will be no examination or treatment. We find the following results when applying the NE formula to each joint action:

{(John, Endoscope), (Mary, Surgery Equipment)}

Hence, all three patients but Chris were assigned relevant and available equipment for their task. As we can see, the NE does not always give fully satisfactory results. This is due to having two “rounds” of NE, first for personnel then for equipment. As no equipment was available for Chris, we need to check if our last patient Sarah can be assigned to the now available personnel Smith. As this can be the case for several patients, we run the NE matrix again with the remaining patients and personnel:

{(Sarah, Smith)}

Finally we run the last NE equation, checking if the necessary equipment for Sarah is available. We achieve the following result:

{(Sarah, Blood Test Set)}

As a result of this, patients John, Mary and Sarah are scheduled for examination/treatment in this timeslot. Chris will have to wait for the next timeslot where the necessary equipment hopefully is available.

This concludes the technical specification of the AgentMedic prototype. By effectively assigning personnel and equipment to patient examination and treatment cycles, the patient scheduling system presented should be applicable at any medical institution. The next chapter will further investigate the prototype building process, but focus on the experiences during agent oriented development.

6 Results

This chapter will present the experiences gathered from my development of the prototype, and thereby directly investigate answers related to this thesis' research question presented in chapter two and the hypotheses presented in chapter four. Through the main components of the applied design methodology interesting experiences will be presented in a chronological manner in the first section, while the measurable results are presented in the final section. Some of the technical specifications fully presented in chapter five will be revisited for convenience. All results are discussed in detail in the next chapter, with the aim of structuring the key findings according to the key development factors hypothesis.

6.1 Development Experience

As presented in chapter four, the development process followed the steps of the MESSAGE/UML methodology. Being traditional in the sense of its stepwise phase progression, this section will introduce the experiences gathered during the course of the three main development phases; analysis, design and implementation. As such, this section will present appropriate examples from functionalities in the prototype to underline the findings. For a full specification of the case study, please refer to chapter five.

6.1.1 Analysis

The analysis part of MESSAGE/UML mainly consists of two processes: Identifying the main components of the system and defining an overview of interaction. Whereas the identification of main components should help the developer differ between agent based entities and more passive objects, the interaction overview should assist in establishing the connections between them. The two processes are described below with the corresponding findings from the case study.

Identifying components

During early analysis the MESSAGE/UML methodology is concerned with defining classes of agents labeled organizations with assigned roles and tasks. After a study of the entities in the case study scenario (a medical institution), I found it challenging to identify the corresponding components of the system. This required balancing fully operational agents with more passive entities like objects or variables. While agents allow sophisticated structures and capabilities, they also require more computational power and increase interaction complexity.

Hence, one of the main challenges in this definition process was to find suitable entities to fit agent characteristics. I saw no point in establishing agents if they didn't need any typical agent behavioral abilities. In MESSAGE/UML such characteristics is represented as purpose, roles and services. The more passive entities can therefore more effectively be designed as for example objects or as entries in a database.

This in mind, it is worth noting that the agent abstraction was not always the main focus throughout the development process. As suggested above, some tasks were best left to more traditional approaches. The balance between agent oriented structures and traditional functional operations is imperative to make effective use of the agent paradigm.

As an example of the above-mentioned balancing challenge, the scheduling mechanisms for the prototype were defined to be operated by a mix of agents and passive entities. While the agents provide the necessary data, optimization operations are designed in a more traditional

manner. As these are plain calculations there was no use for anthromorphistic abilities. In my prototype, personnel and equipment agents all deliver their health condition, competence and expertise respectively. The matrix calculations are operated subsequently as straight end-to-end operation.

Furthermore, I found there was no need to define all capabilities and functionality at this early stage in development. Rather, I thought along the anthromorphism lines; which entities in the organization could benefit from human-like behavior?

So when choosing suitable agent types for my patient scheduling system, there were many factors to consider. But as the prototype is rather specialized with relatively few variables, entities and operations, the simplest approach was to select the agents based on real life entities and their operations. Though fairly complex, both entities patients and personnel, roughly have the same internal goals and tasks, not differing much from patient to patient, or from personnel to personnel.

Defining interaction overview
The second main process during analysis was to define the interaction between the components described above. In MESSAGE/UML, this early process does not enforce the developer to define any detailed interaction operations as of yet. Rather, the process should define a rough estimate of which entities should communicate what to whom.

During analysis, this process worked in tandem with the process described above. While working on selecting components, interaction pathways naturally emerged. Also, while it was useful to think of the anthromorphistic characteristics when identifying possible agents, I found it equally beneficial to consider the communication between them. For instance, an entity might not need any sophisticated structure and mental properties, yet it might require participation in complex decision processes. In such cases the need for discourse abilities might gave rise to considering the use a new of agent abstraction: namely equipment. As this entity was thought to participate in the scheduling techniques, it was clear that it also needed the social abilities to engage in negotiation. Using these assumptions, equipment agents were established in the analysis phase.

Furthermore, in our medical institution scenario, the interaction overview should define how data could transfer among agents to supply the grounds for the optimization calculations. This early on however, it was difficult to picture exactly which agents should handle this traffic and where the scheduling mechanisms should take place. In essence, it was easier to imagine what the agents should transfer, than to whom. This however is in accordance with the MESSAGE/UML methodology, which warns not to get too specific in interaction design at this early stage.

6.1.2 Design

As the process moved into the design phase, the development naturally became more detailed and specific. In this phase the MESSAGE/UML methodology finalize the system modeling by specifying the agent types, defining communication protocols and model interaction with the environment. All three processes are described below.

Agent specifications
Using anthropomorphic analogies, the design phase is focused on defining tasks, actions and roles based on agent purpose. While this process mostly provided a smooth transition from analysis to design, I also encountered serious challenges in situations where such analogies could not be drawn.

More specifically, whereas the patient and personnel agents could easily map abilities from live entities, equipment agents could not. Patients and personnel both have a clear purpose in being treated and treating respectively. Also, based on their roles and as explained in chapter five, the tasks to achieve these goals should assist in the optimization process by arguing their position in the scheduling process. As such, analogies like health condition, expertise and availability could easily be drawn.

In contrast, equipment agents could not be mapped as naturally. These agents differ from the abovementioned agents in that they do not represent a live entity, and for this reason I could not map these entities directly in terms of mental and behavioral properties. Instead, the main consideration when designing equipment as a type of agent was the need for interaction abilities, so that equipment can participate in the optimization operations in the patient scheduling process.

As explained in chapter five, equipment relevance and availability is now evaluated in similar manner as that of the personnel agents. Without this cooperative aspect, equipment would be passive non-argumental resources used rather inefficiently.

Communication protocols
The design phase of development also involved thorough modeling of interaction and communication. As described earlier, agents are optimally pro-active and social entities, making cooperation and coordination structures unpredictable and complex. While a detailed plan of these processes might ease the implementation, it might also be wise to generalize some interactions in order to allow for testing and changes later on.

This generalization approach was largely used in the design of the prototype. As mentioned earlier, the interaction parts of scheduling was hard to define at a design level as it was unclear where the actual optimization operations should take place. Two possible options were possible: at a specialized optimization agent or local calculations at each patient, personnel or equipment agent. After testing at the implementation stage revealed difficulties with a centralized special agent, the latter were eventually chosen. In fact, I found that local calculations were practically impossible. As parameters from all available patients, personnel and equipment had to be accounted for to obtain optimized scheduling, there had to be one agent to evaluate all parameters simultaneously. But, as mentioned, at this stage design of interaction were still kept at a general level.

So although the communicative paths were somewhat unclear, the mechanisms for optimization were decided upon and available. This enabled rigorous modeling of what to communicate but, as mentioned, not to whom this data should be transported to.

Domain interaction
Further, the interaction focus of this stage also demanded a more detailed design of the environment. As agents should inhibit reactive abilities, responding to changes in its domain, an environment in agent terms is not simply an underlying platform providing communication

and such. These operations are technically complex and hard to model and design, but luckily there are tools to aid the establishment of an environment, and a general domain interaction model is sufficient.

As presented in chapter four, the Jade implementation tool handled most of the underlying domain characteristics. This framework allows a single specialized agent to act as a domain supervisor or “yellow page” look-up, keeping track of all available services and agents in the system. The design of domain interaction was therefore mainly concerned with assigning and requesting services to this agent. As this information was readily available in the agent specification models, this process simply linked them to this yellow page agent.

6.1.3 Implementation

Having finished the analysis and design parts of the project, a set of models and specifications was now available to assist in the implementation process. The MESSAGE/UML methodology only supports the developer through the analysis and design phases, leaving the implementation open to other, more applicable tools. As mentioned, the Jade java API was used for this purpose. In general, the implementation process consisted of three relatively distinct parts: Implementation of agents, components and domain. The three processes are described in detail below.

