UNIVERSITAT POLITÈCNICA DE CATALUNYA

DEPARTAMENT DE LLENGUATGES I SISTEMES INFORMÀTICS
PROGRAMA DE DOCTORAT EN INTEL·LIGÈNCIA ARTIFICIAL

PHD THESIS

Agent-Based Management of Clinical Guidelines

Memòria presentada per David Isern Alarcón
per optar al títol de Doctor en Informàtica per la
Universitat Politècnica de Catalunya

Director
Dr. Antonio Moreno (Universitat Rovira i Virgili)

Tutor
Dr. Ulises Cortés (Universitat Politècnica de Catalunya)

Barcelona, Desembre de 2008

A la meva estimada Sònia

”The desire of knowledge, like the thirst for riches,
increases ever with the acquisition of it.”

Laurence Sterne (1713-68) British writer

Agraïments

Expresso el meu agraïment al Dr. Antonio Moreno, director de la Tesi, sense l'ajut del qual no hauria estat possible arribar fins aquí. També vull agraïr els comentaris del Dr. Ulises Cortés, el meu tutor a l'UPC.

Aquest treball s'ha finançat mitjançant els projectes europeus h-TechSight (IST-2001-33173) i K4Care (IST-2004-026968), el projecte del Ministerio de Ciencia e Innovación HYGIA (TIN2006-15453-C04-01), així com la Beca pre doctoral de la Universitat Rovira i Virgili, als quals agraeixo la possibilitat d'haver portat a terme aquesta tesi.

La recerca no és una tasca que fa una persona sola, sinó que durant tot aquest temps he tingut la sort de conèixer i treballar al costat de molta gent a la qual vull agrair tot el suport i la dedicació que m'han donat. Voldria destacar l'ajut de la Dra. Aïda Valls en tota la part de personalització, i sovint, per intervenir en discussions sobre la Tesi. També voldria donar les gràcies a la gent del Advanced Computer Laboratory de Londres que em van ajudar a donar una empenta definitiva a tot el manegament de les guies de pràctica clínica. Tornant a Tarragona, vull donar les gràcies especialment al Dr. David Riaño que sempre m'ha mostrat el seu suport durant tot aquest temps.

Ja ho saben, però sempre tinc present la gent del laboratori, tant els que hi ha actualment com els que ja no hi treballen: la Montse, l'Àlex, l'Albert, el Miquel, en Lucas, la Cristina, l'Angel, el Jaime i el David. Aquest darrer mereix una menció especial. Al Dr. Sánchez, un amic, amb el qual ja fa força temps que ens coneixem, li vull agrair tota l'ajuda que m'ha donat en tot aquest temps, i sense la qual mai hagués pogut acabar aquesta Tesi. A tots us vull desitjar la millor de les sorts.

La família, papa, mama, Anna ... i ara, sogres, cunyaos, i cunyada, que sempre heu cregut en mi i que m'heu animat sempre. També un record pels que ja no podran veure com arribo al final d'aquesta etapa, però que sempre són presents. Gràcies a tots.

Finalment, vull agrair molt especialment a la Sònia (i també Dra. Abelló) tot el que ha fet per mi. M'ha ajudat en els mals moments, m'ha escoltat, i el millor de tot, sempre està allí per donar-me un cop de mà i ajudar-me en qualsevol cosa que li demani. Per la seva paciència i dedicació, li vull agrair poder arribar fins aquí.

Contents

List of Figures

List of Tables

Abstract

Clinical guidelines (CGs) contain a set of directions or principles to assist the health care practitioner with patient care decisions about appropriate diagnostic, therapeutic, or other clinical procedures for specific clinical circumstances. It is widely accepted that the adoption of guideline-execution engines in daily practice would improve the patient care, by standardising the care procedures. Guideline-based systems can constitute part of a knowledge-based decision support system in order to deliver the *right knowledge to the right people in the right form at the right time*. The automation of the guideline execution process is a basic step towards its widespread use in medical centres.

To achieve this general goal, different topics should be tackled, such as the acquisition of clinical guidelines, its formal verification, and finally its execution. This dissertation focuses on the execution of CGs and proposes the implementation of an agent-based platform in which the actors involved in health care coordinate their activities to perform the complex task of guideline enactment.

