Multi Agent Systems

Relating Artificial Intelligence Concepts and Information Technology Models in Competitive Agent-Based Applications

Ph.D. Dissertation

Gianfranco Pedone

Supervisor: László Zsolt Varga, Ph.D

Faculty of Informatics (IK),
Eötvös Loránd University (ELTE)

Doctoral School of Computer Science,
Information Systems Ph.D. Program,
Chairman: Prof. János Demetrovics, Member of HAS

Computer and Automation Research Institute (SZTAKI),
Hungarian Academy of Sciences (HAS, MTA)

Budapest, Hungary, 2011
To my mother and my father
We have been parted so unexpectedly and painfully,
but you gave me all I need
I won’t forget
“Try not to become a man of success but rather to become a man of value”  
(*Albert Einstein*)

“It is a miracle that curiosity survives formal education. The important thing is not to stop questioning. Curiosity has its own reason for existing [...]. Never lose a holy curiosity”  
(*Albert Einstein*)

“And I answered: 'As yet hath my word not removed mountains, and what I have spoken hath not reached man. I went, indeed, unto men, but not yet have I attained unto them’”  
(*Friedrich Nietzsche*)

“The secret of genius is to carry the spirit of childhood into maturity”  
(*Thomas Huxley*)
Declaration

Herewith I confirm that all of the research described in this dissertation is my own original work and expressed in my own words.

Any uses made within it of works of other authors in any form, e.g., ideas, figures, text, tables, are properly indicated through the application of citations and references.

I also declare that no part of the dissertation has been submitted for any other degree - either from the Eötvös Loránd University or another institution.

Gianfranco Pedone
Budapest, March 2011
What makes a man a man?

Is it the ability he has to survive? I don’t think so, as all living creatures have it. Is it his attitude to invent and discover? Neither, I think, as we feel the need to invent because we’re not evolved enough, and through discoveries we acknowledge only about principles and balances which have been governing us from time immemorial. So then, what makes a man a man? Is it the intelligence he has? Maybe, provided we are able to univocally define what intelligence is. All living creatures, in their own way, are capable of evidencing intelligence. So, what must a man have in order to be defined as such?

I saw men crying for the death of their beloved and others rejoicing for the birth of their sons.

I saw men helping the weak with faith and humility, and others fighting the wicked with determination and ostentation.

I saw great men proud of their childhood spirit and others despairing for the loss of it. I saw men sealed with empathy and courage perceiving others’ emotions and fears. I know of men who would sacrifice their lives for others without asking anything in exchange but love and comprehension.

I saw men teaching their sons justice and respect, love and kindness, hoping they’ll rescue this world.

This is what, in the very end, makes a man a man, I guess. In this sense, one day, I hope I’ll be a man, too.
Acknowledgments

Although this dissertation is my own work, I received inspiring support also from my colleagues and family. Without them, this manuscript would not be the same. I would like to express my gratitude here to all the persons who helped and encouraged me during my studies.

First of all, I want to thank those people that had a direct influence on the start of my PhD and offered me the opportunity of a suitable context for my research work.

These include first and foremost ELTE University’s “Professore” István Gyurján and my supportive supervisor Dr. László Zsolt Varga.

I am grateful for the fatherly encouragement I received from Professor András Benczúr at the very beginning of my PhD program and exams, at the Information System Department of Eötvös Loránd University’s (ELTE) Faculty of Informatics (IK).

I greatly acknowledge also the contribution and help of Dr. Péter Inzelt, director of Computer and Automation Research Institute (SZTAKI), where I performed the research work presented in this dissertation and had the chance to collaborate with outstanding people.

I am humbly and extremely grateful to Professor Pasquale Maviglio, wherever he might be now, for his determination and humanity in donating me again the conditions for an “independent” life.

Furthermore, I warmly thank Pablo Dorronsoro Morales and Ákos Hajnal for their fruitful and hilarious discussions on human existence and its apparent advancements in technological field.
I am very thankful for the support and encouragement of my “father-in-life”, Miklós, and all my family, especially my wife Eszter, for her continuous and abiding patience and comprehension.

Last but not least, I do wish to thank my three angels, Sofía, Leonardo and Giulia, who have appeared in my life to lighten the darkness along my path.
Abstract

In today's business scenarios information became the most valuable and strategic asset in an organization. Accessing the right information in the right moment can give a company the market-leading position. But in order to acquire such a capability, an organization is asked to master information, its management and the knowledge deriving from it.

The whirling circulation of information we are facing nowadays is being generated by the exploitation of computer systems in all contexts of our society: from home to military applications, to cite only some extreme examples. Such a huge amount of available information has caused, unexpectedly but inevitably, the curious side-effect of decisional incapability in decision makers. Indirectly, the possibility to access so much information at any time has led individuals with business responsibility (and not exclusively them) to be in such situations which would be rather addressable to the lack of information, on the contrary.

What decision makers would really need is the possibility to delegate to technology part of the decision making process' load. In compliance with the previous considerations, the research work I present in this manuscript aims at demonstrating how the adoption of agent-based technology can help computer science experts design intelligent systems for the provision of critical, long-term, knowledge-intensive, dynamically adaptive services. Nevertheless, it will be demonstrated how agent-technology can help an organization reach, under the proper conditions, a competitive advantage in
its application domain. My agent-focused research activities contributed to provide both theoretical and practical results in the application of the agent paradigm in complex domains modelling.

In particular, in the first thesis I formulated (T1.1 in paragraph 1.6.1) I reorganize and present conceptualization aspects of the agent-paradigm, the opportunity given by the latter to enable a natural description of system’s application knowledge, as well as its peculiarity to permit the embodiment of complex knowledge representations. The second thesis (T1.2 in paragraph 1.6.1) illustrates how agent-based software can gracefully adapt to long-term negotiations and continuous environmental changes, all crucial elements for a project in the achievement of its committed goals.

In real development cases the adoption of an engineering approach in the realization of agent-based industry-oriented applications is strictly advisable in order to guarantee extensibility and adaptability to the system.

According to this premise, my third contribution (T2.1 in paragraph 1.6.2) evidences how the automation of agent-oriented code generation can provide agent-based systems’ fast prototyping process both with a significant boost and much higher semantics. My fourth thesis (T2.2 in paragraph 1.6.2), in conclusion, aims at presenting a fundamental aspect of the agent technology: agent-paradigm can enable formal extensions of actors’ behavioural models, permitting to recognize, manage and embody new emerging application knowledge.

All the principles formulated in this manuscript have been practically applied to, and verified on, a real multi agent-based platform, developed in compliance with the specifications of K4Care European project (presented in the followings), whose objective was to provided all the participant countries with an agent-centred platform assuring long-term home care services.
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>DECLARATION</td>
<td>3</td>
</tr>
<tr>
<td>PERSONAL THOUGHTS</td>
<td>4</td>
</tr>
<tr>
<td>ACKNOWLEDGMENTS</td>
<td>5</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>7</td>
</tr>
<tr>
<td>CONTENTS</td>
<td>9</td>
</tr>
<tr>
<td>1. INTRODUCTION</td>
<td>11</td>
</tr>
<tr>
<td>1.1 Motivations and Manuscript Presentation</td>
<td>12</td>
</tr>
<tr>
<td>1.2 Computational Orientations</td>
<td>13</td>
</tr>
<tr>
<td>1.2.1 Traditional Versus (Relatively) New</td>
<td>14</td>
</tr>
<tr>
<td>1.3 The Agent Paradigm</td>
<td>18</td>
</tr>
<tr>
<td>1.3.1 What an Agent Is and What Is Not</td>
<td>18</td>
</tr>
<tr>
<td>1.3.2 My Vision of Agents</td>
<td>24</td>
</tr>
<tr>
<td>1.4 Agent Oriented Engineering Methodologies</td>
<td>27</td>
</tr>
<tr>
<td>1.4.1 Agent Software Development Orientations</td>
<td>28</td>
</tr>
<tr>
<td>1.5 Research Context Unfolded</td>
<td>30</td>
</tr>
<tr>
<td>1.6 Theses: Main Contributions</td>
<td>31</td>
</tr>
<tr>
<td>1.6.1 MAS Conceptualization and Modelling</td>
<td>32</td>
</tr>
<tr>
<td>1.6.2 MAS Development and Maintenance</td>
<td>33</td>
</tr>
<tr>
<td>2. MAS Conceptualization and Modelling</td>
<td>35</td>
</tr>
<tr>
<td>2.1 From Data to Knowledge (and Beyond)</td>
<td>38</td>
</tr>
<tr>
<td>2.2 Application Knowledge Definition</td>
<td>41</td>
</tr>
<tr>
<td>2.2.1 Model Complexity</td>
<td>43</td>
</tr>
<tr>
<td>2.3 The Concept of “Natural Modelling”</td>
<td>44</td>
</tr>
<tr>
<td>2.3.1 More Natural Terminology</td>
<td>44</td>
</tr>
<tr>
<td>2.3.2 Atomization vs. Complexity</td>
<td>46</td>
</tr>
<tr>
<td>2.3.3 Realization in K4Care Project</td>
<td>48</td>
</tr>
<tr>
<td>2.3.4 Communication Semantics</td>
<td>56</td>
</tr>
<tr>
<td>2.4 Environmental Dynamics through Agent Substitutability</td>
<td>57</td>
</tr>
<tr>
<td>2.4.1 Behavioural Equivalence</td>
<td>58</td>
</tr>
<tr>
<td>2.4.2 Ontological Equivalence</td>
<td>60</td>
</tr>
<tr>
<td>2.4.3 Agent Replacement Methodology (ARM) Preliminary Guidelines</td>
<td>62</td>
</tr>
<tr>
<td>2.5 Long-term Interactions as Complex Computational Result</td>
<td>62</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

We live in a time of continuous reassessments in Information Technology (IT). New technological tools are allowing us to deal with so many different aspects of our lives, as they were bits of information. In many ways, we, ourselves, can be seen as “modelled” information: the way we act, the way we think, the way we exist, can all represent relevant aspects of an IT-centred system. The characteristics and expectations of software-engineered systems, on their side, have dramatically changed in the past few decades, with the result that a range of new software engineering challenges has arisen. First of all, most of the software systems are now substantially concurrent and distributed, and are expected to interact with components and exploit services that are dynamically found in the network. In addition to this, software systems are becoming entities which are “always-on” and that cannot be stopped, restored, and maintained in a traditional manner.

Lastly, software systems tend to be open, in the sense that they exist in a dynamic operating environment, where new components join and existing components leave on a continuous basis, and where the operating conditions themselves are likely to change in unpredictable ways. In this context of constant technological evolution, newer development paradigms are expected to support and lighten the complexity of knowledge circulation, as well as its engineering.
1.1 Motivations and Manuscript Presentation

The main reason encouraging my research work on agent technology was the conviction that technology can really enable a constant process of knowledge acquisition, oriented to the satisfaction of human needs: this is what in substance differentiates mere machine systems from mankind societies.

Agent paradigm embodies characteristics of undisputed potentialities, like behavioural and adaptation skills, which, if exploited in the correct manner, can manifest “intelligence”, an element proper of humans, we could argue. Anyway, a pure philosophical approach is rarely a sufficient solution in a business context, when well-defined strategies and policies rule the participation of actors to the market.

Through my research work I intended to investigate the software engineering process based on the agent-paradigm, in order to, on the one hand, highlight and stress the peculiar aspects of this technology and, on the other, derive some preliminary guidelines to be taken into account when approaching the development of agent-based systems. In the specific, I oriented my investigations toward the application of agents in the industrialization of critical, knowledge-intensive, long-term adaptive services (such as medical treatments in home care), as will be reported in details in the followings.

The present manuscript has been structured according to the following orientations. The first chapter covers an explanatory introduction on the agent paradigm, as well as my personal vision and theoretical contribution to the metaphor. The contributions I present in chapter two regard the conceptualization and modelling of Multi Agent Systems. Chapter three covers two practical phases of agent software engineering process: the development and post-deployment related maintenance and evolution of agent-based infrastructures. Chapter four summarises the
results of my research work obtained in a real simulated application in the business context of interest (introduced in section 1.5). Conclusions, notations, appendix and bibliography sections complete the presentation of this manuscript.

1.2 Computational Orientations

Computer paradigms have long life cycles, due to the reluctance of people to learn new ones and to existing software that has been in use for many years. An example is the never-dying language FORTRAN\(^1\) which has been in use in science for decades now. However, FORTRAN is only a suboptimal solution, speaking in evolutionary terms, and computer science has developed many more paradigms since then, within which we could always isolate somehow the concept of “module”. Modules have changed from subroutines, to objects, to components and now finally to “agents”.

As we will see in the following sections, various definitions of agent have been used over the past years. In general, a software agent is a component that can exhibit both proactive and reactive behaviours. It works in an autonomous mode and possibly has a capability to learn and improve its performance. In the past years several agent-based systems have been built; several methodologies and tools have been conceived to aid the development of such systems; several conferences relating to software engineering of agent-based systems have been held. Despite all, software engineering aspects seem to have taken a rest, although some recent efforts have tried to address this issue.

Generally speaking, it is my personal conviction that, independently of the chosen developing paradigm, there exist several considerations that should be always taken into account when developing an information system. The first, fundamental, is the separation of knowledge description from software implementation: this can grant a high level of interoperability.

\(^1\) [http://en.wikipedia.org/wiki/Fortran](http://en.wikipedia.org/wiki/Fortran)
and independence among the components of a system. Another relevant aspect is that different computational paradigms can still coexist together in the realization of the same system, even though some of them are more suitable than others when compared to the project’s goals, the requirements elicitation or, finally, the nature of the application domain.

1.2.1 Traditional Versus (Relatively) New

In recent history of modern computer science has shown us how computations may be categorized according to the approach followed in the exploitation of “processors” (hereby intended generally as system components capable of receiving inputs and transforming them into outputs).

In a first period, computational problems were solved using single and isolated processors. In a second one, solution processes were distributed to a number of previously interrelated processes, executed by a greater number of processors without any deadlocks. In both cases, traditional computing systems were built from components defined through their functional (input/output) specifications with respect to the overall functional specifications of the desired system. The modularity resulting from such an approach lies in the core of the systems traditional functional modularity. According to this traditional understanding on computation and computers, we can recognize any computing device as an externally passive entity, whose internal activity is based on those ones provided by some finite number of externally passive components, with predefined message passing possibilities. Thanks to the internal activities of these components and their addressed communication possibilities the whole system transforms some inputs (provided by the environment) into some required outputs. If some well-specified requirements are satisfied, such transformation is a computation in the traditional sense developed during the modern history of computing, which started in the 30ties of the past
century with the definition and first studies of (abstract devices equivalent to) the Turing Machine [16]. As illustrated in [17], the important aspect in the traditional point of view is that, considering a Turing Machine working in an environment, it gets its input in advance at a beginning of its work and provides the environment with the outputs at the end of its activity. During the computation, the environment is – from the perspective of the Turing Machine – completely passive. Computations are understood, according to this traditional paradigmatic view, as specific processes corresponding to mathematically defined functions.

Whereas functions declare a specific relation between variables and values in a set theoretic sense (by definition, a function coincides with defining a suitable subset of the Cartesian product between its domain of variables and co-domain of its values), the traditional view of a computation (intended as a function) is procedural one: a computation define a function by means of specifying a step-by-step process of elementary computable transformation steps, which transform the given input variable to a corresponding output value (of the corresponding function).

The central problems in (theoretical) computer science originated from the traditional paradigm of computing I just described and are related with the feasibility, description, execution and effectiveness of algorithmic transformation of input data into desired outputs. On the basis of these premises, the computability is equated to the computing power of the Turing Machine (the core idea of the Church-Turing conjecture).

Another possibility of viewing systems as computing devices consists in considering them as an externally active entity, capable of perceiving its outer environment, and acting in it. This is the core idea of the third period in the history of modern computing, when the more or less freely cooperating and communicating processors’ behaviours result in a “superior” behaviour, interpretable as a solution to a given problem.
The activity of the above mentioned type of systems is based on the coupling of sensed data with appropriate acts performed in the environment, or on the activities of individually autonomous components forming these systems. Components of this type are usually called *agents*, and the structures formed by these agents are called Multi Agent Systems (MASs) - cf. e.g. [13]. The behaviour emerging from massive interaction of individually autonomous agents forming a Multi Agent System might be interpreted as a **computation**. According to [14], an emergent computation is supposed to consist of:

1) A *collection of agents*, each following explicit instructions;
2) *Interactions* among agents (without any, or according to the given instructions), which form implicit global (*emergent*) behavioural patterns at the macroscopic level i.e. epiphenomena;
3) A natural *interpretation* of the epiphenomena as computation.

In work presented in [15] an acceptable explanation of the emergence’s meaning has been provided. A phenomenon is called emergent if it is “... *a product of coupled, context-dependent interactions. Technically these interactions, and the resulting system, are non-linear. The behaviour of the overall system cannot be obtained by summing the behaviours of its constituent parts. However, we can reduce the behaviour of the whole to the lawful behaviour of its parts, if we take non-linear interactions into account*,” as emphasized in [15], indeed.

From a computer programmer’s position, instead of a traditional view of computation (as a result of a prescribed sequence of elementary actions), we have a new approach: a computation seems to be the result of more or less coordinated interactions of a (large) number of individual and (more or less) autonomous agents. Interactions of an agent with other agents and with its (dynamically changing) external environment are a real promise on
how computational power of agents and MASs can be increased, compared to the one of the Turing Machine - cf. e.g. [18].

In general, the interactions within a MAS involve the external world and the activities of individual agents’ behaviour (interpreted as a computation): in this manner, during a computation (rather than before and after it as it occurs in the case of traditional algorithms), the whole system may lead to such computations which cannot be carried out by a Turing Machine, as stated in [19]. In this light, agents and MASs can be considered as very powerful computational devices and may contribute with many innovative concepts to our traditional picture of the theoretical computer science and engineering.