Implementing agents
Both the analysis and design phases were focused on and around the agent abstraction. This greatly eased the design process by abstracting away unnecessary details, keeping the early development stages free of distractions. However, when approaching implementation, this high level of abstraction presented some problems. Without sufficient technical details, I found the implementation phase to be increasingly complex and extensive, and thus lasted longer than expected. However, this gap between design and implementation is to some extent covered by another abstraction; namely the readily available Jade platform.

Based on Java API, this platform provided most of the underlying architecture and interaction structures needed to implement agents based on the design models. Through its java classes and methods, the agents could be implemented much in the same manner as when implementing objects. This greatly helped the implementation process, as the developer both had experience within java and object oriented approaches in general.

By providing easy-to-use agent constructs through java classes, with appropriate methods and communication protocols, the agents were up and running quite early in the implementation phase, despite the problems described above. Having patients, personnel and equipment communicating at this early stage helped a great deal as trial-and-error methods could be applied throughout the rest of the implementation.

Implementing modules
Having defined and modeled the agent types, implementation could be done in a highly modular fashion. One might argue that being able to work on separate modules simultaneously is not such an advantage when there is only one developer on the project. However, while missing the opportunity for concurrency, I discovered other advantages, like preserving the agent abstraction and easier testing.

During the development, this modularity aspect presented a nice overview of the system as well as providing easy access to change the modules individually. It is not hard to imagine

however, that had the system been more complex and focused on pro-active behavior, the interaction between modules would be increasingly hard to manage. Indeed, this is a problem recognized in other case studies. While removing the complexity of a centralized control unit, the emphasis is instead on interaction. As the focus on interaction can help finding solutions to make different types of system cooperate, most agent researchers see this as a step in the right direction. The fact remains however, that the complexity has been moved, not removed.

As mentioned, no such unexpected behavior related problems were encountered during implementation and testing of the prototype. This has however a natural explanation in its limitations in pro-activeness and learning. As time constraints enforced the prototype to be rather specialized, many such behavioral characteristics were simply not implemented. However, as will be discussed in the next chapter, some such behavior was forced upon some agents mostly as a means of gaining experience for this thesis.

Implementing domain
Finally, interaction and environment were dominant focus areas during this phase. As the design of both these factors was purposely insufficiently modeled during analysis and design, they required much attention during the last parts of implementation. Inherently complex, the interaction, both amongst agents and through the domain, had to be trialed and tested rigorously before established. Hence, related to that explained in the last section, a challenge at this stage was to balance the complexity of interaction to avoid pitfalls like unexpected behavior, yet still experience some of the challenges involved for the purpose of this paper.

As mentioned, the prototype did not include much of the potential learning and pro-active aspects of agent systems. These characteristics are in many papers deemed as an intelligent feat of agents and often thereby involve complex operations. Hence, due to time constraints, the prototype did not focus on these areas.

However, some pro-active features were included to test the theories presented above. These behaviors are particularly related to the domain, or in Jade: the supervisor agent. As this agent continuously provides a list of services, other agents pro-actively act on the basis of some dynamic variables of the system. As an example, patient agents continuously look for available resources to meet their objectives; namely examination or treatment.

6.2 Measured Results

In addition to the experienced results presented in the previous section, the prototype also allowed for some common characteristics to be measured and tested directly. As we have seen, not only was this testing process an important phase in the development cycle - it also helped underline some relationships which will be discussed in the next chapter. The tests are presented in this section together with a presentation of its results. The practical approach of the tests is described in chapter four. Although giving some interesting results, please note that the technical limitations and/or accomplishments are not the focus of this paper.

6.2.1 Dynamic System

According to our definition in chapter two, a dynamic system supports adding and removing components. In our case, the components are the three types of agents: Patient agents, personnel agents and equipment agents. The dynamic processes are essential for any patient scheduling system or resource management system in general. In a medical institution new patients arrives continuously and leave when they finish their treatment. Personnel naturally have their various working hours which need to be accounted for, and equipment are changed

and renewed at a regular basis. The table below presents the testing of these processes based on some parameters, while the results are described below:

Action	Hypothesis	Results
Adding Patient Agent	New agents will be added or removed seamlessly. Trying to add an already existing or removing a non-existing agent will produce an error message, as will the entering of a wrong data type.	*Part Success*
Removing Patient Agent		*Success*
Adding Personnel Agent		*Part Success*
Removing Personnel Agent		*Success*
Adding Equipment Agent		*Part Success*
Removing Equipment Agent		*Success*

Figure 6.1: Test Results – Prototype Dynamiccs

Note that in the prototype, the maximum number of beds and offices are restricted for patients and personnel respectively. This restriction was established due to esthetics in the simulation presentation window and was therefore not tested for violations.

As all tests succeeded, the prototype can be deemed dynamic in that it can add and remove agents. However, the results labeled *part success* disclosed some limitations to the prototype. While the adding of non-existing agents gave successful results, the prototype does not account for any type violations or provide any error message when adding an agent that already exists. In fact, adding existing agents wrongfully assign them to the Jade platform. However, the agents do not participate in the calculations or graphic simulation deeming this error as an arbitrary coding problem.

6.2.2 Mobility

This next test is very much related to the previous one. But instead of testing the adding and removing of agents, we look at the transport and preservation of state; can the agents be transported to another system, e.g. another medical institution, preserving information and state? These are quite normal operations in a healthcare environment; patients are moved between institutions for different types of treatment while personnel and equipment are relocated as the need for expertise and application areas change. The results of these tests are presented below:

Action	Hypothesis	Results
Move Patient Agent to Other System	Agent can be moved efficiently, preserving state and not loosing any information when established at the new system.	*Success*
Move Personnel Agent to Other System		*Success*
Move Equipment Agent to Other System		*Part Success*

Figure 6.2: Test Results – Prototype Mobility

A number of assumptions were included in this test - both systems having the same underlying architecture and data structure being the most important. As the agents are designed on to work on the Jade platform and implemented in the Java programming language, the target system had to use the same platform and language to understand the received agents. Furthermore, while the state values of the agents are measurable and comparable numbers, their underlying semantics must be interpreted sensibly, adopting some

of the scheduling techniques in the original prototype. Also, the prototype holds a medical terminology which must have equivalent properties in the target system.

Again, the tests are relatively positive. Agents can be efficiently moved between similar systems, but the patient agents encountered some problems. While the agents could be moved from one platform to another, carrying their state and place in the examination cycle, they do not inhibit any historic data from their visit at the first institution. As explained in chapter five, this historic data is valuable in the calculations of patient utility. This is also however, a relatively simple implementation error which could easily be corrected by adding another history field to the patient agents.

7 Discussion

This chapter will investigate and discuss the results presented above. We will start by reintroducing the key development factors model first presented in chapter four. The chapter is further structured according to the three key factors identified in the above-mentioned model. Throughout these sections, the relationships hypotheses will be tested against the results, re-introducing some examples of functionalities, and hopefully add some interesting changes and viewpoints to the model.

7.1 Key Development Factors

We will in particular look at three distinct factors thought to influence and diverse agent-oriented development processes from more traditional approaches. As explained in chapter three, the development factors are related to the characteristics and behavioristic abilities of multi-agent systems.

Based on these findings, a hypothesis concerning these relationships was presented through the key development factors model in chapter four. The model is reintroduced below:

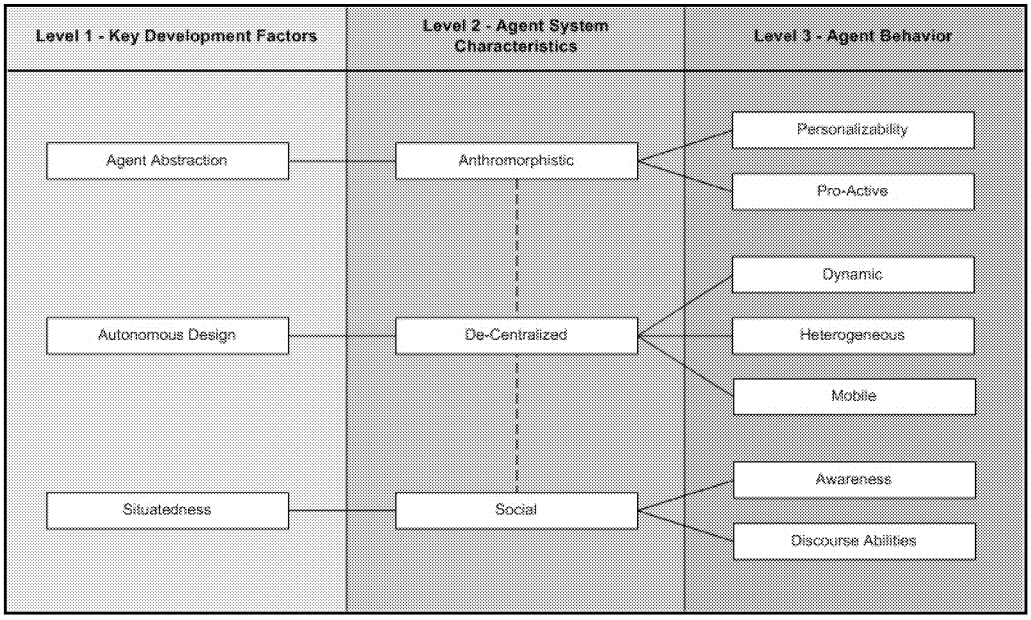


Figure 7.1: Key Development Factors and Relations

As a quick recap, the relationships are presented below, highlighting the various key terms in accordance with the findings in section 3.2. Each term is then discussed in detail in the following sections.