The management of medical and organizational knowledge, and the formal representation of the CGs, are two knowledge-related topics addressed in this dissertation and tackled through the design of several application ontologies. The separation of the knowledge from its use is fully intentioned, and allows the CG execution engine to be easily customisable to different medical centres with varying personnel and resources.

In parallel with the execution of CGs, the system handles citizen's preferences and uses them to implement patient-centred services. With respect this issue, the following tasks have been developed: a) definition of the user's criteria, b) use of the patient's profile to rank the alternatives presented to him, c) implementation of an unsupervised learning method to adapt dynamically and automatically the user's profile.

Finally, several ideas of this dissertation are being directly applied in two ongoing funded research projects, including the agent-based execution of CGs and the ontological management of medical and organizational knowledge.

Resum

Les *guies de pràctica clínica* (GPC) contenen un conjunt d'accions i dades que ajuden a un metge a prendre decisions sobre el diagnòstic, tractament o qualsevol altre procediment a un pacient i sobre una determinada malaltia. És conegut que l'adopció d'aquestes guies en la vida diària pot millorar l'assistència mèdica als pacients, pel fet que s'estandarditzen les pràctiques. Sistemes computeritzats que utilitzen GPC poden constituir part de sistemes d'ajut a la presa de decisions més complexos amb la finalitat de proporcionar *el coneixement adequat a la persona adequada, en un format correcte i en el moment precís*. L'automatització de l'execució de les GPC és el primer pas per la seva implantació en els centres mèdics.

Per aconseguir aquesta implantació final, hi ha diferents passos que cal solucionar com per exemple, l'adquisició i representació de les GPC, la seva verificació formal, i finalment la seva execució.

Aquesta Tesi està dirigida en l'execució de GPC i proposa la implementació d'un sistema multi-agent. En aquest sistema els diferents actors dels centres mèdics coordinen les seves activitats seguint un pla global determinat per una GPC. Un dels principals problemes de qualsevol sistema que treballa en l'àmbit mèdic és el tractament del coneixement. En aquest cas s'han hagut de tractar termes mèdics i organitzatius, que s'ha resolt amb la implementació de diferents ontologies. La separació de la representació del coneixement del seu ús és intencionada i permet que el sistema d'execució de GPC sigui fàcilment adaptable a les circumstàncies concretes dels centres, on varien el personal i els recursos disponibles.

En paral·lel a l'execució de GPC, el sistema proposat manega preferències del pacient per tal d'implementar serveis adaptats al pacient. En aquesta àrea concretament, a) s'han definit un conjunt de criteris, b) aquesta informació forma part del perfil de l'usuari i serveix per ordenar les propostes que el sistema li proposa, i c) un algoritme no supervisat d'aprenentatge permet adaptar les preferències del pacient segons triï.

Finalment, algunes idees d'aquesta Tesi actualment s'estan aplicant en dos projectes de recerca. Per una banda, l'execució distribuïda de GPC, i per altra banda, la representació del coneixement mèdic i organitzatiu utilitzant ontologies.

Chapter 1

Introduction

1.1 Clinical guidelines

A *clinical guideline* (in the following, CG) is a highly matured therapeutic plan that compiles optimum practices for treating patients in a well-defined medical syntax. Thus, the adoption CGs are a promising way for standardising and improving health care practices [Field & Lohr 1990, Mersmann & Dojat 2004].

The National Cancer Institute (NCI), which is a US governmental organization, maintains a thesaurus, a terminological server and a metathesaurus focusing the study in cancer-related terms and relations[1]. It introduced a brief definition of CG:

> *"Guidelines developed to help health care professionals and patients make decisions about screening, prevention, or treatment of a specific health condition."*

The CancerWEB Project[2] defines a CG as follows:

> *"Clinical guidelines are a set of directions or principles to assist the health care practitioner with patient care decisions about appropriate diagnostic, therapeutic, or other clinical procedures for specific clinical circumstances. Practice guidelines may be developed by government agencies at any level, institutions, organizations such as professional societies or governing boards, or by the convening of expert panels. They can provide a foundation for assessing and evaluating the quality and effectiveness of health care in terms of measuring improved health, reduction of variation in services or procedures performed, and reduction of variation in outcomes of health care delivered."*.