We can imagine agents as entities which exist in the form of a large variety of different types of software products – as intended in a wide range of application areas; cf. [20]: not only an innovative product of the development of computer programming techniques but also a product of development of electro-, mechanical-, and computer-engineering, as electro-mechanical devices for automation of different physical processes. This is an important difference between real computers and the abstract Turing Machine. In [21] and [22] it is been stated how computers, as built and used, are the result of a convergence between the development of machine- and electro-engineering and the progress in understanding computations as the process of performing actions on symbols, as Turing Machine does. These two dimensions of agents (interaction with dynamically changing environment and embodiment) converge into a new conception of machines as embodied, autonomously sensing, acting and deliberating agents, in the form of robots.

The above mentioned difference, together with the properties of autonomy and continuity in software behaviour, the relevance of embodiment, the importance of communication between individually
independent, autonomous computational units (intentionally or as an emergent effect of their co-existence in a shared environment), and so forth, are to be considered as crucial for distributed systems, like MASs indeed.

Instead of having specific programs translating mathematical functions into some more procedural languages, we should think in terms of autonomy and continuity of service for those systems which have the ability to sense the environment and act in it. Whereas the principal subject in the study of computations, from the point of view of the traditional paradigm, is to consider them as defined in the form of finite sequence of basic transformations executed (sequentially or in parallel that be) on appropriate data structures, the agent paradigm can be characterized by viewing a computation as “the communication result of collections of individually autonomous agents”, and vice versa, “viewing communication of collections of individually autonomous agents as a computation”.

1.3 The Agent Paradigm

Substantially, the agent paradigm advances the modelling of software systems by embodying a stronger notion of autonomy and control than objects, including the notion of reactive, proactive and social behaviours, as well as assuming inherent multi-threaded control. This allows handling the complexity by powerful abstractions in engineering software systems.

1.3.1 What an Agent Is and What Is Not

Despite developers’ general perception of dealing with an innovative and well-promising technological paradigm, there are still common misunderstandings in the conceptualization of an agent, in the role which this technology plays (or should play) and about the suitable conditions under which it may be utilized. Agent-based solutions have already been developed for many different application domains, and field-tested agent systems are steadily increasing in number. In addition, a range of theoretical
and experimental results attest to the fact that the scientific foundations of agent-based systems are becoming increasingly well understood. But despite these significant advances in the science of agent systems, comparatively little effort has been devoted to understanding how to fully engineer them. I am convinced that a preliminary clarification on some key concepts about agent technology would be surely fruitful to the reader, in order to better comprehend the upcoming understandings of this manuscript.

I have been several times asked about "what intelligent agents are or are expected to be". In order to answer this question, I would first try to demystify some of the most resounding misunderstandings surrounding agent-oriented developments. This could help the reader find the answer, autonomously. In general terms, agents should not be intended as:

- A religious or dogmatic acceptance. Some designers tend to utilize agents irrationally and impulsively. There are plenty of applications for which more "conventional" software development paradigms (such as pure object-oriented programming) would be more appropriate and easy-to-apply. There is a tendency to believe that agents are the right solution to every problem. As a consequence, agent solutions could be developed for quite inappropriate problems. The other form of dogma associated with agents relates to their definition. Most agent developers have their own opinion on exactly what constitutes an agent and no two developers appear to agree on exactly the same opinion (see [23] for a collection of agent definitions).

- A product to be overstressed. Problems that have troubled software engineers for decades are still difficult with agent systems, too. There is sometimes an evident perception that any system developed using agent technology could have been built just as easily using non-agent techniques. Agents may
undoubtedly make it easier to solve certain classes of problems but they do not make the impossible possible. The reason for this is that the atomic problem solving components within agent-based systems still have to be able to perform the necessary domain tasks, and their implementation can only use the limited techniques that are currently available. Naturally, further leverage can be obtained by applying and implementing multiple problem solving methods and by carefully managing the interactions between the components.

**: A trendy management solution.** Managers generally tend to overestimate forecasts about any new promising (in terms of revenue) technology. This is the case of agents, as well. In many cases, managers that propose an agent-based project could not actually have a clear idea about what “having agents” will mean to them. In other words, they have no clear vision of how agents can be used to enhance their existing products, or how they can enable them to generate new product lines. As a consequence, agent-based projects are often initiated with no clear goals in mind (other than to “have” agents). With no goals, there are also no criteria for assessing the success and no way of telling whether the project is going well or not.

**: A conventional buzzword.** One of the reasons why agent technology has become recently so popular is that the “idea” of an agent is quite intuitive. If on the one hand this is a good thing (the concept of an agent is widely applied in many different disciplines), on the other, it also encourages developers to believe that they can easily understand concepts which, in fact, they do not. A clear example of this issue is the belief-desire-intention (BDI) model of agency, the result of Georgeff and colleagues work
The BDI model is really interesting from the point of view of the agent developer because it is sustained by a respectable theory of human agency, it has an elegant logical semantics [25], and perhaps most importantly, it has been proved in extremely demanding applications. Unfortunately, the “brand” BDI has nowadays been applied to so many different types of agent (many of which are not BDI systems), that the expression has lost much of its original meaning.

A generic solution. Numerous agent-based architectures have been developed to date, which deal both with the micro and macro levels of agent systems. In general, these architectures are developed by building a solution for a “particular” problem, and then “generalising”. Indeed, there is a temptation to imagine that the architecture and techniques developed for one problem domain can be directly applied to another, due to the successful agent solution developed. This can represent a serious fallacy as it inevitably leads designers to try to apply the architecture to a problem for which it is patently inappropriate. It would be certainly advisable to attempt to apply the same architecture only to problems with quite similar characteristics.

A synonym of Artificial Intelligence. Another reason of misunderstanding is to equate agents with intelligent problem solving. Those unfamiliar with the achievements and failures of Artificial Intelligence (AI) often believe that agents are capable of human-like reasoning and acting. Obviously, this is not the case: such a level of competence is well beyond the state of the art in AI. Agents may sometimes exhibit smart problem solving behaviour, but it is still very much limited by the current state of the art in machine intelligence. When one builds an agent-based
application, there is an understandable temptation to focus exclusively on the agent-specific aspects of the application. The result is often an agent framework that is too overburdened with experimental AI techniques to be actually usable. In general, a more successful strategy would be to build agents with a “minimum” of AI techniques: once that success is obtained with such systems, they can be progressively evolved into richer systems. This is essentially what Etzioni called the “useful first” strategy [26].

\[ \text{An anarchic technology.} \] Another common misconception is that agent-based systems can be developed simply by putting together a number of agents in a sort of arena; that the system does not require real structuring and that all agents are peers. Many agent systems require considerably more engineering efforts than this. For large-scale systems, or for systems in which the society is supposed to act for a common purpose, this is particularly evident. In such cases, by structuring the society we will reduce the system’s complexity, increase the system’s efficiency, and model the problem being addressed more accurately. The detailed nature of this structuring is clearly dependent on the specific problem.

Having acknowledged that there are still main misconceptions around agent technology, let us try to focus on what it represents, on the contrary.

Primarily, an agent (or an agency of agents) should be intended as:

\[ \text{A well conceptualized approach to domain modelling.} \] Agents are first of all an abstraction tool and in this light they appear to provide a powerful way of conceptualising, designing, and implementing particularly complex classes of software systems. Agent-based solutions require a different
development approach from software designers, who are asked an additional
effort in completely understanding differences which the agent paradigm
involves. Opting for an agent-based system will undoubtedly affect all
phases in software’s realization: from the project’s goals definition to the
Multi Agent System’s maintenance and enhancement. Designers will be
asked to think in terms of “living” entities (actors with roles) rather than
“passive” components of the system (objects and hierarchy).

- **Software development.** Dealing with agent technology requires an
  engineering approach and a development methodology as it does
  with other conventional software, as well. For instance, we should
dedicate particular attention to the definition and formalization of
project goals. On the one hand, it helps us have an assessment
reference at the end of the project realization and, on the other,
not to lose the right orientation during the entire software
engineering process: from system analysis to design, from
development and testing to deployment and maintenance. The
result of this possible neglect is the project block, not because of
agent-specific problems, but because of the omission of basic
software engineering practices. The abandonment of the software
process is often justified with the fact that software engineering
for agent systems is still a research area. Nevertheless, any
principled software development technique is better than none.
Thus, in absence of agreement on widely accepted agent-oriented
development techniques, other incisive, semantically effective
techniques may be used (as I present in the following of this
manuscript, in Section 2.1) to support agent systems design and
development.

- **A distributed system.** Multi Agent Systems tend, by their very
  nature, to be distributed and the idea of centralised MAS is itself
an oxymoron. In building MASs it is vital not to ignore the lessons learned from the distributed systems community, as, for instance, problems such as distribution do not go away, just because a system is agent-based. Distributed systems have long been recognised as one of the most complex classes of computer systems to design and implement. A great deal of research effort has been devoted to understanding this complexity, and to developing formalisms and tools that enable a developer to manage it [27]. Despite this research effort, problems inherent the development of distributed systems cannot be considered as solved, yet. Multi Agent Systems will, if anything, be more complex than a typical distributed system. The MAS developer, therefore, still has to recognize and plan for problems such as synchronization, mutual exclusion for shared resources, deadlock, and livelock.

1.3.2 My Vision of Agents

Objectively, the enormous circulation of information obliges us to dominate a decisional complexity more and more onerous. From here, we gradually perceived the necessity to exploit newer technology which would embed proactive capabilities, such as communication and delegation. My

✓ Standard-requiring technology: Standards enable interoperability and communication in distributed systems. Because agent-based systems are distributed, indeed, they need standard to face complexity and granularity issues. Despite the lack of internationally accepted standards, there are a number of de-facto standards in the area, which may usefully be employed in most cases. A relevant example is provided by the KQML [18], an Agent Communication Language (ACL) that has been employed in practice in many agent development projects.
principal intention has always been that to bring agent-technology exploitation close to real human practices and necessities, moving it from a never-ending theoretical research stadium. I intended to determine the conditions on the basis of which agents would be usefully applied to classes of practical problems. Agents have something more than other technology: they are expected to behave not to simply run; here comes the possible definition of “intelligent agents”.

First of all, as agents are not biological creatures, we should talk about a sort of “artificial intelligence”, rather than mere intelligence. Artificial Intelligence (AI) deals with intelligent behaviours in artificial systems [28] and it has found application in a natural way in the context of agents-based software. Behaviours involve abilities such as reasoning, learning, communication and social perception [29] in complex environments. Agents seem to be a “natural” application for all these expectations [30], as they reflect the intention of researchers in computer science to reproduce artificial systems driven by behavioural models. After considering different definitions of agent, I came to clearly identify my personal vision about the paradigm, which I globally synthesize hereafter.

All definitions of agent encountered in literature presuppose an (implicit) foundation of evidenced intelligence, which in turn might have been expressed by emphasizing some specific characteristics in place of others (mobility in place of social capability, for example, or communication in place of reaction, and so forth).

It is unquestionable that we cannot univocally define intelligence, but what I feel to assert is that it has to be undoubtedly something ascribable to the intellect the catalyser of a behavioural pattern (intended as a sequence of actions). In other words, the actions undertaken by an agent can be seen as the result of internal processes of intelligence synthesis. In work published in paper [1] I propose to represent the definition of agent (and
therefore its intelligence) through a so-called “Agentometric Net”, as illustrated in Figure 1: just as a net is composed of rays, my vision places all main capabilities of an agent along with radial directions. As all the rays in a net are critical elements for its durability and resistance, in the same manner the thoroughness and integrity of the Agentometric Net guarantee the crucial adaptability of an agent to the environment.

Let us focus a little more in depth on the meaning of the previous illustration. In agent-dedicated literature many definitions of agent can be found, as described, for example, in works [20], [23] and [30]. An apparently initial necessity to differentiate among applied agents can be noted, objectively justified by the fact that designers are tacitly influenced both by the application contexts and the projects’ founding characteristics (distribution, adaptation, mobility and so forth). The MuBot definition of agent, for example, stresses concepts such as autonomy and reasoning; the Maes agent is autonomous and goal-oriented; the KidSim agent manifests specialization and persistence; the Wooldridge-Jennings agent is, next to autonomous, capable of socially relevant actions, reactive and pro-active,
whereas, finally, the *Georgeff-Gulyas* agent embodies skills of mobility and reasoning. All these definitions are correct in their singularity, but limitedly constrained in, and by, their development circumstances. This is the reason why I have formatted all single definitions and formulated a more generalized characterization of agent, whose *intelligence is manifested through the exploitation of the most appropriate skill for a given specific goal*, as illustrated in Figure 1, indeed. The agent paradigm represents a tool for modelling individuals’ behaviour separately, and then studying the emerging behaviour of the society of these individuals.

### 1.4 Agent Oriented Engineering Methodologies

According to the actual state of the art in agent-oriented technology, various AOSE (Agent-Oriented Software Engineering) methodologies have been proposed and de-facto often applied, although none of them may be considered as a well-established and accepted standardization. In addition to this, both micro (agent) and macro (agent society, i.e. MAS) levels of agent systems usually leads developers to face even a larger variety of issues and problems (from the behavioural characteristics of a single agent to the global interaction strategies of the whole agent community).

Some of the existing methodologies provide specification-based formal verification, allowing software designers to detect errors at the very beginning of the development process, as illustrated in [31]. Other methods inherit Object-Oriented (OO) testing techniques to be exploited later in the development process, upon a mapping and forcing of agent-oriented abstractions into OO constructs [32]. Researchers in the area of MASs have proposed a number of different approaches for modelling agent systems, based on different *“metaphors”*, but none of them can reasonably be considered of general purpose.

For instance, *“the ant algorithms metaphor”* [33][34] has demonstrated to be useful in efficiently solving complex distributed
problems such as routing and distributed sorting; “physical metaphors” [35], [36], focusing on the spontaneous reshaping of a system’s structure, may have useful applications in pervasive and mobile computing; “societal metaphors” have been effectively applied in robotics applications [37], [38] and in the understanding and control of highly-decentralized systems [39], [40].

My research work primarily focused on the development of medium-large sized agent systems, embedded into dynamic environments, and that had to guarantee predictable and reliable behaviours. For these kinds of systems, I believe (and am going to illustrate why) the most appropriate, the most adherent metaphor is that of a “human organization”, as exhaustively exposed in [41] [42] [43]. Anyway, despite these promising premises, there is still a context of methodological uncertainty, in which the cohabiting of technology born for different purposes could provide agent software designers with a powerful and effective abstraction tool. Ontology, for example, as a means of higher semantic representation, can play an important role in modelling agent systems, as well as in offering the opportunity to automate the development of the MAS, as I will illustrate in details in Chapter 3.

1.4.1 Agent Software Development Orientations

Engineering business-oriented agent-based software requires, as for any software development, the adoption of well-defined methodology, technology and standards. So it was for the project in which my research work has been involved. Three were the elements, in particular, on which I focused and which deserved particular attention: the agent communicative formalism, the analysis and design methodology, and the MAS-supporting technological infrastructure. The ones I selected to support my investigations are briefly introduced hereinafter.
I have defined interoperability of agents through FIPA\(^2\) (Foundation for Intelligent Physical Agents) specifications. They represent a collection of standards, which are intended to promote technology and interoperability specifications, facilitating the end-to-end interworking of intelligent agent systems in modern commercial and industrial contexts. In an open and distributed agent-based environment, for example, the need of standard mechanisms and specifications is crucial for ensuring interoperability of distinct autonomous systems.

As regards the AOSE methodology I have chosen GAIA\(^3\), as the process of building MASs is radically different from the one of building more traditional software systems. Specifically, I would agree with the vision recognized in GAIA according to which MASs can be conceived in terms of an organized society of individuals: each agent plays specific roles and interacts with other agents, accordingly to protocols determined by their roles. In such a social structure a software system is conceived as the computational instantiation of a group of interacting and autonomous individual agents. Nevertheless, each agent can be seen as playing one or more specific roles: it has a well-defined set of responsibilities or sub-goals (altruistic or opportunistic that is) in the context of the overall system and is responsible for pursuing these autonomously.

In compliance with this scenario, interactions are rather seen as a means for an agent to accomplish its role’s goals in the system. The evolution of activities in the organization, deriving from the autonomous execution of agents and from their interactions, determines the achievement of the application overall goal. In other words, the organizational metaphor, other than being a natural approach for developers who are continuously immersed in a variety of organizational settings, opens up the possibility of

---

\(^2\) http://www.fipa.org/about/index.html

reusing a variety of studies and experiences related to real-world organizations.

Regarding the deployment technology, I have finally identified JADE4 (Java Agent DEvelopment Framework) as to be the most appropriate. JADE is an enabling technology, a middleware for the development and run-time execution of peer-to-peer applications which are based on the agent’s paradigm and which can seamless work and interoperate both in wired and wireless environment.

The two major aspects of its conceptual model can be summarized as follows: 1) it is a distributed system topology with peer-to-peer networking, and 2) it is software component architecture with agent paradigm. The network topology affects how the various components are linked together, whereas the component architecture specifies what the components are supposed to expect from one another. JADE is an Open Source project around which a community of users and contributors has progressively and relevantly grown up.

1.5 Research Context Unfolded

Most of my research work has been inspired within the K4Care (IST-2004-026968) European Project. The main purpose of K4Care project was to guide any activity that fostered the realization of an integrated system of Home Care (HC) services for the treatment of elderly, disabled persons, and patients with chronic diseases in Europe. This objective has a direct social implication of helping people partially, temporary or totally dependent to live in their environment as long as possible, and to contrast the improper use of institutionalization.

In addition to this, the aim of the project was also to model and design the Home Care domain taking into account the requirements of a

4 http://jade.tilab.com/
typical European HC patient, providing detailed information about the structures. The project itself proposed the definition and realization of an ICT based model of Home Care, where HC stands for the care services provided to a patient who requires assistance at home. The typical HC Patient (HCP) is an elderly patient, with co-morbid conditions and diseases, cognitive and/or physical impairment, functional loss from multiple disabilities, and impaired self-dependency. We shall refer to this “average patient” as the HCP. The healthcare of the HCP is particularly complex because of the growing number of patients lying in such circumstances, and also because of the great amount of resources necessary to guarantee a long-term assistance of quality.