7.1.1 Key Factor 1 - Agent Abstraction

The *agent abstraction* factor is related to the *anthromorphistic* nature of agents. This characteristic denotes agents' emulation of human abilities like learning, reasoning and conversing. In multi-agent systems, this result in *pro-active* behavior and the ability to learn and adapt through *personalizable* behavior. Further, the anthromorphistic characteristic of agents can be related to their social ability which is presented below.

7.1.2 Key Factor 2 - Autonomous Design

The *autonomous design* factor is directly related to the *decentralized* structure of multi-agent systems. A key characteristic in agent system is the lack of a central control or process unit. Closely related is the *dynamic*, *heterogeneous* and *mobile* ability of multi-agent systems. These three terms denotes the ability of seamlessly adding and removing components (i.e. agents), allowing widely different types of components and uncompromising transport of components to other systems respectfully.

7.1.3 Key Factor 3 - Situatedness

Situatedness refers to designing the close relationship between an agent and its environment. This development factor is related to the *social* abilities of agents, in essence; its input and output. While the input should include *awareness* of environmental changes, the output must be sophisticated enough for *discourse abilities*. Clearly, as mentioned above, these terms are also related to anthromorphism, with especially discourse abilities representing a human feat.

7.2 Agent Abstraction

The presence of the agent abstraction factor was widely supported by the experiences during my prototype development, both through positive and negative aspects. As we have seen, this high level of abstraction helped in both the analysis and design phases by disregarding unneeded and irrelevant aspects. However, the high level of abstraction caused some problems during implementation, as some concepts were too abstract and revealed a lack in necessary technical details. This section will discuss these findings further, and draw lines to challenge the relationships proposed in the model below:

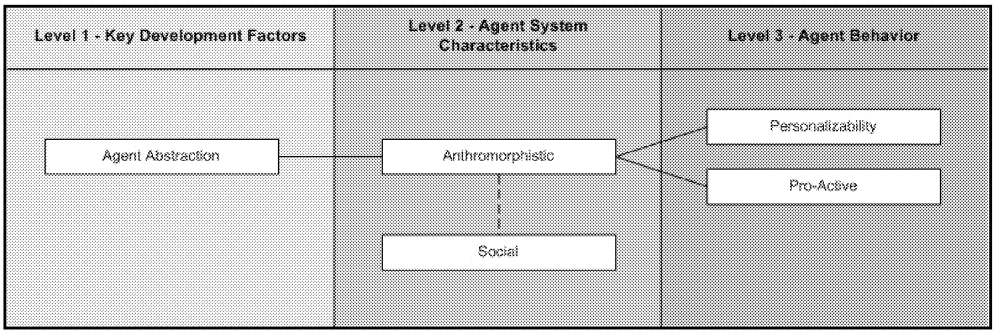


Figure 7.2: Key Development Factors and Relations - Agent Abstraction

7.2.1 Agent System Characteristics – Anthropomorphic and Social

First and foremost, the agent abstraction was useful due to its close relationship with anthropomorphic features. The agent abstraction involves the adoption of human qualities, which were found very useful when identifying and specifying agents during early parts of development. As the prototype consisted of patients and personnel, qualities like health condition and expertise could be mapped respectively. As seen in chapter three, this relationship is also supported by the agent research community. Wooldridge and Ciancarini (2001) call this relationship a natural metaphor:

> *"Just as the 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. For example, a scenario currently driving much R&D activity in the agents field is that of software agents that buy and sell goods via the Internet on*

> *behalf of some users. It is natural to view the software participants in such transactions as agents."*

This example can be related to my own experience. As the prototype were to be focused around personnel treating patients, there were similar negotiation techniques involved in this process as in the operations described above. As we explained in detail through chapter five, patients argue their scheduling position based on their health condition, while personnel argue their participation through expertise values.

Though these analogies were very useful during analysis and design, there were, as mentioned, some problems during implementation. The anthropomorphic entities represented highly abstract structures which I found somewhat problematic to implement on the Jade platform. While mental attributes like purpose and possible tasks were well defined, details like variable fields and specific functionalities had to be evaluated and established in this late phase. For example, the distribution patterns of necessary scheduling data between agents and the location of the optimization operations were not defined until early implementation. This problematic aspect of the agent abstraction and its anthropomorphic nature further validates its presence and relation in the key factor model.

To make decisions based on common goals, these agents also needs to exhibit some discourse and negotiation abilities, thereby underlining the anthromorphism – social relationship of our model. In the prototype, these social abilities were an important factor in negotiating the scheduling sequences of patients. As explained in chapter five, the argumentative data from patients, personnel and equipment had to be evaluated in parallel to reach an optimal solution. As such, there is a lot of interaction involved in these operations. During development the anthromorphistic nature of agents helped in assigning these social properties to suitable entities. For example, when designing patient and equipment agents, I knew very early on that patients would have to negotiate their diagnosis with relevance to available equipment.

Also, the need for discourse abilities in some aspects of the prototype helped establish an agent abstraction which would otherwise not have been discovered. As equipment agents do not naturally map to human qualities, they were early in the project thought to have a more passive role. A related point is also made in the literature. Foner (1993) argues that there are some agent systems where human-like features are not necessary:

> *"…there are several extent systems that might fit the "agent" mold that are clearly not anthropomorphic (e.g. mail-sorting programs that learn how to sort based on watching the user's actions and making inferences), while there are some that clearly are."*

Hence, many agent system development processes do not take this anthromorphistic approach, but rather choose agent entities based on classes of tasks or goals in the system. In environments where different classes of people performs the same tasks or have the same goals, this might be a more sensible approach. For example, if individual patients used widely different functions based on their condition or personnel used specialized types of data based on their field of expertise, it is easy to imagine that building agent structures based on goals and tasks would make more sense.

As such, it can be argued that approaching an agent abstraction through goals and tasks compromise the anthropomorphic characteristics of agent systems. However, these relations can be defended by using a wider definition of the term, as anthropomorphic properties can be seen as not derived from an entity itself but from its mental and behavioral abilities. Hence, the learning and practice involved in mail-sorting can easily be mapped to mental and behavioral abilities like learning and reasoning.

This brings us back to the specification of the non-humanistic equipment agents in the prototype. While it seems unnatural to assign mental properties like belief and goals to such a rather static object, I had to consider its purpose in the system. As touched upon above, equipment agents were largely involved in the scheduling and optimization techniques – just as the more human-like entities of patients and personnel. Following the argumentation above, defining a broader view to the anthropomorphism term, equipment agents needed to inhibit social qualities and goals to participate and argue in the before-mentioned optimization operations, and as a consequence eventually were established as agents.

7.2.2 Agent Behavior – Personalizability and Pro-activeness

Although the pro-active and personalizability factors were not required to figure prominently in the prototype, some simple examples of these properties were implemented simply to challenge the theories in this thesis. I found that these relationships were, for this very reason, harder to establish.

However, as described in chapter six, a full implementation of these aspects were not hard to imagine if a larger scope of functionalities would be desirable. Further elaborating on anthropomorphic abilities, examples of this could be support for personalizing patient agents implementing specific functions related to various diagnosis results, or pro-active search for medical attention at certain health conditions. It is clear that such scenarios always attach these behaviors to agent abstractions, and being anthropomorphic in nature, the relationships described above can be established.

In literature, both pro-activeness and personalizability are associated with anthropomorphic properties. Jennings and Wooldridge (2000) relates initiative and goal directed behavior to this factor:

> *"… agents are proactive - able to opportunistically adopt new goals and take the initiative in order to satisfy their design objectives."*

Drawing on this, the prototype could for example have included patients which more actively look for scheduling slots due to new goals or critical health conditions. For personnel or equipment agents, a pro-active functionality example could be to increase the search scope for available and relevant patients during times when the queue of patients to be treated is decreasing. As mentioned, these are only hypothetical functionalities, not included in the prototype due to time-constraints. I could easily have implemented such features however, given more time.

The personalizability factor is even more mapped to human-like behavior, as the term directly denotes an adoption of properties and abilities to simulate the behavior of some real-life representative. Foner (1993) explains the anthropomorphic relationship:

> *"The point of an agent is to enable people to do some task better. Since people don't all do the same tasks, and even those who share the same task do it in different ways, an agent must be educable in the task and hand and how to do it."*

In the prototype, these anthropomorphic features could be assigned to agents to further enhance their tailoring to its specific representative. For example, a patient agent could be designed to handle specific types of patients, evaluating health conditions and priorities differently according to diagnostic results. Personnel and equipment agents could have similar functionalities, tailoring for example heart surgeons to have explicit access to related equipment. On the same account as pro-activeness, personalizability were not implemented due to time constraints, but clearly, agent tailoring are highly applicable by refining agent abstractions further than simple agent, personnel and equipment divisions.