Numerous CGs have been produced[3] and disseminated by a variety of government and

[1] For more information, please visit http://www.nci.nih.gov/cancerinfo/terminologyresources [last visit 01/12/2008].

[2] The project CancerWEB is being supported by the Dept. of Medical Oncology, University of Newcastle upon Tyne. Website http://cancerweb.ncl.ac.uk/ [last visit 01/12/2008]

[3] These CGs cover some medical areas such as psychiatry (*e.g.*, altered mental states, depression, dementia), nutrition (*e.g.*, altered nutritional states, dehydration), oncology (*e.g.*, breast cancer, cutaneous melanoma, epithelial ovarian cancer), and general medicine (*e.g.*, heart failure, osteoporosis, urinary incontinence).

professional organizations. The dissemination is made through multiple formats, including books, journals, technical reports, and the Web. Some of the biggest websites are:

- *National Guideline Clearinghouse* (NGC)[4] maintains a database of evidence-based clinical guidelines and related documents. NGC is an initiative of the Agency for Healthcare Research and Quality (AHRQ), U.S. Department of Health and Human Services. NGC has almost 1,000 publicly accessible guidelines.

- *The Guidelines International Network* (G-I-N)[5] is an international non-profit association of organizations and individuals involved in development and use of clinical practice guidelines. Founded in November 2002, G-I-N has grown to 76 member organizations from 36 countries.

- *OpenClinical*[6] is a non-profit organization created and maintained as a public service with support from Cancer Research UK under the overall supervision of an international technical advisory board. The main goal of this website is tracking developments on advanced knowledge management technologies for healthcare.

- *Scottish Intercollegiate Guidelines Network* (SIGN)[7] was formed in 1993. Its main objective is to improve the quality of health care for patients in Scotland by reducing variation in practice and outcome, through the development and dissemination of national clinical guidelines. The membership includes all the medical specialities, nursing, pharmacy, dentistry, professions allied to medicine, patients, health service managers, social services, and researchers. Currently, SIGN has 99 evidence-based clinical guidelines - published, in development, or under review - covering a wide range of topics.

[4]NGC website: http://www.guideline.gov [last visit 01/12/2008].
[5]G-I-N website: http://www.g-i-n.net [last visit 01/12/2008].
[6]OpenClinical website: http://www.openclinical.org [last visit 01/12/2008].
[7]SIGN website: http://www.sign.ac.uk [last visit 01/12/2008].

1.2 Management of clinical guidelines

Several steps must be considered in the management of clinical guidelines: *representation*, *acquisition*, *verification* and *execution*. The first three tasks concern the authors of the guideline, whereas the later is related to practitioners. Briefly, these steps can be described as follows:

a) *Choice of a representation language*. A CG contains several elements to be modelled, such as actions, required patient data, decisions to be taken, constraints between tasks, temporal constraints in a global plan, etc. Different researchers have defined formal languages to model computer-interpretable clinical guidelines, such as PRO*forma*, EON, GLIF, GUIDE or Asbru [Clercq et al. 2004, Fox & Das 2000, Peleg et al. 2003, Wang et al. 2001].

b) *Acquisition of CGs*. Medical guidelines are based on the evidence collected from clinical trials and existing literature [Davis et al. 2007, Priori et al. 2003]. Some authors are also currently working in the semi-automatic construction of guidelines, by applying Machine Learning techniques from a corpus of clinical data collected in a medical centre ([Riaño 2004]) or directly from textual documents ([Hrabak et al. 2007]).

c) *Verification of CGs*. Verification includes two aspects: is a medical guideline well formed?, and, which of these two available medical guidelines is the best? The first question seeks to verify the formal correctness of the guideline [ten Teije et al. 2006, Hommersom et al. 2007]. The second question is more difficult to answer since it is necessary to quantify how appropriate is a medical guideline. To tackle this problem, some authors proposed an evaluation procedure called AGREE which calculates a set of parameters for a given medical guideline to evaluate its *quality* [Agree 2003]. In addition a methodology to facilitate the whole development and evaluation of clinical guidelines can be found in [Ricci et al. 2006].

d) *Execution aspects*. As mentioned above, a medical guideline contains a great amount of information to be considered (decisions to be taken, constraints between tasks, temporal restrictions). All these data have to be collected and monitored when enacting the guideline.