The K4Care assistance model\(^5\), defined essentially by medical partners, was established by defining the actors of the HC domain (physicians, social assistants, nurses, rehabilitation professionals, HCPs, relatives, informal care givers, citizens, social organizations, and so forth), their roles (integrating old and new EU country working modes) and their interactions. Interactions among health professionals, computer scientists and technology centres have basically occurred in order to define the health-care model, whose critical aspects have been identified and solved in a cyclic process of evolving construction. The work was developed through the identification of the initial actors and roles, together with a dictionary of concepts and the required international codification systems (ICD9CM, ICD10CM, ATC, etc.), which were also formally defined and standardized.

1.6 Theses: Main Contributions

My personal contribution and support to the realization of the HC platform concerned about the research work in the conceptualization, modelling, development and maintenance of agent-oriented techniques and

---

\(^5\) http://www.k4care.net/fileadmin/k4care/public_website/downloads/The_K4CARE_Model_Validation.pdf
technologies, whose application was oriented to manage, on the one hand, the project’s complexity, and highlight, on the other, the agent paradigm competitiveness in real application contexts. Actually, the lack of well-established and widely accepted agent-oriented development methodologies has induced me to consider any relevant aspect of the computational paradigm and to focus on the entire lifecycle of MAS. My intention was to provide the scientific community with results of practical utility. The main contributions and the scientific results of my dissertation can be summarized in four points, further organized in two thesis groups.

The first group aims at clarifying the conceptual approach which should be pre-emptively adopted when modelling a domain through agents, and its consequent long-term advantages in actors’ communications and negotiations. The second group focuses on the extensibility and maintenance aspects of MASs, giving particular attention to the ever-present, post-deployment changes in the environmental dynamics and service management policies.

1.6.1 MAS Conceptualization and Modelling

Discussion focuses both on theoretical and practical relevance of agents’ exploitation in modelling complex, highly populated domains usually characterized by the existence of long-term interactions among actors of the system.

\textbf{T1.1} The agent-paradigm enables a more natural description of the application knowledge: it permits to govern and embody complex knowledge representations by atomizing the behavioural characteristics of domain actors.

The work synthesized in thesis T1.1 aims at highlighting how agent-paradigm can represent a natural and strategic competence in the definition of intelligent systems, in which knowledge plays a
fundamental role in the characterization of participants’ abilities. I will provide, among the others, a \textit{characterization of the minimal level of application knowledge} representation that an agent should embody in order to be considered autonomous enough. Sections from 2.1 to 2.3.4 sum-up the research work involved in T1.1.

T1.2 \textit{Agent-based software gracefully adapts to long-term active negotiations, in which continuous environmental and strategic changes are crucial for the achievement of the initial goals.}

Today’s business services complexity and orchestration lead individuals to long-term negotiations and adaptations to the environmental changes, which will cause, in turn, a continuous generation of new knowledge and strategies. One of the most relevant aspects in this scenario is the concept of \textit{service continuousness}, which I strictly relate to agents’ availability through the definition of a \textit{replacement methodology}. These are the key-points of Thesis T1.2. Sections from 2.4 to 2.5.3 illustrate the research work I conducted in relation to the previous issues, adapted to agent-based systems.

1.6.2 MAS Development and Maintenance

The adoption of an engineering approach in the realization of agent-based applications, whether not compulsory, is at least recommendable in order to guarantee systems with extensibility, adaptability and testability, as well as to decrease the objective development complexity which an agent-based system inevitably involves.

T2.1 \textit{Agent-oriented code generation provides semantic relevance to fast prototyping process of MAS-based applications.}

Generally, costs and efforts involved in the realization of software architectures can be significantly lightened by the adoption of
semantics-based prototyping techniques. This is particularly true in the case of Multi Agent Systems, which naturally come with a higher semantic simulation and testing context. Sections from 3.1 to 3.3.1 cover the application of principles mentioned in Thesis 2.1.

**T2.2** The agent-paradigm enables formal extensions of the overall system’s behavioural model, avoiding repercussions on the application’s deployment cycle. This is a fundamental aspect in the recognition, management and embodiment of new emerging application knowledge.

Knowledge Management (KM) process, in a specific domain, does not end with the release of the software designed for it, because its utilization will inevitably implicitly cause the generation of new user’s knowledge, previously unrecognized and unrevealed. The behavioural model of an agent can be orchestrated and extended through its elementary components (actions) in order to permit the system to embody the newly manifested knowledge by the end-users and to gracefully evolve towards newer application scenarios. This is the essence of Thesis T2.2, whose investigations are reported in sections from 3.4 to 3.6.3 in this manuscript.
Chapter 2

MAS Conceptualization and Modelling

Technology by itself is not the solution to human problems. It is usually perceived as if it were, because of its nature of being the closest interaction element to humans, to the usability and provision of a service and to our sensory perception. Technology can be seen as the materialization of a service; services, on the other hand, are the means through which our needs, our necessities are finally satisfied. In other words, technology is only the final product of a far more important process: “knowledge management”.

Technology, intended in its widest meaning (as it could be, for instance, a new software paradigm development, like agents), enables the management of knowledge (as explained in section 2.2). It is irrefutable that technology embodies knowledge-related aspects (designers capabilities who created it), but, per se, technology should not be considered as anything else but the result of a conceptualization process. Nevertheless, it is unquestionable that the variety of human needs refers to different specific domains, in which some technology can be more suitable and naturally adoptable than other.
Basically, according to my own vision, the identification of human needs finds its foundations in philosophical debates, which, in turn, lead to the recognition of new approaches in problem-solving. The latter justify the efforts put into the realization of new technological scenarios, which would finally represent the catalyser of knowledge in a specific application domain (enabling, in turn, new philosophical brainstorming).

Illustration in Figure 2 aims at introducing the concepts I have mentioned before. I have named it *Business Evolution Diamond* (BED).

![Figure 2: Business Evolution Diamond – BED](image)

I have substantially recognized and defined the following associations on BED’s elements (see Figure 2 for reference):

1) The *philosophical issue* with the *intention* of designers to create artificial systems with human-like behavioural aspects and characteristics;

2) The necessity of adopting *new approaches* with the *conceptualization* of the agent paradigm;

3) The practical realization of a *technological context* with the *development* of Multi Agent Systems;
4) The *application domain* with the *highly populated and knowledge-demanding, long-term interaction-based business contexts*.

The illustration in next Figure 3 (a sort of hybrid “*onion graph*” that re-organizes concepts from Figure 2) highlights how the positioning of technology in a scenario of new needs definition is crucial, indeed, but not decisive, being technology by itself the result of precedent layers’ existence. As we clearly understand, technology is neither the very goal nor the initiator of the brainstorming necessary to advance towards new services conceptualization: it is, on the contrary, the unsubstitutable means enabling such a process.

![Figure 3: Technology's positioning in new application conceptualization](image)

One of the primary motivations because of which humankind continues the research of new technology is (apart from their natural and instinctive attitude to invent and assemble) the continuous *perception* of new (should it be *real* or *not*) needs to be satisfied. But where does technology really fit in? How can new technology, such as the agents for instance, represent a mile-stone in an evolutionary scenario and influence our every-day life?
Technology is not the mere answer to our needs but the means for their satisfaction: it can naturally accelerate the automation process of new technology creation, by leveraging on the embodiment of designers’ knowledge. This is the point I would like to stress: agent-technology, as all other technology, is nothing but the result of knowledge conceptualization, realization and exploitation.

2.1 From Data to Knowledge (and Beyond)

What is "knowledge" anyway? And should the previous question find a suitable answer, what is "knowledge management" then? Of course, these are not trivial interrogatives, at all. According to [44], [45], knowledge can be seen as the result of a process of evolution or adaptation perception. Despite this, numerous system experts and analysts tend to confuse competence and knowledge with information, and information with data. Permit me to clarify some crucial aspects related to these issues, before we try to apply them to the development of MASs. As illustrated in Figure 4, knowledge is directly related to information (and not to data) and it is itself the result of a "transformation" process, which ends up with the creation of competences (skills and capabilities). Now, let us consider a little more in depth each of the concepts reported in the following illustration.
Data (green colour square in Figure 4) can be seen as a sequence of quantified or quantifiable symbols: a piece of text, pictures, figures, recorded sounds and animations are all examples of quantifiable data. In this light, data are necessarily mathematical entities, and thus are purely syntactic. This means that data can be totally described through structural, formal representations. Being quantifiable, it can obviously be stored into computers, as well as processed by them (machine-related component in Figure 4). Data manipulations are related to machines, which are able to "transform" (in a deterministic manner) data into other data through programs. Data, finally, can be sent (received) to (from) somebody (other computers) through "messages", which may have the form of letters, documents, reports, and so forth. The message is the abstraction key point in the transformation of data into information and it presupposes the existence of two elements: a sender and a receiver (of the message).

Information is an abstraction (first orange colour square on the left in Figure 4) in the mind of a person, in the form of thoughts which represent something of significance to that person; it is an element of the human-related aspects. The reader should keep in mind that this does not pretend to be a definition; it is more a characterization, because terms such as "mind", "thought", "something", "significance" and "person" cannot be univocally defined and formalized. I assume here an intuitive understanding of these terms. It is not possible (in the sense seen before) to automatically process information in a computer; for this reason, information has to be transformed into data, and, then, it is not information anymore. Similarly, it is not possible to store information (always in the sense seen before) in a computer; what is stored is the representation of information in the form of data. The crucial problem is to make two people (much harder is the case of two computers or artificial systems, in general) have precisely the same understanding of that representation, as information derives from the
**understandings of the message we receive.** Information can be considered as an inner property of a person and so it may originate from an inner perception. A fundamental distinction between data and information is that the former is purely syntactical, whereas the latter necessarily contains semantics (implied by the words “meaning” and “understandings”, used in its characterization before).

**Knowledge** is also a personal, inner and further abstraction of the information. **It is all the information we gather** of something (events, objects, person, facts, and so on), **assorted with our beliefs, intentions, desires, emotional profile, cultural aspects**, and so forth. It does not depend only on a personal interpretation, as with information, because it requires a personal experience with the **object** of knowledge. As previously introduced, the representation of information through data can be stored into a computer, but, on the basis of this assumption, knowledge is not subjected to a **complete** and **direct** representation (in such a sense, we cannot univocally represent, for example, beliefs, emotions, and sensations); so it cannot be **inserted** into a computer in a classical way (as a sequence of symbols).

**Competence** is knowledge plus experience. It can be considered as an ability of executing a task in the “real world”. A person has competence in some field if he or she has demonstrated through past accomplishments the ability of executing a required “task”. Competence can be even associated with **creativity** and **improvisation**. In this light, competence requires knowledge and personal capacities for realizing something concrete: therefore, it is impossible, in such a sense, to directly introduce **competence** into a computer system.

Having realized that only information can be directly reduced to data, the challenge in Information Technology is to try to adopt sort of formalisms which could permit designers to deal directly with knowledge-
and competence-related aspects in artificial systems; formalisms which may carry the same abstraction opportunities seen in the previous section. I will explain how Multi Agent Systems can theoretically embody these peculiarities: it depends on how they are going to be conceived and exploited. Despite the previous characterization of knowledge is not directly adoptable in a business context, the combination of data and information, to which it is added an expert opinion, skills and experience, can result in a valuable asset, utilized to support decision making processes.

This is the crucial point: the acquisition of knowledge can improve and enhance decision-making skills, which are to be measurable in terms of quantifiable revenues. The emphasis, then, is on knowledge being applicable or executable (for it is business-related) and on knowledge being a refinement of information (for this is how we derive value from data).

2.2 Application Knowledge Definition

Application Knowledge (AK) is all the knowledge (in the sense introduced in the previous section) necessary to successfully achieve an objective (like running a business, for example). Both knowledge and competence univocally characterize and differentiate entities (humans) of a system (society). Through Multi Agent Systems we intend to transfer such properties to computer-based systems, so that we could reproduce intelligent models for systems which may evidence similarities (in behaviour and constitution, indeed) to a society of “individuals”. MASs are the technological realization which most resembles these characteristics, simplifying the formalization, representation and utilization of the application knowledge in a domain.

The work I have presented in [2] introduces domain application knowledge as, generally, a complex entity to be defined because of the
heterogeneity of its components. Despite these difficulties, particular classes of knowledge can be generically identified and circumscribed.

For instance:

1) **Declarative Knowledge** (DK): DK characterizes “what” a domain is composed of; its constitutive elements, their relations, their interactions and possible hierarchical dependencies. DK is composed, in turn, of:

   a. **Domain Description Knowledge** (DDK): it refers to the profile of subjects acting in the domain (people, professionals, social organism, institutions and so forth, and the skills, concerns, aspirations of the individuals which they represent), the environmental resources to be accessed and the policies governing the environment;

   b. **Service Knowledge** (SK): it comprises the definition and description of all services conceived in the “society of individuals”, in a specific context or domain;

   c. **Technological Knowledge** (TK): it intends to establish a common conceptualization of the technological architecture and its constructs, and to enable the automation of development processes. TK includes various elements, such as the general conceptualization of the development components (i.e. messages, encodings, communication languages, and so on), the methodology or the approach to be adopted, and the implementation and deployment policies that must comply with selected guidelines.

2) **Procedural Knowledge** (PK): it represents what an actor of the domain is able or entitled to do, emphasizing its “dynamic behavioural capabilities” (that is, its personal capabilities which can be combined together in order to define much more complex
interactions with other actors of the domain). One of the most important issues regarding the PK is the possibility to formally describe it by formalisms enabling its orchestration and reutilization. PK comprises, in turn, the Objective Knowledge (OK), which describes the modalities and the strategies by which agents of the system try to achieve their own goals.

2.2.1 Model Complexity

In section 2.2 I have proposed a view of the application knowledge as an assemblage of well defined components, which determine the characterization of a business context. It clearly resulted that modelling AK is a complex task.

Although there can be a touch of genius in all of us, in the realm of industrialized software we cannot rely (only) on divine inspiration to carry us through. We are asked to consider more disciplined ways to master complexity. When conceiving new application solutions, software designers are often required to manage (at the same time) elements deriving from different information domains: business model, environmental structure and its regulations, and technological requirements and difficulties. Trying to model a technological solution becomes a challenging task: it is not expectable (nor feasible) to govern such a complexity as a whole. The “Divide and conquer” seems to be the resolving approach to this problem, used from early software projects and still in use today. The technique of mastering complexity has been known since ancient times: “divide et impera”\(^6\). However, the line along which the complexity is divided changed and there is still a debate on which way is the best, even if divisions by responsibility and business competence seem to meet the highest acceptance in the community of system designers.

When designing a complex software system, it is essential to decompose it into smaller and smaller parts (of course, not indefinitely as I present in next section 2.3.2), each of which we may then refine independently. In this manner, we satisfy the very real constraint that exists on the channel capacity of human cognition: to understand any given level of a system, we need only comprehend a few parts (rather than all parts) at once. Indeed, as Parnas observed, intelligent decomposition directly addresses the inherent complexity of software by forcing a division of a system's state space [69]. The “divide et impera” philosophy applied to software engineering is what best seems to approximate the one underlying the agent-based approach; the philosophy which leads to consider the agent-paradigm as a natural modelling technology in highly populated domains.

2.3 The Concept of “Natural Modelling”

How may the agent paradigm be considered as more effective than others in conceptualizing new software solutions? Why the choice of the term “natural”?

There are several reasons why I consider agent paradigm as a more natural (in the sense of more suitable) solution for designers when dealing with the modelling of “crowded” (populated) domains, as presented in the followings sections.

2.3.1 More Natural Terminology

First of all we should not forget that agent-based software is, indeed, software. This consequently implies the fact that both modelling and realization are the result of a commitment (and mutual understanding) between the contractual parties involved: the customer, on the one side, and the developing company (the contractor, in general), on the other. By modelling domains though agent-based systems, IT designers and customers
will be naturally led to utilize a common “language”, use common terms. This represents a relevant factor of failure avoidance in a project’s development and it is often underestimated by managers.

Illustration in Figure 5 proposes to clarify this concept. From one side, we have the *eclectic* (and sometimes too abstract) language spoken by business experts; from the other, the *incomprehensible* terminology of most IT designers, who tend to hermetically think only in terms of algorithms, connections, flow-charts, objects, methods, and so forth. In any case, what we really have to deal with in the real world is people, things and rules. All other abstractions can be inherited from this assumption.

Agent paradigm help reach *convergence* in exploited terms (labels in Figure 5), by synthesizing human-specific concepts, such as actor, role, actions, tasks, behaviours, strategy, and so forth. These terms are much better understood by domain experts, who are not IT experts, generally. Thinking in terms of “actors” (an agents) “acting” in a specific “environment” (MAS), often mapping a tangible human “role”, is easily
perceived (and imagined) by parties as a *more natural* manner to model, in comparison with the classical terms traditionally engaged.

In other words, agent-related concepts tend to **reduce the communication gap between the commissioner and the software designers** at a very early stage in the conceptualization of the software, by bringing closer to domain experts understanding those concepts which are typical of the IT world (and vice versa); close to the way environment is naturally perceived by humans.

It is my opinion that the fundamental concept which differentiates agents from other technology, as well as defines their uniqueness, is that they act in compliance with a **behavioural model**, by synthesizing and deriving actions even without the intervention of humans, and following the outlined guidelines of their inner strategy and objectives.

We can characterize behaviours as a sequence of actions executed by an agent in order to achieve a goal, be it explicitly or implicitly defined (even taking no action, in this sense, is the simplest behaviour that an agent can assume). I have presented some major examples on agent behavioural characterization in my works in papers [6] and [8], for the definition of care treatment and administration services provision.

### 2.3.2 Atomization vs. Complexity

Let us imagine, for a while, a scenario in which software designers are asked to provide the planning of an artificial system’s development and the moment when they are about to decide on the technological solution to adopt for the addressed objective. In any case, they will need to (I named them as the “*three R*” – 3R):

a) **Replicate** the contextual environment;

b) **Reproduce** the business domain of interest (together with its topology, functionality, strategies, logics and services);
c) **Realize** the technological architecture which will embody all the components mentioned in a) and b), not to mention all the specific arrangements necessary to orchestrate the system as a whole.