7.3 Autonomous Design

Autonomous design also posed both positive and negative impacts on my development project, supporting this key factor in the model. Especially during implementation, it was convenient to work with largely independent components focusing on specialized functionality and structure. By separating key concepts in the system, I found it easier to stay organized and focused on individual parts of the system, one at a time. However, as the autonomous agents were highly distinct in structure and functionality, interaction between them was increasingly difficult to model. This section will study these findings, also investigating the hypothetical relationships presented in the model below:

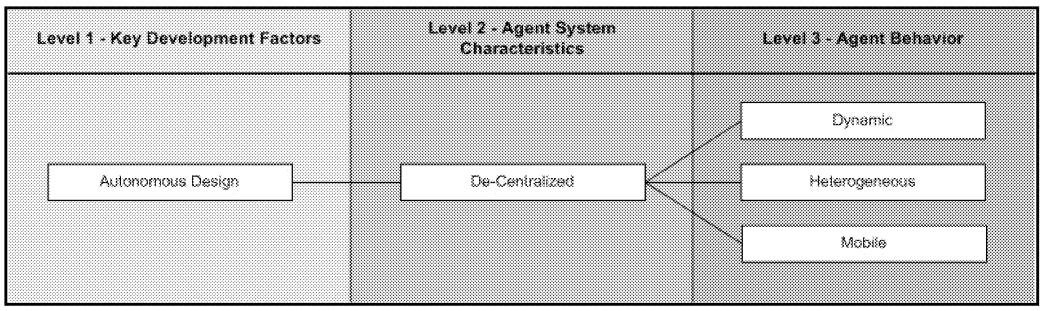


Figure 7.3: Key Development Factors and Relations - Autonomous Design

7.3.1 Agent System Characteristics - Decentralization

This divide-and-conquer approach is a well known tool for handling complexity and is recognized by many researchers. Jennings (2000) assigns this factor to the agent software development process:

> *"Decomposing a problem in such a way aids the process of engineering complex systems in two main ways. Firstly, it is simply a natural representation for complex systems that are invariably distributed and that invariably have multiple loci of control. This decentralization, in turn, reduces the system's control complexity and results in a lower degree of coupling between components."*

When implementing the prototype, this modular approach was very useful. As the three distinct agent types; patients, personnel and equipment was eventually decided upon, I found that working on their individual functionalities separately yet in parallel was highly beneficial. For example, as diagnostic medical data in patient agents were implemented - personnel and

equipment could be updated subsequently before tested with new functionalities. This ensured that the focus in implementation could shift from entity centric approaches; for example creating utility functions for patients, to system centric approaches; for example implementing ontological data across the three agents.

Having such highly individual entities did however put some new requirements on the interaction between them. To ensure that independent agents could be implemented and tested, I had to develop a highly general way of interacting, avoiding the explicit communication patterns found in many legacy systems. Based on the above-mentioned ontology, such as pre-defined utility values and medical conditions, the agent types could be individualistic as long as they followed the set standards in the ontology sets.

These experiences give evidence to the autonomous design – decentralization relationship, moving complexity from system operations to interaction between autonomous entities. Furthermore, Zambonelli et al (2003) puts the relationship in context with modern distributed systems:

> *"In multi agent systems, applications are designed and developed in terms of autonomous software entities (agents)…The autonomy of the application components reflects the decentralized nature of modern distributed systems and can be considered as the natural extension to the notions of modularity and encapsulation for systems that are owned by different stakeholders."*

Reflecting on the diversity of modern information systems, the authors argue that design of modern systems can especially profit on the divide-and-conquer approach. Many modern information systems connect a wide range of individual systems to better coordinate business strategies or share data. On a micro-level, this property can also be assigned to my own experiences. For example, the agents in the prototype have very different stakeholders according to their representatives. As touched upon earlier, patient and personnel agents can be tailored to represent certain types of individuals, based on health condition or expertise. In this context, such personalization is a valid analogy to the modern information system stakeholders described above.

Also, concerning the engineering process itself, there is often a wide array of competence within a development group where each resource can be used more efficiently if focusing on areas of expertise. As discussed, this aspect could not be experienced in my case study development, as there was only one developer. However, by working on modules in an interleaved fashion this advantage was not hard to imagine. For example, when designing the patient scheduling system, a developer with medical competence would be assigned to establishing and implementing an appropriate ontology, while a developer with optimization strategies and coordination as a field of expertise could handle the actual scheduling operations.

7.3.2 Agent Behavior – Heterogeneous, Dynamic and Mobile

The decentralized structures give rise to many agent specific properties and behaviors. As mentioned, the high level of autonomy allows a greater degree of individual specification of the various components, without affecting the main operations of the system. This heterogeneous factor is also a recurring term in the agent literature. Jennings (2000) elaborates on this individuality:

> *"[agents] have their own persistent thread of control (i.e., they are active) and they decide for themselves which actions they should perform at what time."*

Performing such specified actions underlines the heterogeneous nature of autonomous agents, allowing for highly specific functionalities based on roles in the systems. For example, I discovered that patient agents having certain types of health conditions didn't need to participate in scheduling negotiations concerning unrelated treatment slots. This aspect was not implemented but could quite easily have been incorporated using the personalizability approach described in section 7.2.2. These functionalities help establish the relationship between the decentralized structuring of agents and their heterogeneous properties.

Further, the autonomous characteristics of agent system engineering make the process of adding and removing system components during development more applicable. However, these properties do not only manifest themselves during the engineering phase, as described above. When the system is operational, during late testing and maintenance, these abilities ensure a higher degree of dynamics - or in essence: The adding and removing of agents.

As we have seen in chapter six, the testing of the prototype dynamics gave partly successful results. While the limitations revealed by these tests concerned some lack in type handling and error messages, it did give evidence to the basic dynamic feature of agent systems. The agents in the system being highly autonomous, I found that the adding and removal of patient, personnel and equipment agents easily could be implemented. Indeed, these features were very important in my case study application, as patients should be able to come and go based on their treatment cycles, while equipment and personnel should be able to share their expertise with other similar medical institutions.

This however, implied that the interaction had to be sophisticated enough to handle the general cooperation between components, not only explicit communication between agents. The acceptance of new agents within these general structures thereby gave evidence to the relationship with the decentralized structure of the prototype. This relationship also figures in the agent system literature. Zambonelli and Omicini (2004) relate the removal and adding of components to the property of autonomy:

> *"Autonomy…reduces the complexity of managing systems with a high and dynamically varying number of components."*

Note that the authors here focus on the management of the system, not the engineering aspects. A natural extension to this feat is moving components to other systems. As I had already implemented dynamic abilities, the practical adding and removal of agents were accounted for. However, the mobility factor is also concerned with the preservation of state information. As mentioned, patients should be able to move from one system to another, under certain assumptions, keeping valuable health information and state. In my prototype, the tests for this property were positive. A patient agent can for example do his/her examination in one system, whilst continue treatment in another, preferably obtaining a higher level of equipment and personnel expertise in the process. Consequently, this relates to the decentralization factor much in the same manner as that discussed for the dynamic features described above. The process is described by Lange and Oshima (1999):

> *"A mobile agent is not bound to the system on which it begins execution… Created in one execution environment, it can transport its state and code with it to another execution environment in the network, where it resumes execution."*

On a final note, as explained above, the issues with unpredictable behavior related to pro-activeness were not rigorously tested during development, but implications were easy to imagine. Most researchers see this as a possible challenge, but contrary to our model also relate this problem to autonomy. Weiss (2002) explains:

> *"Another difficulty of agent orientation results from its emphasis on autonomy…not handling autonomy carefully enough can lead to an unpredictability of the patterns and outcomes of interactions, and thus to undesirable emergent phenomena at the overall system level."*

I found evidence to the same relationship when implementing the different agents. While pro-activeness certainly is an anthropomorphic ability, it also became clear that these characteristics could not have been implemented to the same extent in a centralized system. By having loosely connected patients and personnel, patient agents could for example make decisions based exclusively on their own experiences. Such feats are harder to imagine if developing using a more traditional approach.

This new relation constitutes a change in the hypothesis model as pro-active behavior was originally exclusively linked to the anthropomorphic characteristic of agent system. The revised model is presented and explained in the summary section of this chapter.

7.4 Situatedness

Finally, a recurring factor when developing agent systems is the close relationship between agents and their environment or domain. As indicated by the hypothesis model, this factor is related to the social abilities of agents, including both the anthropomorphic adopted discourse abilities used amongst agents, and the close reactive relationship with the underlying domain. This section will study this hypothesis, investigating both prior research and my own experiences.