Concretely, while representation, acquisition and verification stages are currently active research areas ([Chesani et al. 2006, Leong et al. 2007, Peleg et al. 2008, Seyfang et al. 2006]), the execution of guidelines is a less developed field, and it is the main focus of this thesis.

1.3 Execution of clinical guidelines: benefits

Nowadays, the execution of *clinical guidelines* is one of the most interesting topics of study within *clinical informatics*.

A guideline execution engine should ideally fulfil the following requirements:

- To keep a repository of guidelines.
- To facilitate the creation of guidelines through a graphical editor, or even define a methodology to create or reuse guidelines.
- To provide a formal language for encoding medical guidelines.
- To provide mechanisms to coordinate the services required in the use/management of guidelines.
- To allow the user to analyse the behaviour of the guideline (*e.g.*, by providing a run-time engine or a simulator).
- To provide a connection with a computerised patient record[8].
- To allow the use of standard vocabularies inside guidelines.
- To provide security to both transmissions and storage of sensitive data related to patients.

Numerous studies show the benefits provided to both patients and practitioners of the inclusion of CGs in the daily practice ([Barahona et al. 2001, Elkin et al. 2001, Hart 2003, Lenz et al. 2007, Lenz & Reichert 2007, Rutten et al. 2005, ten Teije et al. 2006, Woolf et al. 1999]). Some of the most relevant are the following:

- For healthcare professionals, the use of CGs can improve the quality of clinical decisions and activities and, in consequence, the patient outcomes are also improved (*e.g.*, a clinician will not forget an important aspect to be checked before ordering a certain treatment).
- CGs facilitate reuse of knowledge, because a guideline can be adapted, tailored and applied to different clinical situations.
- Guidelines support rapid dissemination of updates and changes. CGs promote interventions of proved benefits and discourage those that are ineffective.
- CGs help doctors to use the clinical knowledge about the patient at the appropriate point of his care.
- Guideline authors are encouraged to employ rigorous formal techniques, which help to ensure syntactic, logical and medical validity of CGs.

[8] At the simplest, a computerised patient record is the computer replacement for existing paper medical record systems. It provides mechanisms for capturing information during the medical visit, stores it in a secure fashion, and permits retrieval of that information by those with a clinical need [Coiera 2003*a*].

Although the number of available clinical guidelines grows continuously, they are not being widely used in daily practice. Several factors limiting or restricting complete physicians adherence to clinical guidelines were identified in [Cabana et al. 1999, Bond 2007]. Such factors, named *barriers*, were organised into groups based on whether they affected physician knowledge, attitude or behaviour.

- *Knowledge* barriers such as lack of awareness with the guideline's existence, or simply a lack of familiarity with the guideline content.
- *Attitude* barriers such as lack of agreement with specific guidelines or with guidelines in general, lack of physician self-efficacy, lack of outcome expectancy or a lack of motivation for the inertia of habits or routines.
- *Behaviour* barriers, also called external barriers, which were divided into: guideline-related factors (*e.g.*, difficulty to apply CGs in daily practice, a CG changes established behaviours), patient-related factors (*e.g.*, inability to consider patient preferences with guideline recommendations), and environmental factors (*e.g.*, insufficient staff or resources).

These factors could be tackled (in most cases) with an automation and computerisation of the daily management of both clinical guidelines and patient data [Blaser et al. 2007]. To deliver patient-specific advice at the time and place of a consultation is an important contribution to improve clinician performance, and the introduction of IT applications is one of the preconditions to be accomplished [Maviglia et al. 2003, Raghupathi & Tan 2002, Zielstorff 1998]. Furthermore, the IT inclusion into clinical practice is a critical task, as current processes will necessarily change and adapt to new circumstances.