Now, trying to manage all these elements as a single integrated system, with centralized controls and directives, would lead to an ungovernable complexity, as well as to the loss of the initial business orientations, which should always remain clear during the whole process of the software development. I have contributed to the MAS platform developed during my research period (as I illustrated, for example, in papers [8] and [11]) complying with these guidelines. Developments evidenced how the adoption of a multitude of atomized agents enormously lightens the work of designers, who can concentrate on the modelling of each actor’s singularities and capabilities, observing how the system complexity can be the result of an emerging phenomenon. In other words, complexity can be seen as the resultant of communication constellations occurring among actors (agents) in the environment (MAS).

![Figure 6: Atomization of a Multi Agent System](image-url)

According to these introductory assumptions, the agents I have conceived reflect the approach illustrated in Figure 6: they constitute the
populated environment whereas their capabilities are such that they can recognize guidelines specific of the DDK, SK, TK and OK (see section 2.2 for more details). Agents, specifically, can be considered as the smallest entities (and this is the sense of the term “atomization”) acting in a business domain which are to embody elements and directives coming from all levels of the AK (refer to section 2.2 for a complete characterization of Application Knowledge).

An agent is not expected to be the smallest system’s developed component in absolute terms, but it is the smallest entity in the system with the most complete minimal representation of the AK (intended as DDK + OK (core of PK) + TK + SK) necessary for the achievement of its goals. The atomization of the system components behaviour leads us to consider complexity as a natural emerging phenomenon; as a mere derivation of entities interactions.

2.3.3 Realization in K4Care Project

In this section I will report the major aspects concerning the developments (based on the principles just introduced) realized within the K4Care project, trying to privilege the same differentiation of classes illustrated in previous Section 2.2. The concrete design of K4Care AK model was based on the specific domain’s resources and processes of the Home Care, used in the clinical institutions of K4Care medical partners.

**Domain Description Knowledge (DDK)**

The requirements analysis necessary to provide agent-based HC services pointed out two main difficulties, as it can be acknowledged in [11].

On the one side, numerous different types of professionals, with very diverse skills and knowledge, had to be properly coordinated in their activities in order to offer an efficient HC assistance.
It was necessary, first of all, to model medical and social practitioners, as well as care givers and relatives of a patient (as well as skills, permissions, responsibilities, and so forth), as formally described in the project by the *Actor Profile Ontology – APO*.

<table>
<thead>
<tr>
<th>Table 1: List of Main Actors in K4Care</th>
</tr>
</thead>
<tbody>
<tr>
<td>- family doctor – FD</td>
</tr>
<tr>
<td>- physician in charge – PC</td>
</tr>
<tr>
<td>- head nurse – HN</td>
</tr>
<tr>
<td>- nurse – Nu</td>
</tr>
<tr>
<td>- specialist physician – SP</td>
</tr>
<tr>
<td>- social worker – SW</td>
</tr>
<tr>
<td>- specialised care giver – SCG</td>
</tr>
<tr>
<td>- informal care giver – ICG</td>
</tr>
</tbody>
</table>

Next to all physical actors, there is one which deserves a further introduction, consisting of a logical grouping of practitioners: the Evaluation Unit – **EU**. An EU is composed of a FD, a PC of the HC, a HN, and a SW whose objective is to assess problems, define intervention plans, identify the proper procedures, evaluate the results and verify the achievement of the goals defined. On the other side, the typical Home Care patient – **HCP** – is an elderly patient with co-morbid conditions and diseases, cognitive and/or physical impairment, functional loss from multiple disabilities, and impaired self-dependency. Modelling of very general diseases and their health care treatments were formally described within the project by the *Case Profile Ontology - CPO*.

It is particularly relevant to stress that the majority of elderly patients usually have multiple co-morbidities and medications, which must be taken into account. The presence of different medical conditions on a given patient complicates the treatment enormously, and makes it very difficult to follow standard clinical practice guidelines. A recent paper argues that the combination of recommendations for the treatment of patients with co-morbidity is one of the ten grand challenges in clinical decision support [62]. The following Figure 7 and Figure 8 report DDK in a graphical...
dimension, in which HC domain concepts result formalized in an ontological interconnecting network.

According to GAIA (see Section 1.4.1 for more details), individuals (agents) I developed may have in a society one or more roles, acting as a single or in a group. A role, in turn, is characterized by the actions that can be executed, the resources that can be accessed, the permissions received, the obligations to be honoured, the procedures to be followed, and so forth.
**Service Knowledge (SK)**

In K4Care model, HC processes are represented and executed via agent-based services. A service is a specification of complex interactions among actors (agents) of a certain type involving the execution of one or several actions according to some conditions.

I have circumscribed the following concepts:

- **Service**: it is an abstract representation of a complex activity, typically accomplished via the collaboration of several actors (green rectangles in Figure 9, indicated by an “Sx” label). It may have several instantiations, which are called procedures.

Different procedures instantiating the same service may be used for different localisations, e.g., in different countries or medical centres. In this manner, each medical centre can implement a service taking into account both its particular point of view and the effective availability of resources;

![Figure 9: Service Knowledge description (pseudo SDA* representation)](image)

- **Procedure**: it is a formal description of a set of tasks (nested services or atomic actions) organized in some workflow, be it sequential, parallel, or with conditional branches (orange rectangles...
in Figure 9, indicated by a “Px.y” label). The procedure may be the instantiation of a medical or management service in the medical centre. The workflow control structures are described in SDA* language (see Sections 3.5 and 3.6 for a more detailed explanation).

Procedures are created by humans, usually medical centre managers or physicians in charge of Home Care units;

- **Action**: it is an atomic activity that can be executed by an actor singularly (third parameter of a Task – see after for more details). The set of actions that an actor is able to perform can be considered as his/her skills or abilities. These actions are described in a standard manner using an application ontology, which provides independence between the workflow description of HC activities and the concrete specification of the HC model of an organization.

A very simple service can be even composed of a single action, indeed (Right column in Figure 9);

- **Task**: it represents an execution step in a procedure or intervention plan. The task T is described by a 4-tuple (light-cyan rectangles in Figure 9):

\[
T = \langle \text{subject, object, service/action, document} \rangle
\]

The subject is the type of agent (e.g., a nurse, a social worker) who has to initiate the nested service (which, at the same time, is associated with a new procedure with several tasks) or execute the action. The object is the actor on which the service is expected to be executed (e.g., a specific patient). The results of the execution of an action are included in the given document. All actors must reflect their activities into documents.

**Technological Knowledge (TK)**

In addition to domain conceptualization and modelling, it has been necessary to determine, isolate, formalize and represent concepts deriving
from other development contexts, necessary to build the agent’s internal representation of the environmental infrastructure.

Figure 10: Technological Knowledge description (Ontology)
I have organized technological assets, such as MAS components, as illustrated in Figure 10. They have been categorized according to the specific aspects of the development process, and grouped into hierarchical ontology sets (further detailed in section 3.2.1): FIPA, GAIA and JADE ontology. In particular, the Technological Knowledge model of the K4Care highlights the relations I have recognized and established among the three different ontology levels, missing until then in agent-oriented literature.

I have given particular importance to the associations reported in following Table 2, which have been defined (conceptual links in Figure 10) in order to enable a higher level of automation in agent code generation.

<table>
<thead>
<tr>
<th>Table 2: Major Relation in K4Care TK</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 A GAIA Activity with a FIPA Action;</td>
</tr>
<tr>
<td>2 A JADE Behaviour with a GAIA Activity;</td>
</tr>
<tr>
<td>3 A JADE Agent with a GAIA Role;</td>
</tr>
<tr>
<td>4 A FIPA Interaction Protocol with a JADE Agent (which relies, in turn, on a JADE Behaviour embodying JADE Interaction Protocols);</td>
</tr>
<tr>
<td>5 A JADE Message with a FIPA Message.</td>
</tr>
</tbody>
</table>

**Procedural (PK) and Objective Knowledge (OK)**

Procedural Knowledge is composed of all the treatments for co-morbid conditions and refers to the need to define personalised treatments to be customised to each specific patient, according to his/her particular medical and social circumstances. Teams of professionals that are in charge of patients can utilize K4Care agent platform to build PK through the so-called Individual Intervention Plans (IIPs). IIPs are tuned to the patient’s specific necessities, taking into account the health care recommendations supported by the clinical evidence acquired in clinical trials and represented in the form of so-called Formal Intervention Plans (FIPs).
A FIP is a very general intervention plan, whose goal is to provide a primary orientation in the health care of specific well-known diseases. Illustration in Figure 11 is an example of SDA-codified PK for the provision of a medical treatment, called *Case Management* (in yellow, decision branches; in cyan, tasks; and in grey-bordered rectangles, document to be filled and/or comments).

![Diagram of Case Management](image)

**Figure 11: Scratch of Objective Knowledge description (Example of Case Management)**

The Objective Knowledge, for an actor (agent) of K4Care, is generally represented by the following two elements:

1) The business logics (explicitly or not) formalized through its behavioural models and atomized into the elementary actions (ontologically introduced in the APO) taken by the agent;

2) The provision logics (described by an SDA model) followed in the coordination of a HC service execution.
2.3.4 Communication Semantics

Agent-based communication is a far more powerful and elegant abstraction than a method invocation within an object. The work we presented in papers [2], [3], [8], [10] and [11] highlights how the agent is an active entity within a system of individual entities, whose interaction capabilities are enabled by their own communication skills, rather than by mere scheduled method invocations.

In general, communication is an articulated process, relying on the transportation and interpretation of messages. It consists (at least but not exclusively) of a common language (chosen by the involved parties and transporting the semantics of communications), roles (initiators or receivers), protocols (intended as the sequence of messages sent in a specific order among the participants to the speech) and communicative speech acts (which clarify the intention of a single message sent).

Agent-oriented communication targets all the above mentioned concepts, necessary to add semantics to the communications, from the ACL language to the interaction protocols typical of a business negotiation. The agent language, seen as a system of rules, offers an inclusive view of the speech as a form of communication, by considering the “communicative act” in terms of the components which comprise it and the functions which can be served through it. For instance, accept a proposal from another agent; agree on something previously stated; cancel an activity; call for a proposal; confirm or disconfirm something; acknowledge the failure of an event to another agent; inform an agent about something; say it did not understand the other’s request; propagate an order/information; propose something to another agent; query whether something is available or not; refuse a proposal of an agent; request for something; subscribe; and so forth.
2.4 Environmental Dynamics through Agent Substitutability

The panorama of all potential dynamics occurring in a business environment cannot be completely predicted at design time. This is due, mainly, to the interoperability and availability of actors and environmental resources, as well as to the evolution of business strategies, strictly correlated with each single actor's objectives.

So, actually, which is the sense of the term "substitutability"?

Not that one of a mere computer component replacement with a new one, to be implemented (or maybe just retrieved) and deployed into the system, and causing most of times a re-deployment of part of the system itself.

*With the term substitutability I rather intended to focus on the business equivalence of already existing entities, thought to be permanently available into the system and assure the expected service continuousness.*

This should be seen as one of the most valuable aspects in the exploitation of agent-based technology: to have the perception (and de-facto the confidence) of the reliability and availability of components continuously acting for the achievement of the initial objectives; the perception to rely on knowledge- and consciousness-driven resources. In MAS-oriented platforms this can be seen as a specialized definition of *reliable and robust systems*.

Refocusing on my research application, we have to point out that K4Care was a large-size, knowledge-intensive system in which agent interaction dynamics had to be formally governed. This reason suggested me to investigate the exploitation of usable and flexible, but at the same time trustable and rigorous methodologies in the evaluation of agent substitutability. I have addressed two approaches: one ontological, based on the semantic equivalence of agent-centred K4Care services definition, and
one behavioural, by evaluating agents replacement through capabilities simulation.

The former has been fully applied both in my research work and in the development process presented in this manuscript. The latter is still in a conceptualization phase and would further require a formal definition of explicit contracts between users and agents, as contemplated, for example, in the theory on Contract-Driven Systems (which I introduced in paper [46]). Accordingly, I will present it in the followings only in theoretical terms.

In HC, despite the availability of general medical guidelines, treatments are mostly personalized to the particular patient case (which is, in practice, unique in its characteristics). Care practitioners have to manage domain actors’ capabilities, in order to model new HC services. Nevertheless, they have to determine a new workflow of actions to be coordinated during a service provision. The capabilities of actors are various and, considering the level of responsibility and the sensitive nature of treatments, the “designer” wants to have the absolute assurance of the attainability and validity of these capabilities.

This problem is even amplified in case of the unavailability of requested actors: physicians will have to investigate available agents, in order to collect “equivalent” behaviours, trying to minimize the “cost” of a possible mismatch (communications, read-write operations, expected inputs) with the capabilities initially required. The equivalence in agents’ capabilities can be of two types: “trusted” (reported in the following as ontological equivalence) or “investigated” (by simulating agent behaviours).

2.4.1 Behavioural Equivalence

My intention was to find a method to validate agent behavioural capabilities, systematically: this required me to introduce an agent-oriented engineering methodology to explicitly define these expectations from an
agent’s activity. In other words, the idea was to consider agent behaviours as tacitly conceptualized agreements, on the basis of which assuring the reliability and validity of agent’s actions.

I have tried to extract and adapt behavioural validation guidelines proper of Object-Oriented systems to K4Care MAS. Undoubtedly there appeared additional adaptation requirements which still necessitate clarification and formalization, as reported hereafter.

One of the most important notions ensuring software component’s correctness to an acceptable extent, is the concept of abstract data types (ADTs), introduced by Liskov and [47], which relies on foundation work done by Hoare [48] and Parnas [49], [50].

Correctness formulas of the form P{Q}R (also called Hoare triples) are a mathematical notation and form the basis for assertions. On the basis of this notation we can introduce concepts such as “preconditions”, “post-conditions”, and “invariants”, to express the correctness properties of methods and classes. The idea of assertions led to the concept of a contract, which binds a method to its clients and thereby imposes obligations and grants rights for both of them. This concept is also called Design by Contract (DBC) and it was initially introduced by Meyer [51].

When trying to adapt contract assertions to the agent paradigm, in any case, different aspects must be taken into account. I have identified the followings as the major ones:

1) Agents are an extension (improvement) of the OO paradigm, in the sense that they manifest, in addition to an object, autonomy, pro-activity, social and learning capabilities;
2) Pure agent-oriented designing approaches do not involve methods-based interactions (as though programming techniques permit it) but the definition of behavioural models, which are triggered by message passing mechanisms;
3) Agents can individually reason over the environmental elements, through ontology;

4) Agents cannot be obliged to follow a contract in the strict sense introduced earlier, as they are capable of “inference”, rather than being a “simply reactive” object in the system.

This introductory analysis led me to the following considerations:

1) DBC theory can still be adapted and applied to Multi Agent System design, but pre-conditions, post-conditions and invariants concepts should be referred to the behavioural mechanisms of an agent. That is, agents have to reason over preconditions in order to grant the verification of expected contract-based assertions (invariants and post-conditions);

2) An agent does not own environmental parameters within its class, but rather these should be considered as an external resource accessed by the agent;

3) Each agent can be seen as an “intelligent” black-box, whose behaviour can even remain unrevealed to the user’s understanding (Q terms in Hoare triple), but whose effects are clearly constrained and validated by the definition of a contract (P and R terms in Hoare triple).

On the basis of all previous considerations, I feel to affirm that validating an agent’s behaviour would finally imply the verification of the contract conditions (agreement) between the two parties involved: the user and the agent (or even two agents). At the moment I am still investigating on the proper formalism to be adopted in the definition of agent-oriented contracts, evaluating DBC guidelines contained in [46].

2.4.2 Ontological Equivalence

Ontology is a standard AI knowledge representation mechanism [52] [53]. Its main components are a set of concepts (or classes) C, which are
taxonomically related by the transitive “is-a” relation and non-taxonomically related by named object relations \( \{ R \times C \times C \times \text{String} \} \).

In K4Care platform ontology represents the knowledge assets and the catalyst for agent behavioural models, as well as the foundation for the agent code generation. A necessary and sufficient class definition in an ontology, which consists of restrictions on a set of properties, implies that an individual which satisfies the property restrictions belongs to the class. This is the philosophy of the first approach I proposed in validating agent capabilities, as illustrated on Figure 12.

The designer of a new service could need, for instance, to invoke the intervention of a family doctor, who could results (temporarily or permanently) unavailable. On the basis of the analysis of object properties, another actor could be addressed (physician in charge), which evidences the same capabilities of interest (from the APO: “does action BO.01” or “initiates service for brochure consultation”).

This would lead to an ontologically equivalent actor (agent) replacement for the provision of the same services in the platform, as introduced in the followings.
2.4.3 Agent Replacement Methodology (ARM) Preliminary Guidelines

In this section I finally outline a practical scenario of general guidelines to be eventually followed by the practitioners who are involved in the provision of a new K4Care agent-based medical service.

Let us start by saying that the ARM process is naturally characterized and triggered by the unavailability of one or more agents embedding the required capabilities (in terms of actions).

The methodology should comprise the following minimal steps:
1. Realization of a requested agent’s activity unavailability;
2. Analysis of the main objectives for the new medical treatment service;
3. Attempt to capture goals’ hierarchical importance by a parameterized rating (time, physical location of agents, overall capabilities, and so forth);
4. Decision on the best approach to adopt (ontological, behavioural, or even both);
5. Application of parameterizations to agents resulting available;
6. Analysis of results obtained: user (physician) will be always asked to provide a decision of responsibility on the patient treatment;

This methodology is to be intended as an additional decision support tool for practitioners, to be coupled with the major knowledge and experience of physicians (who remain the ultimate responsible of the intervention plan): ARM does not aim at substituting their invaluable attitude to handle unexpected treatment complications which should arise.