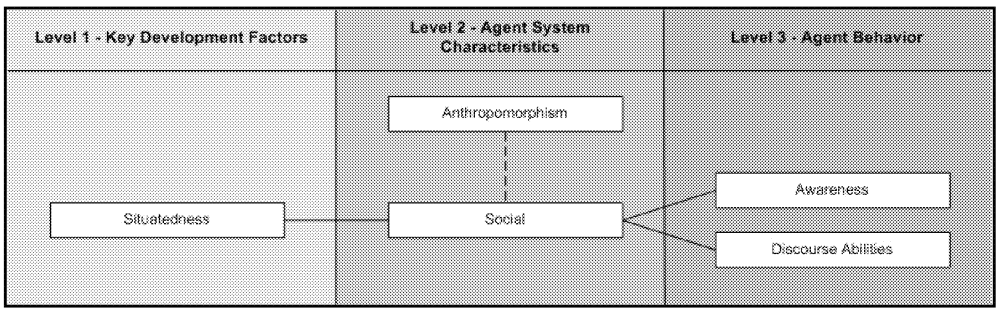


Figure 7.4: Key Development Factors and Relations - Situatedness

7.4.1 Agent System Characteristics – Social and Anthropomorphic

As explained in the previous chapter, the analysis and design phases did not require much modeling of this relationship as the implementation tool abstracted away environment variables through a supervisor or "yellow page" agent. Thus, these aspects could not be tested to the full extent. However, during implementation I had to consider many of the social ability

aspects to allow for interaction with this domain controlling agent. As big parts of governing the scheduling mechanisms were done through services in the environment, the communication with this agent had to be sophisticated enough to inhibit both awareness and discourse abilities. For example, personnel agents advertise their expertise through the yellow page agent, enabling patients and equipment to couple and initiate scheduling. As such, a relationship between social abilities and situatedness can be established.

In the agent research community the situatedness term is often related to mental and social characteristics, ensuring the interactive cooperation mentioned above. Jennings and Wooldridge (2000) describes this relationship:

> *"…agents are situated problem solvers: they have to act in pursuit of their objectives while maintaining an ongoing interaction with their environment."*

In my prototype, patient agents pursue their goals much in the same way; looking for available expertise to get examination and treatment slots. Furthermore, the social abilities are also related to the anthropomorphic nature of agents. By drawing on human-like abilities, agents can engage in complex interaction to solve goals. Wooldridge and Ciancarini (2001) describe this relationship:

> *"Agents do not simply act in response to their environment; they are able to exhibit goal-directed behavior by taking the initiative."*

This link between the agents' anthropomorphic abilities and their close contact with the environment were a balancing challenge during the testing stages of implementation. As discussed in chapter six, the analysis and design on and around these relations were purposely postponed to allow evaluation of some hard-to-predict aspects at later stages in the development. During this test- and develop phase I found that there was a clear difference in approach to the two forms of interaction; namely between agents and towards the environment. While social acts between agents were complex, handling most of the actual optimization, the interaction with the environment was less sophisticated, mostly serving as a look-up service and information central. This supports the hypothesis model, as increasingly complex interaction is intrinsically anthropomorphic. This can be seen as consequences of both the abstractive view on environment that Jade supply through yellow page agents and the relatively simple problem the case study of this thesis presented.

7.4.2 Agent Behavior – Awareness and Discourse Abilities

Drawing on the conclusions above, though fairly simple, the interaction with the environment did consist of some typical agent behaviors. As we have seen, agents continuously look for available services and therefore inhibit the awareness factor. Sauinier et al (2006) explains this term:

> *"Another important part of the solicitations of real-life situations come from other means than direct transmission, and are enabled by a particular state of the participants, the awareness."*

In my case study, these abilities could for example be related to the mobility and dynamic support described in section 7.3.2. As personnel agents are moved to a new medical institution, patient agents should be able to consider the new expertise offered. They are aware not only of the inclusion of a new agent, but also which services it can provide.

This leads us to how these services can be advertised. First, Zambonelli et al (2003) relates the discourse factor to the dynamic nature of agent systems and domains:

> *"…the high-level and dynamic nature of multi-agent interactions is appropriate to open systems in which the constituent components and their interaction patterns constantly change."*

The agents in the prototype needed to draw on some of the discourse abilities described above. However, in my experience, these properties did not feature heavily in the interaction with the environment, but rather amongst the negotiating agents. This is also suggested in the quote above, not mentioning environment in particular but rather the components of the system. Furthermore, the authors relate a more passive term to the communication with the environment; namely services:

> *"Most software systems are now de facto concurrent and distributed, and are expected to interact with components and exploit services that are dynamically found in the network."*

The service term suits the domain interactions of the prototype, as agents can advertise their functionality through the supervisor agent waiting for bidders to use their competence. As we have seen, this is particularly useful for personnel agents when moving to new systems. But clearly, we can also make use of these abilities when considering the equipment agents. Moving specialized equipment to appropriate institutions before advertising their services in the negotiation framework, they can also participate in the dynamic nature of the system.

This in mind, further changes were done to the hypothesis model to account for this discovery. Instead of relating discourse abilities to the environment, it makes more sense to relate these feats to the anthropomorphic characteristic of agent systems. All changes to the factor model are summarized in the next section.

7.5 Key Development Factors Revised

As a consequence of the findings presented in chapter six and the discussion presented over the last three sections of this chapter, the key development factors model have been revised and remodelled to fit the conclusions reached from this work. The final model is presented below:

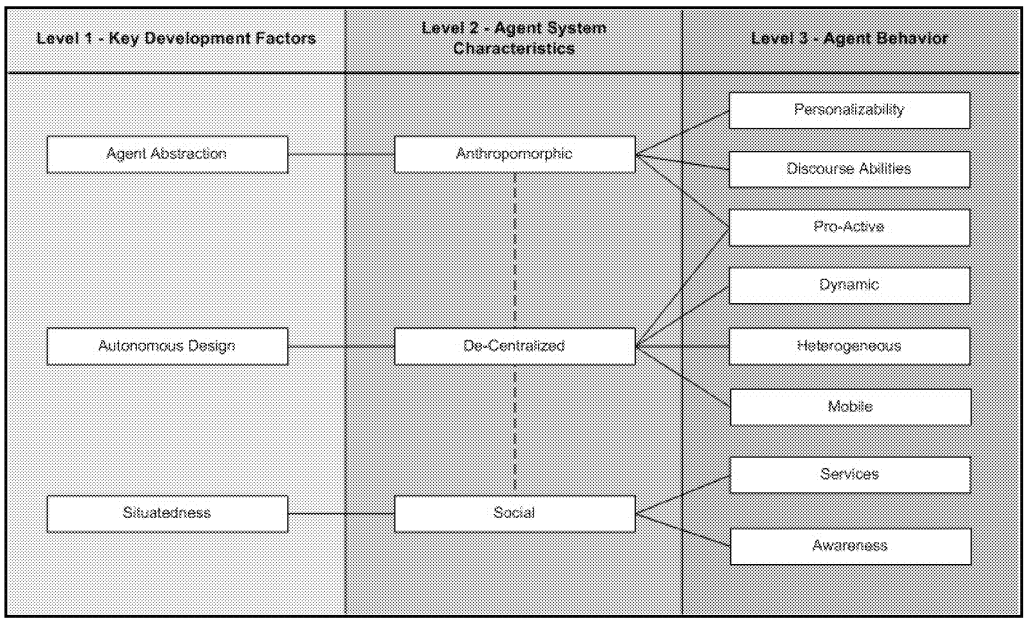


Figure 7.5: Revised Key Development Factors and Relations Model

Some changes have been done as a result of the discussion above. The changes are summarized below:

1. The service behaviour has been introduced in relation to the situatedness factor as the need for discourse abilities towards the yellow page agent was not at all necessary.
2. In consequence, the discourse abilities were found to have a stronger relationship to the anthropomorphic features of agent systems and were moved accordingly.
3. Pro-active behaviour has been included as a consequence of decentralization, as this agent behaviour is dependent on autonomous structures.

The next chapter will present these findings more thoroughly, highlighting some implications, limitations and possibilities of future work.

8 Conclusion

This thesis has studied the effects of using an agent oriented approach during development of a patient scheduling system for a medical institution. By looking at both positive and negative implications of using the agent approach in contrast to more traditional development paradigms, the thesis have attempted to investigate a yet unexplored area in the agent research community (e.g. Jennings, 2000; Luck et al, 2005). While there are many frameworks, methods and applications, there is a clear lack of validation and documentation around these tools and processes.