1.4 Goals and contributions

The main goals of the present work are:

- To present an open and distributed architecture of healthcare organizations that delivers patient-oriented medical services. The architecture changes the point of view of traditional healthcare providers with monolithic and closed packages. The designed platform may include external elements or be included into an existing workflow by using the appropriate interfaces.
- Several studies have shown the benefits of the inclusion of clinical guidelines into daily care. Usually, the same studies show that they are not widely used due to some barriers (as explained previously). One of the goals of this dissertation is to design a system that helps to tackle those barriers taking into account these basic subgoals:
  - it is easy-to-deploy,
  - it allows a customisation to diverse medical centres,
  - it includes information about the patient's preferences,
  - it automates as much as possible the management of medical data by taking into account the information stored in clinical guidelines,
  - it implements a system independant of the guidelines representation language.
- The internal entities (actors) should coordinate their daily activities in order to accomplish a complex task as the execution of a clinical guideline.
- Medical informatics deals with a particular kind of knowledge that requires an effective and flexible representation. In addition, clinical guidelines include information about the execution that is done within a healthcare organisation. These two points-of-view should be combined appropriately in order to create a flexible and robust knowledge-driven system.
- Due to the sensitive nature of the data used in this domain, any system should guarantee their accurate management. Security measures during storage and transmission of patient's related data should be added in order to ensure authentication and privacy.

Through this thesis, we make the following contributions towards the adoption of clinical guidelines in daily care:

- **The adequacy of intelligent agents to healthcare problems has been analysed**. The main characteristics of multi-agent systems and medical informatics problems have been studied. Then, the suitability of applying the agent paradigm to solve common problems in the healthcare domain has also been investigated. Finally, a classification and review of agent-related works in the healthcare domain has been provided.

- **To build a framework to analyse and compare existing agent-oriented software engineering methodologies**. This framework combines topics from both agent technology and standard software engineering. It is based on previous works in the area but it includes new issues about agents' organizations required in our research work. From the evaluation of several methodologies, INGENIAS was selected [Pavón et al. 2005].

- **The design and implementation of a distributed careflow framework** that can be adapted to the characteristics of different healthcare organizations. From the software-engineering point-of-view, the system has followed an agent-oriented methodology (INGENIAS, [Pavón et al. 2005]) during the analysis and design of the platform, allowing an improvement of the quality of the final product.

- **Actors involved in the enactment of clinical guidelines coordinate their activities.** The internal elements of the platform act autonomously and proactively, and that improves the flexibility and robustness of the system, and allows to coordinate the execution of tasks in an effective way.

- **The implementation of a new method to offer personalised medical services.** The designed method implements a personalised delivery of medical services. It maintains a user's profile that is employed to rate and rank alternatives composed by the platform, and a novel unsupervised learning method that allows the system to maintain dynamically this user's profile.

- **The implementation of different application ontologies that deal with all medical and organizational knowledge managed among all entities**. It allows the separation of the knowledge representation from its use. This approach allows to describe declarative and procedural knowledge accurately. In addition it allows to adapt the system to different (execution) circumstances without changing the internal behaviour of agents.

- **Implementation of an ontological representation of clinical guidelines** that has been designed as a generalisation of different representation languages (PRO*forma*, SAGE and SDA*). This approach allows to manage clinical guidelines coded using those languages, and the ontology provides a transparency between the representation and its use.

- **Several ideas of this dissertation are being applied in research projects that provide a practical validation**. The *K4Care* and *Hygia* research projects have adopted the agent-based approach for the enactment of clinical guidelines, and the implementation of several ontologies to embed medical and organizational knowledge managed in both systems.

1.5 Thesis organization

This thesis is organized as follows:

- **Chapter 2** presents an state-of-the-art of guideline-based clinical decision support systems. It provides an analysis and a comparison of the existing guideline execution systems in order to know the requirements of this kind of systems, their common features, and their main advantages and shortcomings. One of the main goals of this dissertation is to go beyond these systems in order to build a guideline-based execution system including those common features and improving their general performance using a distributed approach based on intelligent agents. A preliminary version of this chapter can be found at [Isern & Moreno 2006], and an extended and up-to-date release has been recently published in [Isern & Moreno 2008*b*].

- **Chapter 3** analyses the use of the agent paradigm in the healthcare domain. First of all, the main characteristics of multi-agent systems and medical informatics problems are studied. Then, the adequacy of applying intelligent agents in the healthcare domain is evaluated. Finally, a classification of topics where researchers have included agents is also provided, with a a brief review of some systems in each category.