2.5 Long-term Interactions as Complex Computational Result

Agent metaphor revolutionizes the concept of computing, becoming “something” that happens through communications between interacting
entities. In this light of radical re-conceptualisation of computing, the network is to be considered as the “real computer”. Agent paradigm enables computing as an activity which is inherently social, rather than solitary, leading to new ways of conceiving, designing, developing and managing computational systems. A clear example of the influence of this aspect is the model of software as a service in Service Oriented Architectures (SOAs), as presented in [64] and [65].

According to this, IT applications are no longer conceived as monolithic, functioning on a single machine and for single user applications, or distributed applications managed by a single organisation, but instead as societies of components, societies of agents. Agents provide services to each other and they may not all have been designed together or even by the same software development team, as results clear, for example, from our works published in [4], [8] and [11]. They may be created, operate and be decommissioned according to different time scales; they may enter and leave different societies at different times and for different reasons; and, finally, they may form coalitions or virtual organisations with one another to achieve specific temporary objectives (see [9] and [10]).

Agents of a MAS (and their services, consequently) may be owned and managed by different organisations, and thus have access to different information sources, have different objectives, and have conflicting preferences. Health care management systems spanning multiple hospitals or automated resource allocation systems are indeed the best examples in such a sense. Nevertheless, agents are usually self-activated when certain conditions hold. Even these preconditions may be distributed across components, so that action by one component requires prior co-ordination and agreement with other components. The intelligence of an agent may autonomously lead it to even undertake self-assembly of software and systems (as it will be explained in detail in the case of a new IIP semi-
automated creation, in section 3.6.1), so enabling adaptation or response to external or internal changing. Such systems resemble those of the natural world and human societies; ideas from biology, statistical physics, sociology and economics play an increasingly important role in agent-based computing systems, indeed. Starting from the abstraction of an agent which is capable of flexible and autonomous actions in a dynamic environment (usually containing other agents), we have encapsulated these software entities and we have demarcated the society in which they operate (e.g. Multi Agent System). In the sense that it is a new paradigm, agent-based computing is disruptive. As outlined above, it causes a re-evaluation of the very nature of computing and computational systems, through concepts such as autonomy, coalitions and ecosystems, which make no sense to earlier paradigms (see Section 1.2.1 for further explanation). In the light of the previous premises I feel definitely to affirm that an agent-based computation is not seen as a mere functional transformation of inputs and outputs, but as the result of massive agent interactions.

In other words, we may more formally define the following relation:

\[ ABO \neq \Phi(ABI) \]

(Where \( ABO \) = Agent-based Outputs, \( ABI \) = Agent-based Inputs and \( \Phi \) = transformation function)

Not to be intended passively, but rather as a sum (intended here as result) of agent interactions:

\[ \text{ASO/S} = \sum_{j=0}^{N} I_j(A_1, A_2, \ldots A_m) \]  

(Where \( \text{ASO/S} \) = Agent-specific Objective/State, \( I_j \) = \( j \)-th agent Interaction and \( A_m \) = \( m \)-th Agent involved in a given interaction).

It is interesting to note how the previous formulation still holds even in case of boundary conditions:
1) **N=0 (independently of m)**: no interaction occurs among agents. In this case ASO/S can be interpreted as the result of a specific agent’s own activities;

2) **N≠0 and m=0**: a given agent interacts with itself N times (we can consider it as the reflexive property of the ASO/S);

3) **N≠0 and m ≠0**: we are in the generic case of multiple agent-based interactions within the system.

In the light of these considerations, teams of agents are intended to work together in order to achieve (potentially common goals); abstraction hierarchies strengthen modelling in a more effective manner and from different perspectives; intermediary agents acting as a single point of contact for a number of agents lead and coordinate complex negotiation processes.

### 2.5.1 Realization in K4Care Project

The work presented in [8] introduces the main development guidelines adopted in the realization of K4Care agents’ long-term interactions (refer constantly to next Figure 13).

![Figure 13: K4Care Architecture](image-url)
Any actor (e.g. physician in charge, nurse, patient) interacts with the system through a web-based application and is represented in the system by a **permanent agent** (Agent Actor) that knows all the details about his/her roles, permissions, pending results, pending actions, and that manages all queries and requests coming from the user or other agents.

In order to allow information exchange between agents and human-actors there has been built an intermediate bridge constituted by servlet and Gateway Agents (GAs). The servlet is connected with the browser user session and creates a GA each time an actor logs into the system. GA’s mission is to keep a one-to-one connection with the corresponding permanent agent. Agent-based modules embed all the system logics. Agents act semi-automatically, in the sense that several actions, such as exchange of information, collection of heterogeneous data concerning a patient (results, current treatment, next recommended step, past history), or the negotiation of a medical visit, can be performed by the agent without the intervention of the user. Of course, other actions such as the confirmation in the formation of an evaluation team or the evaluation of some result received from a laboratory require the user validation.

![Figure 14: Agent coordination during a service execution](image-url)
Fundamentally, K4Care MAS is composed of: actor agents, representing practitioners and patients who use the Data Abstraction Layer in order to access their data; a servlet; several gateway agents that allow to exchange information between the MAS and a web-based application; SDA* agent execution engines that permit to enact procedures and recommend next actions to be taken, according to the patient’s current state.

With the help of the illustration in Figure 14 I will shortly describe how a service is executed and coordinated by agents. The figure shows the data-flow involved in the provision of a specific service (hereafter reductively as S.3):

- Let’s suppose that a Family Doctor (John Smith, in the specific fiction example) logs into the system and requests service S.3 to patient David Jones - DJ (step 1).
- The request is then received by the servlet that creates the corresponding Gateway Agent (GA) for John Smith, which will enable interactions with his actor-agent (steps 2-3).
- John Smith agent retrieves the appropriate procedure for service S.3 and then forwards it to the SDA* Execution Engine (SDA-EE) (steps 4-6).
- The first time the procedure is required, a new instance of the SDA-EE is created (a single SDA-EE instance is associated with a single procedure, which refers to a single patient, only). The SDA-EE receives the FIP state in which the patient actually is and then, recommends the next action to be performed. At a certain point of the procedure, the SDA-EE could recommend executing the action “A 27” to the patient, and that the action should be undertaken by a Social Worker (step 7).
- John Smith agent should then look for an agent capable of performing the specific action. Finally, a Social Worker (hereby Peter Brown) can perform this task (step 8).
- Peter Brown’s personal agent, actually, receives the request to perform action “A 27” on David Jones, and stores it (steps 9-11).
- When Peter Brown logs into the system, he will be notified, by his agent, of all pending actions he has. After selecting the pending action on Mr. Jones (steps 12-14), his actor-agent retrieves from the repository the document to be finalized in action “A 27” (steps 15 and 16).
- This document is forwarded to the social worker through its agent, gateway agent, and servlet (steps 17-19) and is finally shown in the browser.
- After completing the document, all the data is stored in the EHR of the patient and it can be collected in the next step by John Smith (steps 20-24).

2.5.2 Intervention Plans

The Comprehensive Assessment is one of the most articulated and crucial medical services provided by K4Care. Its application will result in the syndromes, symptoms and diseases evidenced by a patient. These elements (concepts) are associated with a FIP, which is a very general intervention plan. A single FIP, anyway, is not directly applicable to any real treatment or intervention, due to the uniqueness of the patient’s physical conditions and medical history. It is a usual practice, on the contrary, to recognize at the same time several diseases, which can lead to a very complex and articulated assessment. The platform is able to automatically retrieve all FIPs involved, whereas the EU members are asked to combine and personalise them, giving form to a customized and personal IIP.

The automation of the merging process for different FIPs is not a trivial task and implies an accurate analysis of the objectives (this due to the complexity and heterogeneity of the physical parameters and the HC domain’s variables). The SDA* description of an IIP is processed in order to guide the execution of actions and interactions among agents. The
execution of administrative processes, codified also in the SDA* language in the form of procedures, follows exactly the same pattern. The enactment of an IIP is a complex task which requires the interaction of both agents and humans. This service can be requested by the Physician in Charge of the HC unit. The person who is responsible of the management of the execution of an IIP is usually the Head Nurse. Note that each Home Care unit (each consisting of a separate K4Care platform) has only one PC and one HN, but it may have many Family Doctors, Social Workers and other types of actors (nurses, specialist physicians, informal care givers, etc.).

I am going to explain in more detail the previous scenario through the example reported in Figure 15:

1. **Step 1**: the Physician in Charge (PC) logs into K4Care platform using Web interface. The information is forwarded by the servlet to the Gateway Agent (GA) coupled with PC. The agent transmits the login message to its corresponding permanent Actor Agent (AA). The AA registers the login event and records the address of the actual GA. In
this specific case, the PC further desires to request the execution of an IIP for a patient called David Jones. Through the GA, a request message to start the IIP is sent to AA containing both the unique identifier of the patient and IIP to be started;

- **Steps 2–3**: at this point, AA is asked to find out who is capable of managing the required task. The agent retrieves this information from K4Care knowledge bases (through the Data Abstraction Layer, which, in turn, accesses the APO. See DDK description in section 2.3.3 for more details). In this case, the supervision of an Individual Intervention Plan execution is responsibility of a Head Nurse;

- **Step 4**: the AA of the PC contacts the AA of the Head Nurse, which stores the request to begin the execution of an IIP. AA then forwards the IIP triggering message to the HN, together with the information relating to the initiator’s identifier;

- **Step 5**: when the HN logs into the system, her corresponding agent gathers all her pending actions (actions that have been requested by other users while she was off-line);

- **Step 6**: the HN selects the pending action (“execution of the IIP of Mr. Jones”). HN then confirms the execution of the IIP by sending a proper message to the AA;

- **Steps 7, 7a, and 7b**: The AA of the Head Nurse then dynamically creates an SDA-Executor Agent (also called SDA Execution Engine in Figure 14), and sends a message to start the IIP to the just created SDA-E Agent containing both the identifier of the IIP and the patient. The SDA-E Agent requests the corresponding SDA* representation of the IIP from the EHR of the patient through the Data Abstraction Layer;

- **Step 8**: when the SDA-E receives the IIP, it is ready to start the execution by sending to the AA of the HN the information contained
In each of the elements of the plan (i.e., states, decisions and actions). In the example in Figure 15 the first step is the establishment of a drug therapy by a Family Doctor. Actions to be performed can be of either sequential or parallel nature, according to the SDA* description. The AA of the HN dynamically delegates these tasks to the appropriate AAs (as described by the IIP) and waits for their completion. When a task is completed, the SDA-E is notified about it, so that it can send the next task to be performed.

2.5.3 Scheduled and Time-Constrained Activities

As previously mentioned, the interpretation of SDA* structures results in the definition of tasks. A task description contains a subject (see Section 2.3.3), which is the type of agent expected to execute the action, or initiate a nested service. Actions are to be executed atomically by the corresponding actor; nested services imply the automatic initialization of a new service, which will be composed, following the formalism previously described, of several SDA* components. The execution protocol is the same as in the case of an explicit initialization of a service by an actor.

When the AA of the HN receives a task, its primary objective is to assign it to a specific actor (who is represented by an AA in the system). Different types of assignments can take place:

a) Dynamically, when the task execution time arises.

The most suitable agent could be selected by negotiation, for instance applying FIPA Contract Net protocol\(^7\), allowing each agent to make a proposal when a new task is available, and making the assignment decision dynamically at execution time. The advantages of this solution consist in the fact that 1) we exploit the benefits of the agent paradigm and 2) the medical staff does not have to assign

\(^7\) http://fipa.org/specs/fipa00029/index.html
actors to branches which are not actually executed. This is the assignment model implemented in K4Care platform;

b) \textit{Statically}, when the IIP execution begins.

In this case, agents are preliminary assigned to each task of the entire IIP. The advantage of this solution is that once the agents are assigned and the IIP execution is started, there isn’t any other overhead, and the SDA-E agent can talk directly to the relevant agents. However, in this case we have to assign agents to actions which may not be executed, as they are in a decisional branch which will not be selected during the execution. This is the very inconvenient for the medical staff, because they are asked to assign actors hypothetically.

Sometimes the actor assignment is a simple operation because both the patient and agent type univocally point to specific agent instance. If the action has to be performed by a PC, a FD, a HN or a SW, then the assignment is made automatically to the patient, by referring to the corresponding person in the Evaluation Unit. Otherwise (e.g., if a nurse is needed) the HN has to select a concrete person to perform the action. The platform provides the Head Nurse with graphical facilities, so that she can quickly select the actors currently available. The concrete list of actors is dynamically retrieved by the system according to: the AAs currently running, their associated actor-type and, consequently, the list of actions they are entitled to execute (information stored in the APO).

It is important to mind that, as the system supports the personalisation of an actor’s profile, this has to be taken into consideration when assigning the action or proposing the actor as a performer. Concretely, an actor may decide, at runtime, not to perform some kind of actions. This requirement is processed by means of the personalization of an individual
copy of the APO. Once the changes are validated by the Physician in Charge, the new personalised profile is immediately taken into account. From the point of view of the action assignment, the system automatically requests each AA whether the pending action may or may not be performed in that specific moment by the concrete actor. Now, recalling the example depicted in Figure 15, we finally have these additional execution steps:

- **Step 9**: once the AA of the HN (Jane White) has been acknowledged that the FD (Peter Brown) is assigned to action “Establish a Drug Therapy”, it sends a message to his associated AA, requesting the execution of the action (step 9a). The request embeds the action identifier, the patient to whom it is to be applied and the identifier of the document to be filled out. The document identifier is retrieved (step 8b) as a result of a previous request to the APO (via DAL), by specifying the action to be executed (step 8a). When the FD logs into the system, he will find the request in his list of pending actions (step 9b);

- **Steps 10, 11, and 12**: after selecting this action (“Establish a Drug Therapy”), a message is sent by the Gateway Agent to his AA. Then, the AA retrieves (via DAL) the document to be completed, in order to reflect the performance of the action in the real world;

- **Steps 13, 14, 15, and 16**: FD’s AA finally forwards the document to the user interface (through its Gateway Agent), so that the human user can fill it out (steps 13 and 14). When the document is completed, the AA will store it in the EHR of the patient (step 15) and sends a confirmation message to the AA of the HN, indicating that the action has been successfully performed (step 16). In the end, the AA of the HN can request to the SDA-E the next element in the IIP execution (which may be again a state, decision or action).
Another factor increasing the complexity of an SDA model is the presence of temporal constraints in the execution of agent actions (e.g., after the execution of the treatment for Pressure Ulcers, wait one month and then perform a Physical Examination to the patient). In those cases, the request for action execution will be stored and timed, waiting for the expiration of the specified period of time, before sending it to the assignee-actor.

As we can perceive, even through a not so complex example as was the previous, continuous reliability of actor agents is crucial for the provision of a HC service. In this context of long-term HC treatments to be guaranteed, the ARM methodology I proposed and described in Section 2.4.3 for agent substitutability can demonstrate an effective contribution against the critical occurrence of service denial in multi agent-based architectures.
Chapter 3

MAS Development and Maintenance

Although there have been substantial advancements in software engineering methods and tools during the past twenty years, requirements engineering still remains a key problem in the development of complex, software-intensive systems. According to [54] and to the report of the Defense Science Board Task Force on Military Software, we can affirm that still nowadays one of “the hardest part of the software development task is the setting of the exact requirements” [55]. One of the primary sources of continued difficulty is the lack of early requirements validation. Validation of requirements is problematic because requirements often are not well understood prior to development, change frequently during development, and multiply as a result of development.

An early validation is of great importance, indeed, as many requirements errors are passed undetected to the later phases of the life cycle, and correcting these errors during or after implementation have been found to be extremely costly. Early defect fixes are cheaper than late defect fixes, and the early requirements and design defects typically leave more serious operational consequences. Clearly, better techniques are needed for early validation.
The waterfall life cycle model, which requires a complete pre-development requirement specification, also contributes to these problems [57] [58]. For large, complex systems it is difficult to completely specify requirements in advance of, and independent from, design and implementation. The assumption that it is possible to create a complete specification prior to development has been a major cause of problems in many projects, due to frequent changes to the specification during and after development.

Validation of requirements early in the life cycle is one of the key issues in software development because a failure in the validation requirements can result in frequent and expensive changes in later life cycle phases.

I strongly believe that designers cannot, with any amount of effort and wisdom, accurately describe the operational requirements for a substantial software system without testing by real operators in an operational environment, and iteration on the specification. The systems built today are just too complex for the mind of man to foresee all the ramifications purely by the application of the analytic effort and imagination.

3.1 Agent Software Fast Prototyping

Due to the difficulties evidenced before, prototyping has been discussed in literature as an important approach to early requirements validation [60] [61]. A prototype is an enactable mock-up or model of a software system which enables evaluation of features or functions through the user and developer interaction with operational scenarios.

Prototyping exposes functional and behavioural aspects of the system as well as implementation considerations, thereby increasing the accuracy of requirements and helping to control their volatility during development.
In the following years since the waterfall model was developed, computer science has come to recognize that setting the requirements is the most difficult and crucial part of the software building process, and requires further elicitation iterations between designers and commissioners.

In best modern practice, the early specification is embodied in a prototype, which the intended users can themselves drive in order to see the consequences of their imaginings. Then, as the design effort begins to yield data on the cost and schedule consequences of particular specifications, the designers and the users revise the specifications.

Agent-oriented prototyping further strengthens the competitiveness of such a software development process, by adding to it the advantages hereinafter evidenced:

1) An immediate global view of the whole system’s behaviour and functionalities;
2) A highly reduced misunderstanding on the domain concepts (who, what, why, when and how) between IT designer and commissioner;
3) An adherent representation of individuals society: domain actors will still be actor-agents of the MAS; domain rules will be codified into behavioural rules of the MAS; domain actors’ roles will be mapped into a methodologically accepted formalism (GAIA agent roles), and so forth;
4) The opportunity to observe the evolution of actors’ interactions towards new scenarios, unpredictable or missed until then;
5) The possibility to extend the overall behaviour of the system (as we will see in next sections), by enhancing single agents’ capabilities and without terminating activities of the entire agent population.
3.1.1 Governing Complexity through MAS Cyclic Refinements

The requirements for a system or a class of systems are gathered in most cases (if not all) in an evolutionary fashion. Requirements knowledge cannot be considered ever totally complete, but rather evolves over time as new requirements are identified, existing requirements are expanded, and obsolete requirements are discarded.