As such, this thesis has both added to the validation of agent applications and methods, and the documentation of experience in such development processes. By building a fully functional patient scheduling system, addressing common complex information system challenges like optimization strategies and dynamic behaviour, the thesis have attempted to answer the first research question:

1. *Given the ever-increasing size and complexity of modern information systems, can agent technology contribute in making the build- and maintenance processes of such systems feasible?*

During this agent oriented engineering process, experiences was gathered and documented for later evaluation. This gave rise to answering the second research question:

2. *What are the significant positive and negative aspects when building such type of systems using the agent approach and how do they relate to agent and agent system characteristics?*

To obtain a comprehensive answer to this second question, a hypotheses model was derived from theories and documented experiences in prior research, attempting to identify the most common factors from the agent oriented engineering process and related agent and agent system characteristics. The resulting hypotheses model is presented below:

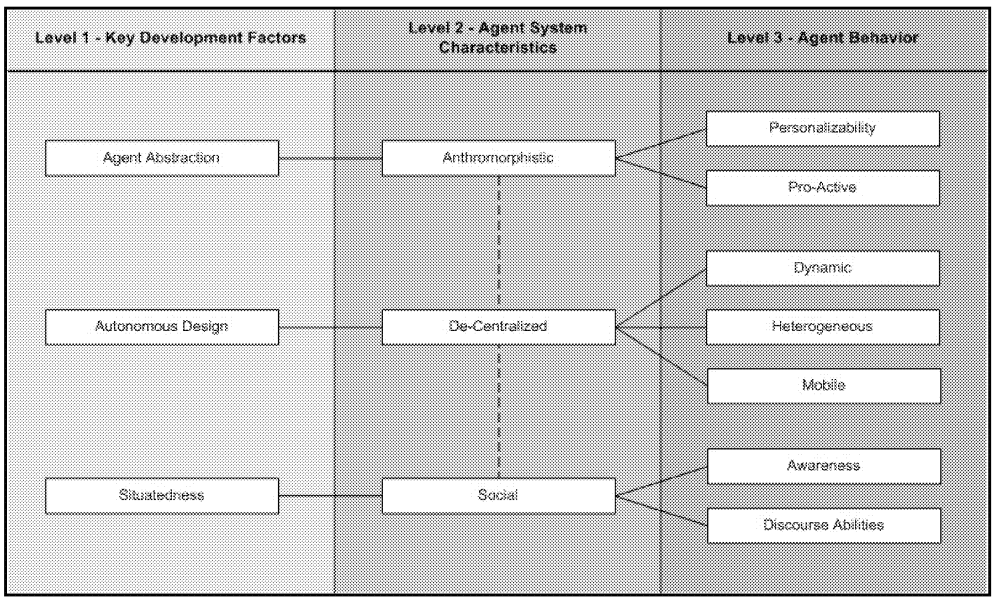


Figure 8.1: Key Development Factors Hypotheses Model

This literature study resulted in the identification of three distinct agent oriented development factors: *Agent abstraction*, *autonomous design* and *situatedness*. Further, these factors were found to be strongly related to three agent system characteristics: *Anthropomorphism*, *decentralization* and *social abilities*. The relation between factors in development and characteristics in agent systems could also be drawn to the agent behaviour level. Seven behavioural abilities were identified in this context: *Personalizability*, *pro-activeness*, *dynamic*, *heterogeneous*, *mobile*, *awareness* and *discourse*.

This work was done in tandem with the prototype development. Apart from addressing the first research question, this phase of the project resulted in a functional prototype from which both experience and measurable results could be drawn. When both the hypotheses model and the prototype was done, knowledge from prior research and own experiences could be compared and discussed.

This comparison resulted in three distinct changes to the hypotheses model. First, because of the yellow page agent structure of the Jade development platform, I discovered that there were no use for discourse activities between agents and the environment. As the prototype was focused on optimization amongst sets of agents, the discourse abilities instead became apparent between negotiating agents. As such, the discourse behaviour of agents was instead related to the anthropomorphic nature of agent systems – a relation which also had some support in the literature.

Furthermore, as this yellow page agent function as a blackboard listing available services in the domain, the interaction with the environment were found to consist of two major parts: Input in the form of advertising services on the platform and output in the form of awareness and ordering of available services. As such, the advertising services behaviour was added to the social characteristic and situatedness factor of agent systems.

Lastly, while the pro-active behaviour clearly was identified as an anthropomorphic property, I also found that this behaviour was dependent on a decentralised structure and should as such also be related to this agent system characteristic. This relationship was also present in some of the literature, and consequently, new changes were done in the hypotheses model. The final revised model with all factors, characteristics, behaviours and relations is presented below:

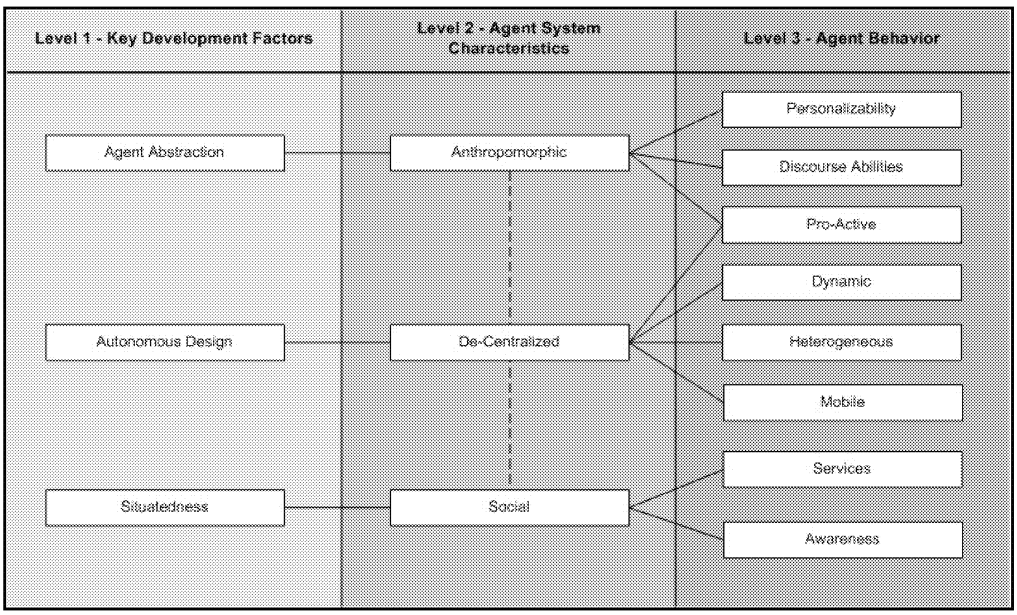


Figure 8.2: Key Development Factors and Relation Model Revised

With these changes, the terms and relations presented in this model should function as a valid contribution to a relatively unexplored area within the agent research community. Hopefully these humble suggestions could help better understand the differences between agent oriented development and more traditional approaches, taking both positive and negative aspects into account. However, this is by far an exhaustive investigation of all such differences and factors. Hence, through the last two sections of this thesis we will look at limitations and possible further work within this research field.

8.1 Limitations and Further Research

As mentioned in chapter four, student projects are always limited in terms of time (Remenyi et al, 2003). This general constraint posed natural restrictions to the scope of the project. For example, the prototype could have included more functionality than simple patient scheduling, grasping instead all general workflows in a medical institution. Examples are personnel work schedules, patient monitoring and sharing of medical records.

Furthermore, by including more functionality related to medical care, the prototype could have accounted for several more agent oriented development factors through agent system characteristics and behaviors. This could include for example the sharing of global or regional data using agent transport and protocols, measuring of workflow efficiency and seamless agent transfer between institutions.

Also, the time constraints put restrictions on the scope of the literature study. As mentioned, there are a vast number of contributions to various methods and frameworks. While chapter three gives a good overview, this presentation is by no means an exhaustive presentation of this field. It is impossible to know whether a more extensive literature study would have suggested some more terms to our hypotheses model, but it is clear that to think of the existing relations as exclusive would be rather naïve.

On a more realistic level, the prototype could have included some specific functionality which would have contributed in validating the model further. First, there is a clear lack in pro-activeness in the patient scheduling process. As explained in the discussion, some pro-active

behaviour was implemented simply for the cause of this validation, but due to time constraints this functionality is not very extensive nor is it contributing noteworthy to the operations of the prototype.

The same could be said about the personalizability behaviour. While patients, personnel and equipment agents obviously are personalized in some way to represent their distinct live entities, no further refinement were done to personalize these agents to for example specific types of patients based on conditions or specific types of personnel based on expertise. Such features would both enlarge the scope of the prototype as well as further clarify the personalizability behaviour in relation to the agent abstraction factor in our model. This functionality aspect is further discussed in chapter seven.

Aside from the time constraints, the project also suffered somewhat from the novel approach that agent orientation software development represents. First, the project taking place in a relatively small town, there were no commercial or industrial cooperation possibilities. Needless to say, this would have enhanced the validation of this study, as more extensive testing could have taken place, both in terms of functionality of the prototype, and gaining valuable feedback on the identified development factors and related agent characteristics.

This paradigm novelty was also apparent in the academic institution in which this thesis originated. While there were extensive competence within software development and research processes in general, there were no specific expertise within the agent and agent system fields. Again, it is difficult to pinpoint the implications of this, but one should think that especially the literature study and development of the research questions could have drawn on some feedback in terms of their direction and scope.