- **Chapter 4** presents a methodological design of our agent-based platform using an agent-oriented software engineering methodology [Isern, Gómez-Alonso & Moreno 2008]. From a previous study of all available approaches, shown in Appendix B, the most appropriate to our requirements was selected and used following these steps: to analyse the problem, to design the entities and relationships between them, and finally, in this particular case, to generate pieces of source code (basic skeleton of agents).

- **Chapter 5** changes the point of view of traditional healthcare applications enabling the citizen (or patient) as an active partner when requesting medical services. The chapter is divided in two main parts. The first introduces the HECASE system and the basic information services delivered to the user through the agent-based platform [Isern et al. 2003*a*, Moreno, Isern & Sánchez 2003, Moreno, Valls, Isern & Sánchez 2003]. The second introduces an algorithm to personalise the information presented to the user. This personalisation is performed during the booking of a medical visit. The user's personal agent maintains a profile of the user's interests, which are used to guide the search for free slots of the doctors. The user's profile evolves according to his use of the system. The learning algorithm has two stages: a rating step to sort and filter the proposed alternatives, and a post-processing learning step to adapt the user's profile [Bajo, Corchado, Fernández, Fuentetaja, González, Isern, López & Valls 2007, Isern et al. 2006*a*, Isern et al. 2006*b*, Moreno et al. 2006].

- **Chapter 6** introduces, from the ontological engineering point of view, the main components (classes, relationships and instances) and uses of the two ontologies developed in this work. First of all, after studying and analysing the main existing languages to represent clinical guidelines, a high level ontology to cover the features of some languages has been implemented. It permits to design a guideline execution system without a dependence on a particular codification, but at the same time, allowing the use of guidelines coded in those languages. After that, an application ontology designed to represent medical and organizational knowledge is introduced [Isern, Sánchez & Moreno 2007*b*, Isern, Sánchez & Moreno 2007]. Finally, a case study that shows a complete view of the ontology-based execution of CGs is presented [Isern, Sánchez & Moreno 2007*a*].

- **Chapter 7** presents two research projects that employ heavily the ideas of this thesis with the inclusion of an agent-based platform to enact clinical guidelines. The projects, called *K4Care* and *Hygia*, are ongoing at the moment and an overview of them is made. *K4Care*'s-related papers are [Isern, Moreno, Pedone & Varga 2008, Hajnal et al. 2007, Isern, Millan, Moreno, Pedone & Varga 2008*a*, Isern, Millan, Moreno, Pedone & Varga 2008*c*].

- **Appendix A** contains an state-of-the-art of available agent-oriented software engineering methodologies. First of all, a classification of all approaches is done. Then, a brief summary of the main features offered in each case is given. Finally, a novel multi-dimensional framework is defined in order to compare all the approaches from different points of view including agent-based features, communication skills, and agent designer support tools. A first version of this chapter was given in [Gómez-Alonso et al. 2007].

- **Appendix B** explains the meaning of several symbols used during the methodological analysis done in Chapter 4.

Chapter 2

Computer-based Execution of CGs: State-of-the-art

As commented in the introduction, the main goal of this dissertation is the implementation of a guideline-based execution engine improving the general performance of current approaches using a multi-agent system, which coordinates both the collection of data and its transmission to the correct point of care. The first task to perform is to analyse all the existing guideline execution systems. After sketching the main features of each of them, a comparison is made, and some concluding remarks are also done.

Other systems related to the execution of guidelines found in the literature such as order entry systems and alarm-based systems are out of the scope of this dissertation. Computerised Physician Order Entry (CPOE) systems refer to a variety of computer-based systems for ordering medications, which share the common feature of automating the medication ordering process. A basic CPOE system ensures standardized, legible, complete orders by only accepting typed orders in a standard and complete format [Georgiou et al. 2007]. With the same goal, there are decision support systems that can provide advice on drug selection, dosages, and duration [Coiera 2003*a*]. There are computerised systems that monitor certain events and trigger alerts about at-risk states and reminders of appropriate physical assessments and screening activities. Sometimes active systems provide an explanation that offers background information, definitions and risks [Coiera 2003*a*, Hripcsak et al. 1996, Shiffman et al. 1999]. These systems consider neither complex sequences of actions nor coordination aspects, but just react to specific situations.