On the basis of the new requirements acquired, incremental prototypes are to be designed and verified (incrementing the costs each time) until the creation of an operational architecture, which will be finally transformed, through codifications, integrations and tests, into a deployable product. This is the basic conceptual foundation of the *Spiral Model* in classic software development (see Figure 16).

![Figure 16: Spiral Model in Classic Software Development](image)

The MAS-specific development model I propose in Figure 17 has been conceived starting from the classical model shown in Figure 16.
I have highlighted and summarized the most important differences between the two models in the following considerations:

1. In a first analysis, the third phase (quadrant) of the spiral (Development and Test) has been specifically oriented to Multi Agent Systems realization. Consequently, all inherent agent-specific concepts and technology have been applied:
   a. Analysis and design methodology (in my case GAIA);
   b. Codification, implementation formalism and support (JAVA, FIPA and JADE).

2. The second crucial aspect is that the model does not terminate with the release of the software product. The behaviour of an agent, as we have seen, is a “particular” sequence of elementary actions, defined to be executed for the achievement of an objective. All these elementary actions,
theoretically, can be combined together, by a proper formalism (as detailed in section 3.5), in infinite ways, giving form to new behavioural models in the MAS. These new behaviours (what I call MAS Behavioural Evolution in Figure 17) are conceived to meet new requirements and achieve new goals, which could (often) arise in the post-deployment phase of the end-MAS. Nevertheless, it is worth to stress that, according to this model, MASs (relatively to the services it provides), can be directly extended by the end-users, whose orchestrations can permit to reformulate actors’ activities in the definition of new MAS behaviours. This very relevant aspect allows end-users to capture and embody into the system new application knowledge.

3.2 Semantic Representation of a MAS

The more abstract is a formalism the more powerful is its modelling expressiveness. Agent paradigm seems to naturally have this peculiarity: terms are a natural extension of human perception and so it is the semantic relevance of their representation. Ontological concepts like society, individual, actor, behaviour, action, responsibility, permission, and so on, are mapped to very similar elements of the MAS architecture, augmenting the level of elegance and reliability of software development processes. This is the leading aspect in following section’s contents.

3.2.1 Ontology Based Approach

The Home Care model created during the K4Care project defines the actors of the HC domain (e.g. physicians, nurses, social workers, and patients), their roles and interactions. Concepts of the model have been formalized using ontology as a standard AI knowledge representation mechanism.
The chosen language for ontology was the OWL\textsuperscript{8} (Ontology Web Language) from the World Wide Web Consortium\textsuperscript{9} (W3C), and we used Protégé-OWL\textsuperscript{10} plug-in to create and manage OWL ontology. They represent the knowledge assets of K4Care application and the catalyst for the agents’ behavioural model, as well as the fundaments for the agents’ code generation. In this manner domain knowledge description was separated from the software realization, assuring a high level of interoperability and independence among elements of the system.

I have divided application ontology into different sources, in relation to their application coherence, according to the description available in section 2.3.3. The ontology is composed of several inner elements, independent of each other, except for K4Care global one, which includes all the other ontology, as well as all the cross-references necessary to relate concepts coming from the different elementary ontology.

The application ontology in K4Care (characterizing the AK previously introduced in section 2.3.3) has been structured as follows:

- **Domain ontology** divided into **APO** and **CPO**. As already mentioned APO represents the profiles of subjects in K4Care model (healthcare professionals, patients and social organisms) and contains skills, concerns, and aspirations of the people that they represent, together with the definition of healthcare services that those people offer to or receive from K4Care model. CPO represents all the relevant medical concepts (e.g. symptoms, diseases, syndromes). Domain ontology describes “know-what” knowledge about actors accessing K4Care model and pathologies K4Care model gives support to;

- **FIPA ontology**. It defines the general conceptualization for MAS development standardization. It represents the “know-what” knowledge of a

\textsuperscript{8} http://www.w3.org/TR/owl\textsuperscript{features/}
\textsuperscript{9} http://www.w3.org/
\textsuperscript{10} http://protege.stanford.edu/
MAS (messages, encodings, communication languages, and so on) and “know-how” knowledge of their interactions (communicative acts, interaction protocols, communication ontology, and so forth);

- **GAIA ontology.** It includes the ontological description of the MAS development methodology adopted in K4Care project, with particular attention to concepts such as role, responsibility, permission;

- **JADE ontology.** It expresses implementation and deployment concepts that must comply with the elected MAS of the project, i.e. JADE (agent-types, behaviours, containers, mobility, and so forth);

- **K4Care ontology.** This ontology was required in order to model the inevitable ontological cross-references: agent capabilities, for example, are expressed in terms of “actions” in the APO; these are then considered as “responsibilities” or “permissions” in GAIA; the behavioural logic of an agent inherent to its capabilities is expressed in JADE in terms of “behaviours”, which can contain, in turn, FIPA compliant “interaction protocols”.

### 3.3 Agent Code Generation

In K4Care project I have automatically generated agent-oriented code deriving it from the application ontology. The accuracy and the level of details of code are directly proportional to the formalism adopted to describe the domain model. Particularly important is the representation of the agent capabilities, which consist of a formal representation of their business logic.

The objective of the automation process was to analyze, recognise and extract the entire MAS-compliant agent-oriented information: we had to take into account, first of all, the implementation requirements related to the elected MAS structure (agent class definition, protocols, message structures, message contents, behaviours class). In order to guarantee
flexibility in the ontological traversing and parsing, and to decouple the conceptualization layer from the parsing one, the introduction of a mapping dictionary was necessary. This thesaurus described to parser the “root” level of concepts where to start collecting agent’s knowledge.

All extracted information has been temporarily represented by an internal model. I have obtained agents code combining together two categories of elements:

- **Fixed and invariable** (keywords related to programming language, JAVA, and the MAS, JADE) – a sort of static context;
- **Derived from the knowledge model** – a kind of dynamic context, necessary to embed expressions which comply with the correct code completion of an agent.

An agent’s capabilities (from domain knowledge) are represented by its actions, listed in the APO: from the paradigm’s point of view, these must be translated in terms of behavioural models compliant with the MAS. The code was built taking into account the following aspect: different behaviours have been dedicated to different skills in order to interact in parallel with different other entities (called Execution Test Engines).

This is a fundamental aspect in JADE agents: even if behaviours can be of different nature (parallel, cyclic, one-shot, and so on), the execution of a specific behaviour inhibits (blocks) the execution of all the others (following the policy of a “behaviours queue”). With regard to this problematic I proposed as a possible solution the instantiation of another identical behaviour at the very beginning of the first invoked. In this way we would grant real parallel interaction invocations of the same nature (a sort of “dynamic behavioural pooling”).

3.3.1 MAS Code Generation in K4Care Platform

The platform chosen for the system prototypes was JADE (as already introduced in section 1.4.1). K4Care agent code generator has been
implemented as a Protégé plug-in, which accesses domain ontology via Protégé’s OWL API and shows a user interface with fields specifying both deployment parameters (JADE home, JAVA home, output directories, output package names and so forth) and facilities to generate and compile agents’ source files. Nevertheless, the plug-in is able to start JADE agent platform, launch test agents for each actor type (the number of agents is parameterized), and one (or several) test engines that invoke random actions of randomly selected agents in specified time intervals. This latter functionality can be used to benchmark, stress test the current configuration (hardware, software), measure performance, response times of agents at different loads.

Generated agent code consists of a set of Java classes. These classes can be grouped into three main categories: behaviour classes, message ontology classes, and agent classes. Behaviour classes implement different actions of agents. Message ontology defines the ontology of elements that agents can use within content of messages. Agent classes implement different agent types that start the behaviours and register the related (message) ontology. The code generator relies on the application ontology presented in the previous sections, and also on an implementation of different actions declared in the APO. There are several alternatives in the definition of actions at different abstraction levels (e.g. using business process description language); the plug-in currently expects a Java library in which actions are implemented by simple Java methods.

Method names correspond to action names in the domain ontology so that they can be derived by the code generator. Complex actions based on specific protocols (for example, those ones requiring interaction with other agents within the action body) are implemented by several methods invoked at specific states of the protocol. Activities of agents are embedded in agent behaviours. Possible activities of agents are the actions; therefore behaviour
classes can be generated by iterating through all the actions in the APO. For each action a unique behaviour class has been created, which extended a behaviour schema corresponding to the protocol of the action (as declared in the domain ontology).

Depending on the given protocol, all related call-back methods had to be overridden to perform specific activities at specific states. These methods were generated by the plug-in with body containing the appropriate method calls to the external library of actions. The proper communicative acts of messages, the conversation-id, protocol, ontology fields are automatically set by the generated code according to the given protocol.

Message ontology classes define the concepts that can be used in messages exchanged between agents. According to FIPA standards, request messages contain agent actions (AgentActionSchema in JADE), while result messages contain predicates (PredicateSchema). Agent actions and predicates can contain slots (fields), as well as substructures defined as concepts (ConceptSchema). JADE ontology consist of a set of Java classes (with the appropriate schema super-class), and ontology classes that register these elements in JADE. All these classes are generated by the plug-in, through the domain ontology. Actors are represented in the system by agents. Capabilities of agents are determined by the actions declared for the actor type in the APO. Behaviours are registered with the appropriate message template: communicative act, protocol, ontology (that is, behaviours can only be activated by a dedicated message belonging to the action). In addition to this, agents are capable of handling unknown messages sent to the agent (logged), and administer its life-cycle.

3.4 Behavioural Dynamism

The life-cycle of an agent-based system does not reflect that one of a "traditional" IT platform, which almost ends with the deployment of the
application itself. Someone may immediately argue that even traditional applications usually include activities typical of the post-deployment phase, such as provision of patches and bugs-correcting upgrades, to cite only some. This is, of course, true but not the point which I would like to emphasize.

The nature of agent-paradigm permits the extension of system behavioural capabilities, beyond the mere correction of programming mistakes: the sense is that it permits to reflect, represent and embody into the system, through a suitable formalism, the emerging knowledge (tacit until that moment) manifested by individuals who act (and interact) within the system. And this all might be possible without modifying either the model or the implemented structure of the MAS itself. But how can it be achieved?

The key concept on which to focus our attention is that of agent behaviour and its elementary components. As long as there is an occurrence of interactions between two entities of the system, the MAS will naturally tend to evolve towards a more suitable configuration of its elements (actors), by increasing agents’ capabilities to interact and adapt to the environment and its newer requirements; by expecting agents to manage events which were unrevealed until that moment. The evolutionary path of a system reflects its capability to “coherently” adapt to the environmental changes and expectations.

**Intelligent Systems Coherence**

With the term “coherence” we generally identify:

1) “The quality or state of cohering, especially a logical, orderly, and aesthetically consistent relationship of parts”\(^{11}\);

---

2) “When the parts of something fit together in a natural or sensible way”\textsuperscript{12};

3) “The quality of being logical and consistent, or of forming a unified whole”\textsuperscript{13}.

In human behaviours the above mentioned definitions reach a further level of specialization and differentiation. Human behavioural reactions are complex and many-sided because psychic activity provokes emotional states, which change the objective parameters of a man. The sense of coherence refers to the way individuals examine their lives and their place and how they develop a general orientation that ensures known and predictable environments.

In artificial systems, speaking clearly, the emotional sphere does not represent a fundamental aspect of the system’s performance; we have therefore the opportunity to simplify both the definition and the application of the term coherence. In [6] I introduce preliminary guidelines towards the definition of coherent behaviours in Multi Agent Systems (evidenced in paper mainly in terms of agent’s substitutability). The behaviour of a MAS, like any other IT architectures, is to be referred to the requirements and expectation of its users. The sense of coherence for an agent can be seen in terms of its “determination” in achieving its mentor’s (user) objectives: an agent acts coherently if it substitutes the human actor in the attainment of his/her interests.

There is not any formal and explicit agreement between an agent and its “mentor”: for this reason, coherence can be intended as the effect of a tacit contract between the agent and the human user.

In MASs I intended to relate the concept of coherence to two levels of analysis: a micro level (single components/personal behaviour) and

\textsuperscript{12} Cambridge Dictionary - http://dictionary.cambridge.org/dictionary/british/coherence

\textsuperscript{13} Oxford Dictionary - http://oxforddictionaries.com/view/entry/m_en_gb0978610#m_en_gb0978610
macro level (the system as a whole entity – assembling of components/global behaviour). At a micro level, agents map users’ inner configuration, their nature and the existential attitudes (domain roles, capabilities, permissions, dues, likes, dislikes, personal goals, and so forth). At a macro level, the MAS should provide a society, a market, or any other application agglomeration with those services and loyalties for which it has been conceived and realized.

Keeping coherence implies to adapt to unavoidable environmental changes; adapting often results into decision-making, which, in turn, is a reflex of intelligence. Intelligence, on the other hand, is, in its indefinableness, manifested by an entity (and perceived by the others) through its behavioural guidelines.

Concluding, in my opinion evolution is given by two main factors:

1) Natural occurrences and combinations of agent pre-established interactions;

2) Enhancements and (theoretically infinite) new combinations of agents’ elementary actions (new behaviours).

In addition, coherence holds until an agent system continues to pursue the objectives which have been (explicitly or implicitly) agreed with the users of the system.

3.5 Interactions Evolution in Agent-based Systems

Like most of researchers do at the end of their investigation work, I have also synthesized my own personal view on agents, by coupling their paradigm with another metaphor of excellence: that of a child. This vision directly derives from my personal experiences, on the basis of which I substantially see agents compared, theoretically, to children in a sort of educational methodology towards a behavioural evolution:

as children, at their birth, have pure and incredible potentials provided by their very nature (instinct, adaptation, immediate learning,
emulation, logical derivation, spontaneousness and many others), but need to be trained and directed under the supervision of their parents (i.e. the “experts” who have already experienced the process), so do agents need to learn a “model” of existence, in order to acknowledge independence and survival.

The process of agent behavioural performance is of incremental nature. At the very beginning, they are able of exploiting simple actions, only. But if they received an adequate “education” they are able to execute even the most complex and long-term operations. One of the goals of my investigations was indeed to find a suitable formalism to strengthen agent behavioural representation: starting from simple concepts but with the enormous potentiality of having them combined together in the definition of more complex ones.

The deployment of a Multi Agent System can be seen as a screen-shot of the specific domain configuration at a specific time (coming from the analysis and design phases): well defined actor roles with pre-established behavioural guidelines (and interaction capabilities) represent the start-up core of the MAS.

Anyway, the environment will (naturally and unavoidably) tend to evolve towards different and unknown scenarios, deriving from the evidence of new knowledge emerging in the system domain: the adoption of formalism describing MAS-related activity independent of the development constructs can let us achieve this fundamental objective.

In such a sense agent-paradigm enables formal extensions of the overall Multi Agent System’s behavioural model, without interfering with, and avoiding repercussions on, the application’s deployment cycle. This is a fundamental aspect in the recognition, management and embodiment of new emerging application knowledge, as introduced immediately after.
3.5.1 Introduction to MAS Behavioural Extensions

It is immediately worth highlighting that MAS extensions are expected to be managed and orchestrated directly by the end-users of the platform (practitioners and medical staff, in our case). Because of this reason, behavioural extensions are to be easy-to-apply from the point of view of a not-expert target in computer science. But what do I intend with the term “MAS extensions” and how do they take form?

As already illustrated in previous sections (like, for example, in 2.3.3) a MAS global behaviour is the resulting of different categories of actions taken by actors (agents) in the system, indeed. Those ones of interest, in our specific case, are the actions characterizing the PK (Procedural Knowledge) of the MAS, which represent the core of services provided by the system.

In the modelling approach presented in this dissertation (see sections from 2.2 to 2.3.3), an agent is capable of acting in a specific domain, under well-defined rules, protocols and behavioural policies. While these elements are usually long-termed and hardly modifiable or alterable, users of specific domains (like HC) often need to re-arrange the provision of certain services to comply with customer’s necessities. These significant modifications in the provision of a service will represent the extensions of the MAS’ global behaviour. Let us see immediately what I mean by a simple and practical example derived from K4Care context. Let us suppose that a doctor (a platform actor) wants to provide a patient with a HC treatment service, let us say a “Treatment of Hypertension”, as reported on page 90.

In addition to this, let us suppose that, for some reason, the actual model for the given care treatment cannot be directly applied to the patient, because his/her physical parameters do not properly fit in the provisional conditions. In other words, the doctor needs to modify the IIP (that is, to alter agent interactions which constitute the MAS overall behaviour for the
given IIP) and introduce further activities and controls, or eliminate some of the existing ones, as illustrated in the following Figure 18.

Let us analyse this scenario from the user’s point of view, first. The IIP on the left side of the previous illustration (Hypertension Treatment Diagram) have been applied by the doctor in precedent care cases but now needs to be customized to the actual patient requirements. The general guidelines of the treatment are acceptable for the new patient but they are not completely complying with expectations. The doctor does not require the “Drug Therapy” in point 1 nor the consequent connections to the next diagram block – the yellow rhombus condition check). On the contrary, he/she wishes to modify the treatment, by adding some other necessary steps after the activity of “Life Style Modifications”. Modification will involve the insertion of two new actions – “Urine Analysis” and “Blood Analysis” – provided by domain actors (agent-based) – points 2 and 3 within the red-dashed rectangle – and of all relating links necessary to connect the above mentioned actions to the condition check – points 4, 5 and 6. As we will see, doctors are able to introduce and store IIP models and
their modifications by the exploitation of a graphic tool (IIP Editor), which is part of K4Care platform, as detailed in section 3.6.1.

What happens, on the contrary, within the MAS?

As we already know, domain actors are mapped into JADE (FIPA-compliant) agents and their skills take form as JADE behaviours. The basic components of agent’s behaviours are all the elementary actions it may take (in the domain), which are formally defined in the APO and implemented as agent cores (JAVA classes), and are available to medical staff as run-time modelling entities (through JAVA reflection technology) in care treatments.