References

Bartelt, A., Lamersdorf, W., Paulussen, T. O. and Heinzl, A. (2002). *Agent Oriented Specification for Patient-Scheduling Systems in Hospitals*. Proceedings of IFBI, Dagstuhl, Saarland, Germany, May 2002.

Belecheanu, R. (2005a). *Software Agents in International Insurance – Acklin and Interpolis*. AgentLink Case Study, November, 2005.

Belecheanu, R. (2005b). *Agent-Based Factory Modelling – Eurobios and SCA Packaging*. AgentLink Case Study, November, 2005.

Bellifemine, F., Caire, G., Poggi, A. and Rimassa, G. (2003). *Jade – A White Paper*. Search of Innovation, vol. 3(3).

Bellifemine, F., Poggi, A. and Giovanni, R. (2000). *Developing Multi-agent Systems with JADE.* Intelligent Agents VII: Agent Theories Architectures and Languages: 7th International Workshop, Boston, Massachusetts, United States, July 2000.

Bernon, C., Cossentino, M. and Pavón, J. (2005). *An Overview of Current Trends in European AOSE Research.* Informatica, vol. 29, pp. 409-421.

Bratman, M. E., Israel, D. J. and Pollack, M. E. (1988). *Plans and Resource-Bounded Practical Reasoning.* Computational Intelligence, vol. 4 (4), pp. 349-355.

Brazier, F., Keplicz, B. D., Jennings, N. R. and Treur, J. (1995). *Formal specification of multi-agent systems: A real-world case.* In Proceedings of the International Conference on Multi-Agent Systems, San Francisco, California, United States.

Brazier, F., Keplicz, B. D., Jennings, N. R. and Treur, J. (1997). *DESIRE: Modelling Multi Agent Systems in a Compositional Formal Framework.* International Journal of Cooperative Information Systems, vol. 6, pp. 67-94.

Bresciani, P., Giorgini, P., Giunchiglia, F., Mylopoulos, J. and Perini, A. (2002). *TROPOS: An agent-oriented software development methodology*. Technical Report DIT-02-0015, Department of Information and Communication Technology, University of Trento, Trento, Italy.

Brooks, R. A. (1986). *A robust layered control system for a mobile robot.* Journal of Robotics and Automation, vol. 2 (1), pp. 14–23.

Brooks, R. A. (1990). *Elephants Don't Play Chess.* Robotics and Autonomous Systems, vol. 6, pp. 3-15.

Caire, G., Coulier, W., Garijo, F., Gomez, J., Pavon, J., Leal, F., Chainho, P., Kearney, P., Stark, J., Evans, R. and Massonet, P. (2001). *Agent-oriented Analysis Using Message/UML.* In Proceedings of the 2nd International Workshop on Agent-Oriented

Software Engineering. Lecture Notes in Computer Science, vol. 2222, Springer-Verlag, New York, United States, pp. 119–135.

Chaib-Draa, N. and Dignum, F. (2002). *Trends in agent communication language*. Computer Intelligence, vol. 2 (5), pp. 89–101.

Cliff, D. and Bruten, J. (1997). *Minimal-intelligence agents for bargaining behaviors in market-based environments*. Technical Report Hewlett Packard Labs.

Dastani, M., Hulstijn, J., Dignum, F. and Meyer, J. (2004). *Issues in multiagent system development*. In Proceedings of the Third Conference on Autonomous Agents and Multi-agent Systems, New York, United States.

Davidsson, P. and Wernstedt, F. (2002). *A multi-agent system architecture for coordination of just-in-time production and distribution*. Proceedings of SAC 2002, Spain, pp. 294-299.

Davies, W. and Edwards, P. (1994). *Agent-K: An Integration of AOP and KQML*. In Labrou, Y. and Finin, T. (Eds.), CIKM'94 Intelligent Agents Workshop.

Decker, K. S. and Lesser, V. R. (1993). *Generalizing the partial global planning algorithm.* International Journal of Intelligent and Cooperative Information Systems, vol. 1 (2), pp. 319–346, June 1993.

Decker, K. S. and Li, J. (1998) *Coordinated Hospital Patient Scheduling*, Department of Computer Science, University of Delaware, Technical Report ’98, vol. 12.

Decker, K. S., Sycara, K. and Williamson, M. (1997). *Middle-agents for the internet.* In Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence, Nagoya, Japan.

DeLoach, S. A., Wood, M. F. and Sparkman, C. H. (2001). *Multiagent Systems Engineering*. International Journal of Software Engineering and Knowledge Engineering, vol. 11 (3), pp. 231-258.

Dick, B. (1997). *Action Learning and Action Research*. Available online: www.scu.edu.au/schools/gcm/ar/arp/actlearn.html

Durfee, E. H. and Lesser, V. R. (1991). *Partial global planning: A coordination framework for distributed hypothesis formation*. IEEE Transactions on Systems, Management and Cybernetics, vol. 21 (5), pp.1167–1183, September 1991.

Finin, T., Weber, J., Wiederhold, G. and Genesereth, M. (1994). *Specification of the KQML Agent-Communication Language*. The DARPA Knowledge Sharing Initiative External Interfaces Working Group, University of Toronto, Canada.

Foner, L. (1993). *What’s An Agent, Anyway? A Sociological Case Study*. Agents Memo 93-01, M.I.T. Media Lab., Massachusetts Institute of Technology, Cambridge, Massachusetts, United States, May 1993.

Foundation for Intelligent Physical Agents (FIPA). (2002). *FIPA ACL Message Structure Specification*. Available online: http://www.fipa.org/specs/fipa00061/SC00061G.pdf.

Gjerstad, S. and Dickhaut, J. (1998). *Price Formation in Double Auctions*. Games and Economic Behavior, vol. 22, pp 1-29.

Giunchiglia, F., Mylopoulos, J. and Perini, A. (2002). *The Tropos software development methodology: Processes, Models and Diagrams.* In Third International Workshop on Agent-Oriented Software Engineering, July 2002.

Gomez-Sanz, J. J. and Pavon, J. (2002). *Agent-oriented software engineering with INGENIAS.* In Proceedings of the 3rd Central and Eastern Europe Conference on Multiagent Systems, vol. 2691, Springer-Verlag, pp. 394–403.

Grosz, B. and Kraus, S. (1995). *Collaborative plans for complex group actions.* Artificial Intelligence, vol. 86 (2), pp. 269–358.

Guo, M., Wagner, M. and West, C. E. (2004). *Outpatient Clinic Scheduling – A Simulation Approach*, Proceedings of the 2004 Winter Simulation Conference, Ingalls, R. D., Rossetti, M. D., Smith, J. S. and Peters B. A. (eds.), pp. 1981-1987

Haux, R. (2004). *Hospital Information Systems - Past, Present and Future*. International Journal of Medical Informatics, Sep. 2005.

Heeks, R. (2005). *Health Information Systems: Failure, Success and Improvisation*. International Journal of Medical Informatics, vol. 75 (2), pp. 125-137.

Hewitt, C., Bishop, P. and Steiger, A. (1973). *A Universal Modular Actor Formalism for Artificial Intelligence,* International Joint Conferences on Artificial Intelligence 3, International Artificial Intelligence Center, Menlo Park, California, United States, pp. 235-245, August 1973.

Horling, B., Lesser, V., Vincent, R., Wagner, T., Raja, A., Zhang, S., Decker, K. and Garvey, A. (2005). *The TÆMS white paper*, Technical Notes of Multi-Agent Systems Lab, Department of Computer Science, University of Massachusetts, United States.

Howden, N., Ronnquist, R., Hodgson, A. and Lucas, A. (2001). *JACK Intelligent Agents: Summary of an agent infrastructure*. Proceedings in T. Wagner and O. Rana (eds.), the 5th International Conference on Autonomous Agents, Workshop on Infrastructure for Agents, MAS and Scalable MAS, pp. 251 – 257.

Iglesias, C. A., Garijo, M. and Gonzalez, J. C. (1998). *A survey of agent oriented methodologies*. In Miller, J. P., Singh, M. P. and Rae, A. S., (eds.), Intelligent Agents V - Proceedings of the Fifth International Workshop on Agent Theories, Architectures, and Languages, Lecture Notes in Artificial Intelligence. Springer-Verlag, Heidelberg, Germany.

Jennings, N. R. (1993). *Commitments and conventions: The foundation of coordination in*

multi-agent systems. The Knowledge Engineering Review, vol. 8 (3), pp. 223–250.

Jennings, N. R. (2000). *On Agent-Based Software Engineering*. Artificial Intelligence, vol. 117 (2), pp. 277-296.

Jennings, N. R. and Wooldridge, M. (2000) *Agent-oriented software engineering*, Bradshaw, J. (Ed.), Handbook of Agent Technology, AAAI/MIT Press.