When the execution of an action (denoted by 4-tuples in SK description – see section 2.3.3 for more details) is removed from, or added to, an IIP, the MAS behavioural scenario is, in any case, altered, because:

1) Agents previously providing actually cancelled actions are asked not to do it any longer, as some other actor will be delegated to execute them;

2) Agents providing newly introduced actions, on the contrary, will be asked to execute them following the IIP’s coordination logics.

Finally, the MAS will realize behavioural modifications, both on micro- (previous point 1) and macro-level (previous point 2), due to the extensions required at run-time by medical users.

Concluding, I can summarize the precedents steps by saying that IIP orchestrations are to be in anyway:

a) **Formal**, as practitioners act on an accepted knowledge-describing formalism in medical contexts;

b) **Extensions**, because they require a re-arrangement of the MAS over-all behaviour, which will result in the provision of a new care treatment;
c) **Deployment-independent**, because agent’s “native” capabilities remain in-fact unaltered: what changes is their orchestration and external coordination in the provision of a specific HC treatment.

### 3.5.2 Intuitive Representation of Behavioural Core Elements

The basic idea I hereafter propose is to decompose the concept behaviour of an agent into **simpler** related **concepts**, which are at the basis of any agent’s activities and, the more important, can be orchestrated and arranged to create new (or extend previous) behavioural lines, even after the deployment phase.


As already mentioned, the role of an agent is primarily the mapping of its domain user’s profile. Nevertheless, competences of a user in a domain may vary, depending on his/her attitude to learn and improve in the application position, and even due to changes in the regulations of the domain itself (formal variations in user’s permissions, responsibilities, competencies and so forth).

**Question:** “Considering that a MAS may be constantly running and its complexity may be finally governable, how can we extend system agents’ capabilities without interrupting the actual provision of MAS services?”

**Answer:** “We need to decompose the Procedural Knowledge (PK) of the domain into reassemblable simpler concepts, to be intuitively and easily adaptable to agent-based technology”.

The concepts I am talking about (which are the foundations of SDA*) are the **decisions**, the **actions** and the **states** of an agent.
3.5.3 Concepts

I am going now to briefly explain each of the previously mentioned concepts meaning:

- **Decision**: the set of decision variables is composed of all the variables that may be required by human users to choose among alternative actions during the occurrence of MAS interactions;

- **Action**: the set of action variables comprises all the variables that represent the personal capabilities of an entity (human actor in the domain, agent in the MAS) in the system;

- **State**: the set of state variables represent terms that are useful to describe and determine the condition (of any kind - physical, behavioural, etc.) of an entity (in our case a domain actor).

Procedural knowledge in K4Care (both related to administrative and healthcare processes) was represented in a formal notation called States-Decisions-Actions (SDA*), following the explanation given before.

In the followings of this section I show how HC processes can be customised to the case of a particular patient, by leveraging on the formalism I have introduced. Nevertheless, I will show how human users of K4Care (practitioners) can coordinate their activities to execute new personalised treatments for a patient, by extending agents’ behavioural capabilities and, doing so, capturing and formalizing new emerging knowledge in the application domain.

3.5.4 Realization in K4Care Project

Developments of the concepts previously introduced have led us to the following specializations:

a) **States**: from a semantic point of view, a state (or Entry Point – EP) describes a general patient condition in which all the variables of the state hold within the context of the disease (for example, diabetes, complete-initial-assessment, etc). From a
logical point of view, a state is a conjunction of state variables. From a functional point of view, the states of a FIP are the EPs of that FIP or, in other words, the points where the treatment described by the FIP can start;

b) **Decisions**: a decision (or branching point) describes in a FIP a point where the treatment can follow alternative courses of action depending on which variables of the decision hold. From a logical point of view, a decision represents the evaluation to true or to false of the conjunction of decision variables of all the alternatives in the decision. From a functional point of view, decisions allow the FIP to be as general and flexible as to combine several variations in the treatment of a disease, and make the application of these variations depend on the particularities of the patient;

c) **Actions**: an action element (or action block) describes a group of things-to-do which a physician proposes in some part of a disease’s treatment. An action only represents the proposal of medical stuff, whereas the application of the action is “out of the FIP description”. Typical sorts of actions are: recommendations, prescriptions, radiographies, analyses, medical, surgical or clinical procedures, specialist consultations, application of an alternative FIP and so forth. From a functional point of view, actions blocks represent the core elements in a FIP since the final purpose of FIPs is to represent healthcare procedures like the treatment of a disease, i.e. the application of actions. Action blocks are independent of the patient situation. Therefore they use to be preceded either by a state that describes what the situation of a patient should be in order to deserve that action, or by a decision that determines whether the patient fulfils some features or not, and according to
that decide if the action takes place or not. Empty actions have the meaning of doing nothing, but since this is the same as not having the action, the SDA* model does not permit empty actions. Flowcharts have been used to represent FIP in a graphical way. Figure 19 shows how states, decisions and actions are represented in this sort of formal flowcharts.

![Elements of an SDA* Model](image)

**Figure 19: Elements of an SDA* Model**

The correct combination of states, decisions and actions compose a FIP in an SDA* model, and is made by means of connections.

### 3.6 Modelling Individual Intervention Plans

As introduced in section 2.3.3 when talking about PK, there exist many standard specifications of medical guidelines (FIPs) associated to concrete pathologies, which can be codified in SDA*.

Generic recommendations may not properly fit in the treatment of a concrete pathology in a specific patient (e.g. the co-morbidity of patients introduces inter-actions among guidelines). In order to solve this situation, K4Care supports the definition of Individual Intervention Plans. An IIP is a personalised view of one or several FIPs to be adapted to the pathology of a concrete patient. Practitioners are the ones who have the capability to synthesize and apply new medical knowledge necessary to new care treatments, whereas agents are the technology which can best adapt to these knowledge-intensive environmental fluctuations. From the medical point of view, the creation and management of IIPs follow a complex procedure (as shown in next Figure 20), controlled by the Evaluation Unit (EU) and centred on all the assessments results.
The results of a Comprehensive Assessment consist of syndromes, symptoms and diseases of the patient. These entities are associated with a FIP, which is a very general intervention plan. A single FIP, anyway, is not directly applicable to any real treatment or intervention, due to the uniqueness of the patient’s physical conditions and medical history. It is a usual practice, on the contrary, to recognize at the same time several diseases, which can lead to a very complex and articulated assessment.

3.6.1 Preliminary Guidelines towards IIP Semi-supervised Definition

Making the point on the key elements, let us recall that:

- **Rectangles** representing tasks to be executed in a FIP are all agent-taken actions (agent capabilities);

- **Circles** expressing conditional reasoning are the evaluations of an agent’s inner state (environmental representation variables);

- **Rhombuses** dictate the conditions to be verified.

**Figure 20: Definition of an Individual Intervention Plan**
A Comprehensive Assessment represents the very basic process thanks to which members of the EU are enabled to reason and react on the patient’s state and conditions. Once all the results of this service are available, the EU members analyse them in order to determine syndromes, symptoms and diseases of the patient. Basically, any of these entities is associated with a FIP, which is, as we have seen, a very general description of intervention guidelines.

The automation in the union process of different FIPs is not a trivial task and implies, among the others, an accurate analysis of the objective feasibility. This is mainly due to the complexity and heterogeneity of the physical parameters and the HC domain variables. As regards this issue, I have defined some preliminary guidelines which I reported in the following sections. Basically, the first stage consists in the creation of a model.

It is necessary to verify both the preconditions to the integration of FIPs and the compatibility of the different drugs prescribed in each of the FIP of interest (whose administration may globally result in conflict). When the model is created, a formal verification of the model avoiding non-deterministic sequences of actions is required. Finally, manual revisions should be performed in order to customize the IIP for the current patient.

3.6.2 MAS Behavioural Orchestration in IIP Modelling

In practice, a FIP’s interface consists of an Entry Points (EP) set (Figure 21). EPs are essential because they represent an agent-based execution’s possible starting point in the provision of the FIP. In an SDA* model, a patient condition contains all the variables observed for the patient in a particular moment (i.e., signs, symptoms, antecedents, taken drugs, secondary diseases, test results, etc.). State variables are contained in the EPs of a FIP and their value is expressed by logical conditions (example: PRESSURE<140 and WEIGHT>75), which are at the basis of the verification of preconditions in the FIPs integration’s process.
I can summarize the procedural steps of the agent-based task integration process as follows:

- Analyse all sets of conditions contained in the EPs of all FIPs and list the state variables;
- Consider the possible intersections of EPs in all FIPs, that is, the set of state variables that are shared among EPs of different FIPs;
- If the intersection is empty, the FIPs can be considered as independent of each other, and the new IIP can be automatically defined as the union of different minor IIPs (each of them deriving from its corresponding FIP), to be personalised, merged and completed by medical staff, as reported at the end of this section;
- If the state variables intersection is not empty, then we can assert that some Model Interaction - MI - has occurred among the original FIPs (rounded corner rectangles in Figure 21). Hereby, I defined a MI as a
couple of EPs of two different FIPs that share at least one variable (1 or more than 1 variable).

![Diagram of IIP definition model in case of compatible conditions in all FIPs entry-points]

At this point, the verification of conditions associated to the state variables in common must be taken into account. For each MI, I have formalized the following additional steps:

- If all conditions relating to the variables shared between 2 different EPs in 2 different FIPs are compatible\(^{14}\), then both original EPs causing the MI can be replaced by a single new EP. This new EP will be non-deterministically connected (stars in) to both FIPs in which the original EPs from the MI were located.

The following example, together with the representations in Figure 21 and Figure 22, aims at clarifying the point in the previous statements.

(Example 1) Consider the following two EPs:

- **EP \(a\)**: \((\text{SUGAR}>40) \text{ and } (50<\text{PRESSURE}<100) \text{ and } (\text{IRON}<20)\)
- **EP \(b\)**: \((\text{SUGAR}>20) \text{ and } (70<\text{PRESSURE}<90) \text{ and } (\text{AGE}>65)\)

---

\(^{14}\) Note that the expression “compatible conditions” only refers to the fact that there is at least 1 (one) value that satisfies both conditions.
We can then define the new entry point as follows (circles coloured in light cyan in previous Figure 22):

$$\text{NewEP} = \text{EP}_n \cdot \text{a1} \cdot \text{b1}$$

\[\text{EP}_n \cdot \text{a1} \cdot \text{b1} \cdot \text{a1} \cdot \text{b1} : (\text{SUGAR} > 40) \text{ and } (70 < \text{PRESSURE} < 90) \text{ and } (\text{IRON} < 20) \text{ and } (\text{AGE} > 65)\]

In other words, for shared variables we take the most constraining condition, and we keep the other original conditions from the two EPs as they are. It is easy to verify that a patient entering the IIP through the new Entry Point (EP $n \cdot a1b1$ in Figure 22) could have also entered it through EP $a1$ or EP $b1$ (Figure 21). The opposite, on the contrary, would not hold, since a patient with sugar 50, pressure 60 and iron 15 could go through EP $a1$ of FIP “a”, but not through EP $n \cdot a1b1$ of the IIP. Entry points not involved in any MI will be naturally kept in the new IIP (rounds coloured in yellow, green and blue in Figure 21 and Figure 22). Figure 22, furthermore, shows two additional Model Interactions occurring among the original FIPs: one between FIP “b” and FIP “c” (leading to the EP $n \cdot b3c1$) and the other between FIP “a” and FIP “c” (leading to the EP $n \cdot a4c1$);

- If there is at least 1 variable leading to incompatible conditions, the involved EPs cannot be merged and in the resulting IIP both EPs will be kept as they were in the original FIPs (red-coloured circles in Figure 23). This case is illustrated by the following example.

(Example 2) Let us consider the following EPs:

- $\text{EP}_a1 : (\text{SUGAR} > 40) \text{ and } (50 < \text{PRESSURE} < 100) \text{ and } (\text{IRON} < 20)$
- $\text{EP}_b1 : (\text{SUGAR} < 15) \text{ and } (\text{PRESSURE} > 120) \text{ and } (\text{AGE} > 65)$

In this case, shared conditions on sugar and pressure are incompatible; therefore, as depicted in next Figure 23, the EPs will not be merged.
Once the FIP-integration’s verification phase terminates, motivations about the integration incompatibility are provided to practitioners as a possible decision-making support base.

3.6.3 Towards a Deterministic Evaluation of IIP Branches.

As reported in the previous paragraph, the introduction of a new entry point comes with a non-deterministically connected branch to both original FIPs (non-deterministic branches are represented as explosions in Figure 23). The reason for such a flow uncertainty lies on the initial impossibility to univocally identify the starting FIP to which the new EP should be connected.

Anyway, despite the objective decisional complexity, some kind of deterministic metrics (even though elementary) can be preliminarily introduced, in order to automate the continuation of an IIP provision. I will now proceed to explain my previous considerations through the example in the followings.

(Example 3) Let us briefly recall the EPs from the Example 1:

**EP a1**: (SUGAR>40) and (50<PRESSURE<100) and (IRON<20)

**EP b1**: (SUGAR>20) and (70<PRESSURE<90) and (AGE>65)
Let us consider the following possible Patient State Conditions (PSCs), before entering the EP n-a1b1. Let us assume that practitioners can consider a variable as not relevant when its measurement is not mandatory for the specific assessment:

- **PSC1**: (SUGAR=50) and (PRESSURE=75) and (IRON=10) and (AGE=Not-Relevant)
- **PSC2**: (SUGAR=60) and (PRESSURE=75) and (IRON=Not-Relevant) and (AGE=80)
- **PSC3**: (SUGAR=60) and (PRESSURE=80) and (IRON=Not-Relevant) and (AGE=Not-Relevant)
- **PSC4**: (SUGAR=60) and (PRESSURE=85) and (IRON=15) and (AGE=80)

The evaluation, case by case, of the branch automatically chosen when entering the EP n-a1b1 (decision-making processes in Figure 23) can be summarized as follows:

1) Case **PSC1**: all relevant variables (and conditions) involved belong to the EP a1 of FIP “a”, while the last variable, the age, though present and belonging to the EP b1 of FIP “b” is not relevant for the evaluation of the patient condition. From here, we can assume that the IIP execution can automatically continue towards the branch of FIP “a”.

2) Case **PSC2**: all relevant variables (and conditions) involved belong to the EP b1 of FIP “b”, while the last variable, the iron, though present and belonging to the EP a1 of FIP “a” is not relevant for the evaluation of the patient condition. From here, we can assume that the IIP execution can automatically continue towards the branch of FIP “b”.

3) Case **PSC3**: this is not a straightforward case, because all relevant variables describing the patient’s condition are exactly contained both in FIP “a” (EP a1) and FIP “b” (EP b1). In this case we can determine, for example, how close the measured state’s value for a
variable is to the boundaries of the condition containing the same variable in the different EPs. In other words, we can measure to which degree the variable’s value is contained in the condition range. The closer the value to the middle point of a range (presence of lower and upper boundary), the more the condition can take into account how far the variable value is from the boundary itself. The previous evaluation must be repeated for each variable and, finally, the majority of verified conditions will lead to a FIP instead to another.

Recalling briefly the actual case, we have:

- SUGAR=60, which better (more largely) satisfies the condition of EP b1 (SUGAR>20) than EP a1: (SUGAR>40). One point to FIP “b”;
- PRESSURE=80, which better satisfies the condition of EP b1: (70<PRESSURE<90) than EP a1: (50<PRESSURE<100), since EP b1’s middle point is 80 while EP a1’s is 75. Another point to FIP “b”.

In this specific case FIP “b” would automatically addressed as the candidate to the continuation of the IIP provision.

4) Case PSC 4: this case leads back to the non-deterministic approach previously mentioned. That is, further information should be taken into account in order to have a decisional opportunity. For example, important guidelines are implicitly provided by the nature of the FIPs: though they can have variables in common, their provision logics (care treatment) can address completely different diseases. From here, I argue that the representation of the care provision knowledge is always susceptible of relevant improvements.
The IIP resulting from the automated definition phase is neither a complete nor an executable IIP by the MAS: personalised drug quantifications, time constraints in the execution of actions, non-deterministic events management, and other aspects, have to be explicitly added to the model, as well. Once the final IIP is completed, it is saved in the Electronic Health Record of the patient. Thus, the medical team does not have to build the IIP from scratch, and can take into account the international recommendations in the treatment of the patient’s conditions, composing a fully personalised and accurate care plan for him. The IIP usually contains follow-up actions in which the state of the patient is checked. If the evolution of the patient through his customized plan follows an undesirable course, the EU can consider changing or even cancelling the IIP.
Chapter 4

Evaluation Results

The evaluation of the work presented in this manuscript is strictly correlated and interlaced with that of the whole K4Care platform and was worked out through an elaborate procedure, in three different steps, by all of the healthcare partners in close interaction with the technical staff.

I will present the acceptance results which directly refer to the MAS conceptualization, realization and functioning (home care services management and execution, mainly), skipping the references to other parts of the platform which concerned aspects, for example, of the visualization and appearance effectiveness which were out of my research’ scope and, therefore, out of this manuscript’s one, too.

The evaluation of the MAS-cored architecture has been executed in different places and was concluded in the city of Pollenza (Italy), where the platform was tested by healthcare professionals and caregivers on real patients, in order to verify the adherence of the application to:

1) Their needs and duties;
2) The possibility of use in every day activity;
3) The possibility of collecting and integrating information from different sources;
4) The possibility to use computer management tools for FIP personalization.
Staff of healthcare providers in real home care facilities has performed the final test. The choice of having the final test of a research project in a small town follows an established tradition of epidemiological studies. Since the seminal study on cardiovascular risk factors that started in Framingham, Massachusetts, in 1948 and enrolled its third cohort in 2002, many have followed this approach. A small town gives indeed the unique opportunity of studying an “entire”, homogeneous population.

The closing test has been performed in a real environment and involved the entire home care facility, GPs, the Municipality, Social Assistants, citizens’ representatives. The K4Care platform has been applied in the usual care activity delivered during the period of study. All the participants involved have provided a structured evaluation of the system, expressing their judgement through a validated questionnaire.