Jennings, N. R., Faratin, P. Lomuscio, A. R., Parsons, S., Wooldridge, M. and Sierra, C. (2001). *Automated Negotiation: Prospects, Methods and Challenges*. Group Decision and Negotiation, vol. 10, pp. 199-215.

Jennings, N. R., Faratin, P., Johnson, M. J., Norman, T. J., O'Brien, P. and Wiegand, M. E. (1996). *Agent-based business process management.* International Journal of Cooperative Information Systems, vol. 5 (2-3), pp. 105–130.

Jennings, N. R., Sycara, K., and Wooldridge, M. (1998). *A Roadmap of Agent Research and Development.* Autonomous Agents and Multi-Agent Systems, 1 (1), 7-38.

Kinny, D. and Georgeff, M. (1996). *A Methodology and Modeling Technique for Systems of BDI Agents, Agents Breaking Away*, Proceedings of the Seventh European Workshop on Modeling Autonomous Agents in a Multi-Agent World, Springer-Verlag, Berlin, Germany, January 1996.

Knapik, M. and Johnson, J. (1998). *Developing Intelligent Agents for Distributed Systems*. McGraw-Hill, London, United Kingdom.

Kumar, M. (2002). *Contrast and comparison of five major Agent Oriented Software Engineering (AOSE) methodologies*, available online: http://students.jmc.ksu.edu/grad/madhukar/www/professional/aosepaper.pdf

Labrou, Y., Finin, T. and Peng Y. (1999). *The Current Landscape of Agent Communication Languages*. Intelligent Systems, vol. 14(2).

Lange, D. B. and Oshima, M. (1999). *Seven good reasons for mobile agents*. Communications of the ACM, vol. 42 (3). pp. 88–89, March 1999.

Luck, M., McBurney, P. and Priest, C. (2004). *A Manifesto for Agent Technology – Towards Next Generation Computing*. Special Issue of Autonomous Agents and Multi-agent Systems, vol. 9. pp. 253–283.

Luck, M., McBurney, P., Shehory, O. and Willmott, S. (2005). *Agent Technology Roadmap: A Roadmap for Agent Based Computing.* AgentLink, Electronics and Computer Science, University of Southampton, United Kingdom. Available online: http://eprints.ecs.soton.ac.uk/11788/01/al3roadmap.pdf

Manansang, H. and Helm, J. (1996). *An Online Simulation-Based Patient Scheduling System.* In Proceedings of the 1996 Winter Simulation Conference.

Mayfield, J., Labrou, Y. and Finin, T. (1996). *Evaluating KQML as an agent communication language.* Wooldridge, M., Müller, J. P. and Tambe, M., (eds.), Intelligent Agents II, vol. 1037, Springer-Verlag, Berlin, Germany, pp. 347–360.

Nash, W. (1950). *Equilibrium Points in n-Person Games*. Proceedings of the National Academy of Sciences of the United States of America, vol. 36, pp. 48-49.

Odell, J., Van Dyke Parunak, H. and Bock, C. (2001). *Extending UML for Agents.* In Proceedings of the 1st International Workshop on Agent-Oriented Software Engineering. Lecture Notes in Computer Science, vol. 1957. Springer-Verlag, New York, United States, pp. 121–140.

Padgham, L. and Winiko, M. (2002). *Prometheus: A methodology for developing intelligent agents.* In Third International Workshop on Agent-Oriented Software Engineering, July 2002.

Payton, D., Daily, M., and Martin, K. (1999). *Dynamic Collaborator Discovery in Information Intensive Environments*. ACM Computing Surveys, vol. 31 (2), June 1999.

Phillips, D. T., Ravindran, A. and Solberg, J. (1976). *Operations Research: Principles and Practice*. John Wiley & Sons. Canada.

Rebollo, M., Julian, V., Carrascosa, C. and Botti, V. (2000). *A MAS approach for port container terminal management*. In Proceedings of the 3rd Iberoamerican workshop on DAI and MAS, Atibaia, Sao Paulo, Brasil, pp 83–94.

Reichertz, P. L. (1984). *Hospital Information Systems – Past, Present and Future*. In Proceedings of Fifth Congress of the European Federation for Medical Informatics, Brussels, Belgium, September 10-13 1984.

Reed, C., Boswell, B. and Neville, R. (2005). *Multi-agent patient representation in primary care*. In 10th Conference on Artificial Intelligence in Medicine, Aberdeen, Scotland, pp. 375-384, July 2005.

Reich, Y. (1995). *The study of design research methodology*. Journal of Mechanical Design, vol. 117, pp. 211–214.

Remenyi, D., Williams, B., Money, A. and Swartz, E. (1998). *Doing Research in Business and Management: An Introduction to Process and Method*, Sage, London.

Rogers, E. M. (1995). *Diffusion of Innovations.* 4th edition, Free Press, New York, United States.

Rosenschein, J. S. (1985). *Rational Interaction: Cooperation Among Intelligent Agents.* Ph.D. thesis, Computer Science Department, Stanford University, Stanford, California, United States, October 1985.

Russel, S. and Norvig, P. (2003). *Artificial Intelligence*. Prentice-Hall, London, United Kingdom.

Saunier, J., Balbo, F. and Badeig, F. (2006). *Environment as Active Support of Interaction*. In Proceedings of the Third International Workshop on Environments for Multi-Agent Systems, May 8th, 2006.

Shoham, Y. (1993). *Agent-oriented programming.* Artificial Intelligence, vol. 60 (1), pp. 51-92.

Smith, R. (1980). *The contract net protocol: High-level communication and control in a distributed problem solver.* Proceedings of the 1st International Conference on Distributed Computing Systems. Institute of Electrical and Electronics Engineering, New York, pp. 185-192.

Sycara, K. P. (1998). *Multiagent Systems.* AI Magazine, vol. 19 (2), pp. 79-92.

Szolovits, P., Doyle, J. and Long, W. J. (1994). *Guardian Angel: Patient-Centered Health Information Systems*. Technical Report MIT/LCS/TR-604.

Tambe, M. (1997). *Towards flexible teamwork.* Journal of Artificial Intelligence Research, vol. 7, pp. 83–124, September 1997.

Tesauro, G. and Das, R. (2001). *High-performance bidding agents for the continuous double auction*. In Proceedings of IJCAI-01 Workshop on Economic Agents, Models and Mechanisms, Seattle, United States, pp. 42–51, August 2001.

Van Dyke Parunak, H. (1987). *Manufacturing experience with the contract net.* Huhns, M. (ed.), Distributed Artificial Intelligence. Pitman Publishing, London, United Kingdom and Morgan Kaufmann, San Mateo, California, United States, pp. 285–310.

Van Dyke Parunak, H. (2000). *A Practitioner's Review of Industrial Agent Applications*. Autonomous Agents and Multi-Agent Systems, vol. 3 (4), pp. 389-407, December 2000.

Von Neumann, J. and Morgenstern, O. (1944), *Theory of games and economic behavior*, Princeton University press.

Vlassis, N. (2003). *A concise introduction to multiagent systems and distributed AI.* Informatics Institute, University of Amsterdam, Amsterdam, The Netherlands, Sept. 2003. Available online: http://staff.science.uva.nl/~vlassis/cimasdai/cimasdai.pdf

Weiss, G. (2002). *Agent orientation in software engineering*, Knowledge Engineering Review, vol. 16 (4), pp. 349–373.

Weyns, D., Steegmans, E. and Holvoet, T. (2004). *Towards Active Perception in Situated Multi-agent Systems*. Journal on Applied Artificial Intelligence, vol. 18 (9-10).

Wooldridge, M. and Ciancarini, P. (2001). *Agent-Oriented Software Engineering: The State*

of the Art. In Agent-Oriented Software Engineering. Ciancarini, P. and Wooldridge, M. (eds), Springer-Verlag Lecture Notes in AI, vol. 1957.

Wooldridge, M. and Jennings, N. R. (1995). *Intelligent agents: Theory and Practice*. The Knowledge Engineering Review, vol. 10 (2), pp. 115–152.

Wooldridge, M. and Jennings, N. R. (1998). *Pitfalls of agent-oriented development*. In Proceedings of Second International Conference on Autonomous Agents, Minneapolis/St. Paul, Minnesota, United States, pp. 385.391, May 1998

Wooldridge, M. and Parsons, S. (2003). *The BDI Model*. Rational Action Lecture, University of Liverpool, Liverpool, United Kingdom.

Zambonelli, F. and Omicini, A. (2004). *Challenges and research directions in agent-oriented software engineering.* Journal of Autonomous Agents and Multiagent Systems, vol. 9 (3), pp. 253–283.

Zambonelli, F., Jennings, N. and Wooldridge, M. (2003). *Developing multiagent systems: The gaia methodology.* ACM Transactions on Software Engineering and Methodology, 12 (3), pp. 317–370.

Zeng, D. and Sycara, K. (1997). *Benefits of learning in negotiation.* In Proceedings of AAAI-97, Providence, Rhode Island, United States.