4.1 Guided Execution Test

A particular part of the testing period was represented by the event dedicated to guide participants in executing a scenario, to get familiar with the platform.

The execution of numerous scenarios (~100) gave the opportunity to both technical and medical partners to observe platform’s functionalities, to face realistic circumstances and to make necessary amendments.

The description and discussion on the different elements in K4Care MAS platform, together with the execution of guided scenarios, provided the opportunity to test, discuss, and refine the following functionalities:

- Agent-based access to the medical data in the EHCR;
- Activation of agent-managed HC services;

---

15 http://www.framinghamheartstudy.org/index.htm
✓ Reception and execution of agent-executed pending actions (with the corresponding editing of documents);
✓ Construction and execution of IIPs.

The virtual history of average HC patients, defining caring processes which necessitate the main elements of K4Care model and platform, was presented and participants were invited to simulate a role of one of the actors in the model and provide an answer to the requests of the case, in order to solve it within the realm of home care. Simulation – based on a realistic procedure, enclosing anonymized real data in key moments of the process – was performed by a real time connection with the K4Care server, accessed by laptops operated by participants.

4.1.1 Managing Real Patients

The evaluation process was completely transferred in the hands of the staff of Pollenza. The four GPs selected the group of patients to enrol and started the procedures of “usual care”, trying to use as much as possible the MAS-based K4Care platform to support their daily activities, to share information, and manage the series of events and procedures implied in the execution of home care services. During the work with real patients, other actors of the HC service were involved, in order to respect – as long as possible – the usual management of HC patients. That means that the whole group of actors involved kept track of each step of the care process (activating services, making requests, creating, filling, and modifying documents).

The activity was performed by a real time connection with the server of K4Care platform, accessed by laptops or desktops operated by participants and located as they virtually were at patients’ homes, GPs’ offices and in the District’s office. Professionals participating in the evaluation were: medical (4 GP, 1 Physician in Charge and Geriatrician as
Specialist Physicians, 4 Nurses), social (1 Social Worker), and institutional and political (2 City Councillors and the Territorial Service Director).

Patients to test the platform in Pollenza were selected from volunteers. Each participant signed an informed consent and each of the four GPs was asked to enrol at least five patients needing home care services.

All of the patients were home-care patients, in charge of the home care services, both health care and social. The mean age of this kind of patient is quite high, defining a target population of mainly geriatric users. This population is highly resource consuming, mostly of nursing services. Both of the syndromes considered by K4Care were included (cognitive impairment, immobility).

4.2 Closing Evaluation and Results

After actively using the platform, it was possible to provide informed judgements about the entire K4Care architecture (in terms of end users); the functionalities tested were those ones introduced at the beginning of Section 4.1. During the period of the K4Care platform testing, 23 patients were chosen by the GPs to be managed with the use of the platform itself. The assistance was performed thanks to the activation of 184 actions, trough 87 logins.

<table>
<thead>
<tr>
<th>Total number of logins</th>
<th>87</th>
</tr>
</thead>
<tbody>
<tr>
<td>New patient entered</td>
<td>23</td>
</tr>
<tr>
<td>Evaluation Units constituted</td>
<td>10</td>
</tr>
<tr>
<td>Pending actions (completed and not)</td>
<td>184</td>
</tr>
<tr>
<td>Documents created</td>
<td>206</td>
</tr>
<tr>
<td>Documents per patient</td>
<td>(\min 2 - \max 26)</td>
</tr>
</tbody>
</table>
Ten patients have required the constitution of an Evaluation Unit in order to make a comprehensive assessment of the case and define intervention plans. Documents created during the activity were of 43 different types, for a total amount of 206. The number of documents per patient ranged from 2 to 26. All the professionals participating in the evaluation in Pollenza have delivered such questionnaires. It has to be clarified that since some figures (City Councillors, GPs, and Geriatrician) coincided in the same individuals, it was decided to collect information only from the “professional” side.

The forms contained an adapted version of a standard Technology Acceptance Model (TAM) questionnaire (see Appendix section for further details) and questions specific for K4Care project, proposed in a TAM-like feature, to make it possible the use of a similar scoring scale (min 1 – max 7). At the end of the evaluation, nine questionnaires were completed: 4 from GPs, 1 from a Head Nurse, 2 from Nurses, 1 from a Geriatrician, and 1 from a Social Worker.

The mean overall scoring of the evaluation was 5.8/7 (SD 0.5). Regarding the score for the different sections, they were:

### Table 4: Some of the agent-managed documents created during the final evaluation

<table>
<thead>
<tr>
<th>Documents created</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>PatientAdministrativeInformationDocument</td>
<td>25</td>
</tr>
<tr>
<td>HCRequestDocument</td>
<td>24</td>
</tr>
<tr>
<td>AnamnesisDocument</td>
<td>23</td>
</tr>
<tr>
<td>PhysicalExaminationReportDocument</td>
<td>20</td>
</tr>
<tr>
<td>MiniMentalStateExaminationDocument</td>
<td>18</td>
</tr>
<tr>
<td>BarthelADLIndexDocument</td>
<td>15</td>
</tr>
<tr>
<td>InstrumentalActivitiesOfDailyLivingScaleDocument</td>
<td>13</td>
</tr>
<tr>
<td>PharmacologicalTreatmentDocument</td>
<td>12</td>
</tr>
<tr>
<td>NortonScaleDocument</td>
<td>10</td>
</tr>
<tr>
<td>ClinicalHistoryDocument</td>
<td>9</td>
</tr>
<tr>
<td>PrescriptionOfNursingCareDocument</td>
<td>9</td>
</tr>
</tbody>
</table>
Through 87 logins, 10 Evaluation Units were constituted, 184 actions activated and 206 documents created (with min 2, max 26 documents per patient). These numbers definitely represent something more than a big amount of work: they probably evidence the fact that the use of an ICT structure provides a tool that allows the execution of a bigger amount of work with respect to the average executed without such structure.

As a matter of fact, such numbers usually represent the work of months, more than weeks. The examiners, who provided a very good score of the platform, with a mean of 5.8/7, have probably promptly perceived this reality. Examining the scores of the related items (see table
3), it can be affirmed that the use has been judged more than easy, the usefulness was quoted very high, and that the professionals had a very positive attitude toward a real use of the platform.

**Table 6: Results of the platform’s evaluation (Continuation)**

<table>
<thead>
<tr>
<th>B) Perceived Usefulness</th>
<th>7 Using K4CARE in my job would enable me to accomplish tasks more quickly</th>
<th>6 6 7 7 5 5 4 6 7</th>
<th>5.9 1.1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>8 Using K4CARE would improve my job performance</td>
<td>6 6 7 6 6 5 5 6 7</td>
<td>6.0 0.7</td>
</tr>
<tr>
<td></td>
<td>9 Using K4CARE in my job would increase my productivity</td>
<td>6 6 7 6 6 5 5 6 7</td>
<td>6.0 0.7</td>
</tr>
<tr>
<td></td>
<td>10 Using K4CARE would enhance my effectiveness on the job</td>
<td>5 5 6 6 5 5 5 6 7</td>
<td>5.8 0.7</td>
</tr>
<tr>
<td></td>
<td>11 Using K4CARE would make it easier to do my job</td>
<td>6 6 7 6 6 5 5 6 7</td>
<td>6.0 0.7</td>
</tr>
<tr>
<td></td>
<td>12 I would find K4CARE useful in my job</td>
<td>0 0 0 0 5 5 5 5 7</td>
<td>5.9 0.6</td>
</tr>
<tr>
<td></td>
<td>mean</td>
<td>5.8 5.0 6.7 6.2 5.5 5.0 4.8 6.0 7.0</td>
<td>5.9</td>
</tr>
<tr>
<td></td>
<td>SD</td>
<td>0.4 0.0 0.5 0.4 0.4 0.0 0.4 0.0 0.0</td>
<td>0.7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C) Attitude Toward Using</th>
<th>1 Good/Bad</th>
<th>7 6 6 7 5 5 5 6 7</th>
<th>6.0 0.9</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2 Beneficial/Harmful</td>
<td>6 6 6 - 5 5 5 6 7</td>
<td>5.9 0.6</td>
</tr>
<tr>
<td></td>
<td>3 Wise/Foolish</td>
<td>7 6 6 - 5 5 5 6 7</td>
<td>5.9 0.8</td>
</tr>
<tr>
<td></td>
<td>4 Positive/Negative</td>
<td>7 6 6 - 5 5 5 6 6</td>
<td>6.0 1.0</td>
</tr>
<tr>
<td></td>
<td>mean</td>
<td>6.8 6.0 6.0 7.0 5.0 5.0 5.5 6.0 7.0</td>
<td>5.0</td>
</tr>
<tr>
<td></td>
<td>SD</td>
<td>0.8 0.0 0.0 - 0.0 0.6 0.6 0.0 0.0</td>
<td>0.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>D) How would you rate the different sections of the K4CARE platform?</th>
<th>1 Managing documents in the EHR</th>
<th>5 7 6 6 5 5 5 6 5</th>
<th>5.6 0.7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2 Access to the data in the EHR</td>
<td>5 7 5 6 5 5 5 6 7</td>
<td>5.7 0.9</td>
</tr>
<tr>
<td></td>
<td>3 Activation of services</td>
<td>0 6 5 6 5 5 5 6 7</td>
<td>5.7 0.7</td>
</tr>
<tr>
<td></td>
<td>4 Reception and execution of pending actions</td>
<td>7 6 5 6 3 4 4 6 7</td>
<td>5.3 1.4</td>
</tr>
<tr>
<td></td>
<td>mean</td>
<td>5.8 5.5 5.5 5.5 4.5 4.8 4.8 6.0 6.5</td>
<td>5.8</td>
</tr>
<tr>
<td></td>
<td>SD</td>
<td>1.0 0.6 0.5 0.0 1.8 0.5 0.2 0.9 1.0</td>
<td>0.8</td>
</tr>
<tr>
<td>Overall mean (A+B+C+D)</td>
<td>6.9 6.2 5.7 6.2 5.7 5.1 5.2 5.9 6.6</td>
<td>5.8 mean</td>
<td></td>
</tr>
<tr>
<td>SD</td>
<td>0.7 0.4 0.9 0.0 1.8 0.4 0.6 0.3 0.8</td>
<td>0.5</td>
<td></td>
</tr>
</tbody>
</table>

Most appreciated were the **agent-managed access** to data and the **activation of agent-based services**. While the users affirm that there is a very high need in health care of tools like K4Care, at the same time they testify the scarceness of such tools in use.

It has to be emphasized that Pollenza health service is usually carried out at a low technological profile and that even at an individual professional
level, the staff is neither used nor familiar with the use of ICT tools during professional activities. That means that K4Care platform succeeded in breaking the wall of negative bias towards the extensive use of ICT usually verifiable in small services with lower use of technologies.

The user-friendliness of the platform was apparently enough to make it possible to appreciate the added value of K4Care approach. Summarizing, the evaluation of K4Care platform can be judged positive both with regards to the concrete realization of the tool and to the possibility and willingness of usage in real health services.
Chapter 5

Conclusions

Agent technology, as well as any other technology, comes with its own specific characteristics, thanks to which, in specific application contexts, it can be applied as the most suitable and adoptable solution. Accordingly, the agent-paradigm can provide real competitive advantages, when designers and users exploit this technology’s full potentialities, by delegating to the system decisional and behavioural competences.

The research work I have presented in this manuscript aimed, first of all, at investigating the real potentialities of the agent-based technology (like intelligence and autonomy) and, then, at providing IT designers with practical and useful guidelines towards the industrialization of agent-based systems. I have customized aspects typical of OO software engineering to Multi Agent Systems.

Nevertheless, I have provided the reader with a personal characterization of the agent paradigm, measured through a so-called Agentometric Net. According to this vision, an agent is not the specialization of a limited set of particular skills but a complete “intelligent” and adaptive entity, which can decide, time by time, which is the most proper ability to apply in the achievement of the specific objectives.

Nevertheless, I have presented the formulation of my four theses, which synthesized the results obtained during my entire period of research.
My discussion homogeneously focused both on theoretical and practical relevance of agents’ exploitation in the modelling of complex, highly populated domains. These domains are usually characterized by the presence of long-term interactions among the actors of the system.

In particular, my first thesis (T1.1 on page 32) deals with the conceptualization of agent-paradigm and relates to the opportunity given by the latter to enable a more natural description of application knowledge. It permits to govern and embody complex knowledge representations by atomizing behavioural characteristics of domain actors. This first formulation aims at highlighting how agent-paradigm can represent a natural, strategic core technology for those systems in which knowledge plays a fundamental characterizing role.

The second thesis I have presented (T1.2 on page 32) intends to illustrate how agent-based software can gracefully adapt to long-term active negotiations and continuous environmental and strategic changes. These are all crucial elements in a project for the achievement of the initial objectives, considering today’s business complexity and orchestration, as individuals are often led to long-term negotiations and changes, which cause a continuous generation of newer knowledge and strategies.

On the other hand, the adoption of an engineering approach in the realization of agent-based applications is to be considered as mandatory, in order to guarantee the extensibility, adaptability and testability of a system, as well as to decrease the objective development complexity, which agent-based systems inevitably involve.

With regard to this aspect, my third thesis (T2.1 on page 32) asserts how the automation of agent-oriented code generation can provide with much higher semantics and promptness the MAS’ prototyping process. Generally speaking, costs and efforts devoted to the realization of software architectures can be significantly palliated by the adoption of semantics-
based prototyping techniques. This is particularly true also in presence of Multi Agent Systems.

Accordingly, the fourth thesis I presented (T2.2 on page 32) aims at consolidating a fundamental aspect in agent technology, that is agent-paradigm enables formal extensions of actor behavioural models, which, in turn, permit to recognize, manage and embody new emerging Application Knowledge. The process of KM, in a given domain, does not end with the release of the software designed for the specific purpose. The utilization of the system itself, by the users, will inevitably cause the generation of new manifested knowledge, previously unrecognized, and so, unrevealed. In other words, agent behavioural models can be orchestrated and extended through their elementary founding components, in order to permit the system to gracefully evolve towards newer application scenarios.

The principles formulated in my theses have been actually applied to a real MAS-based platform, which was developed in compliance with the specifications of K4Care European project. The objective was, among the others, to provide the participant countries with an intelligent agent-based platform capable of assuring the provision of long-term home care services to elderly patients, with co-morbid conditions and diseases, cognitive and/or physical impairment.

Results inherent in testing phase of K4Care MAS platform have been separately reported. They have evidenced both the innovativeness of the approach in modelling home care services through agent-based systems and the high level of acceptance by practitioners (domain experts) involved in the project realization, who provided a positive evaluation with regards both to the concrete realization of the MAS-based platform and the possibility and willingness of usage in real health services.

This is the essence of my investigations which I have conducted and illustrated in the present manuscript.
Appendix

**Technology Acceptance Model (TAM)**

Technology Acceptance Model (TAM) is an information systems theory that models how users come to accept and use a technology.

The goal of TAM is to provide an explanation of the determinants of computer acceptance that is general, capable of explaining user behaviour across a broad range of end-user computing technologies and user populations, while at the same time being both parsimonious and theoretically justified [67].

The model suggests that when users are presented with a new technology, a number of factors influence their decision about how and when they will use it, notably:

- Perceived usefulness (PU) - This was defined by Fred Davis as “the degree to which a person believes that using a particular system would enhance his or her job performance”
- Perceived ease-of-use (PEOU) - Davis defined this as “the degree to which a person believes that using a particular system would be free from effort”.

TAM is one of the most influential extensions of Ajzen and Fishbein’s theory of reasoned action (TRA) in the literature [68]. Fred Davis and Richard Bagozzi developed it. TAM replaces many of TRA’s attitude measures with the two technology acceptance measures— ease of use, and usefulness. TRA and TAM, both of which have strong behavioural elements,
assume that when someone forms an intention to act, this someone will be free to act without limitation. In the real world there can be many constraints, such as limit the freedom to act [66].

Bagozzi, Davis and Warshaw say: “Because new technologies such as personal computers are complex and an element of uncertainty exists in the minds of decision makers with respect to the successful adoption of them, people form attitudes and intentions toward trying to learn to use the new technology prior to initiating efforts directed at using. Attitudes towards usage and intentions to use may be ill-formed or lacking in conviction or else may occur only after preliminary strivings to learn to use the technology evolve. Thus, actual usage may not be a direct or immediate consequence of such attitudes and intentions”. See [66] and representation in Figure 24.

**Figure 24: Diagram of the attitude of people when confronted to new technologies**

Several researchers have replicated Davis’s original study to provide empirical evidence on the relationships that exist between usefulness, ease of use and system use. The extensive research activity produced 147 articles between 1990 and 2003, demonstrating strong reliability and validity of instruments. Many different versions of the original one have been proposed and used, specifically addressing different areas of interest and showing TAM as a good example of how a model is extended and applied.
Abbreviations

[AK] = Application Knowledge (first citation page: 39)
[DDK] = Domain Description Knowledge (39)
[DK] = Declarative Knowledge (39)
[FD] = Family Doctor (45)
[FIPA] = Foundation for Intelligent Physical Agents (27)
[HC] = Home Care (29)
[HCP] = Home Care Patient (29)
[HN] = Head Nurse (45)
[ICG] = Informal Care Giver (45)
[IT] = Information Technology (9)
[JADE] = Java Agent Development framework (28)
[K4Care] = Knowledge for Care (29)
[KM] = Knowledge Management (32)
[MAS] = Multi Agent System (13)
[Nu] = Nurse (45)
[OK] = Objective Knowledge (40)
[OWL] = Ontology Web Language (75)
[PC] = Physician in Charge (45)
[PK] = Procedural Knowledge (40)
[SCG] = Specialised Care Giver (45)
[SK] = Service Knowledge (40)
[SP] = Specialist Physician (45)
[SW] = Social Worker (45)
[TK] = Technological Knowledge (40)
My Research Related Publications


Bibliography


Bibliography