Before undertaking a deeper analysis of the existing guideline execution tools, we describe the methodology used to select them. We searched on the PubMed, SciFinder, ScienceDirect and Citeseer databases. In addition, proceedings of the most relevant conferences in the domain, such as the Conference on Artificial Intelligence in Medicine or the Symposium on Computer-Based Medical Systems, were also examined. Only relevant articles published between 2000 and 2007 and references therein were considered. The keywords used include: guideline-based execution engine, clinical guideline, computer-interpretable guideline, guideline workflow, guideline execution, and guideline enactment. All collected papers were analysed and filtered. Seven projects, six coming from academic research and one commercial system, were selected: *Arezzo*™, *DeGeL*, *GLARE*, *GLEE*, *NewGuide*, *SAGE* and

SpEM.

In the next sections these tools are succinctly described. The basic features of each tool are summarised, paying special attention to the language used to represent CGs and the proposed architecture.

2.1 Arezzo™

Arezzo™ is a commercial product to create, visualise and enact PRO*forma* guidelines developed at Cancer Research, UK [Fox et al. 2006, Fox et al. 2003*a*].

Clinical guideline representation

Arezzo™ uses the PRO*forma* language to represent CGs [Fox & Das 2000, Sutton & Fox 2003]. PRO*forma* is an executable process modelling language that has been successfully used to build and deploy a range of decision support systems, guidelines and other clinical applications. It has a declarative format defining four basic types of tasks (*plans*, *decisions*, *actions* and *enquiries*) as well as logical and temporal relationships between them. An *action* is a procedure to be carried out (usually by an external element like a doctor or a medical resource). A *plan* is the basic building block of a clinical guideline and represents a container for a number of tasks, including other plans. A *decision* is a task that represents an option in terms of different logic commitments to be accomplished. An *enquiry* is a request for further information or data required before proceeding with the application of the guideline.

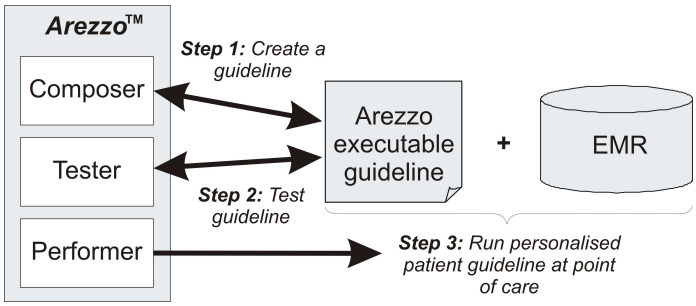


Figure 2.1: *Arezzo*™ architecture [InferMed 2007]

Architecture

The tool is composed of three elements: a *Composer*, a *Tester* and a *Performer* (see Fig. 2.1). The *Composer* is used to create guidelines using the PRO*forma* language. The *Tester* is used to test the guideline logic before deployment (it checks that the statements in decisions, tasks and enquiries are well written). The *Performer* inference engine can then run the guideline, taking into account data related to patients stored in existing healthcare systems (electronic medical record) [InferMed 2007]. During the enactment of a guideline in the performer engine, a task changes its internal state depending on whether it is awaiting for data, suspended, finished, or it cannot be accomplished in the current state of the patient.

Arezzo™ uses the Domino autonomous agent model ([Fox et al. 2003*a*]). The model deals with a large class of medical problems and establishes a relationship between decision making and plan enactment procedures. The main goal of this model is to identify the basic elements required in any language to represent clinical guidelines that can be used for both decision making and plan management. Fig. 2.2 shows the whole model, which is divided in two parts: the left side concerns the decision-making processes (steps 1-4), and the right hand side is related to planning and scheduling of tasks (steps 5-7). The process begins by taking into account a set of patient data. According to the model (step 1) the system proposes a set of possible causes of the health problem (step 2) and identifies a possible set of solutions (step 3) with its associated arguments *pro* and *con*. At this point, the doctor can identify a disease that has to be handled (step 4), and the cycle is started again to treat this specific disease. If, at this point, the doctor knows the appropriate option to be followed, he selects the therapy plan (step 5). The component steps of this plan will be scheduled (step 6), resulting in the execution of actions. An action will often produce postconditions that change the patient's state (step 7). This new information can produce new goals to be managed with other therapies. This is a cyclic model that produces a sequence of decision-making and scheduling steps [Fox & Das 2000].

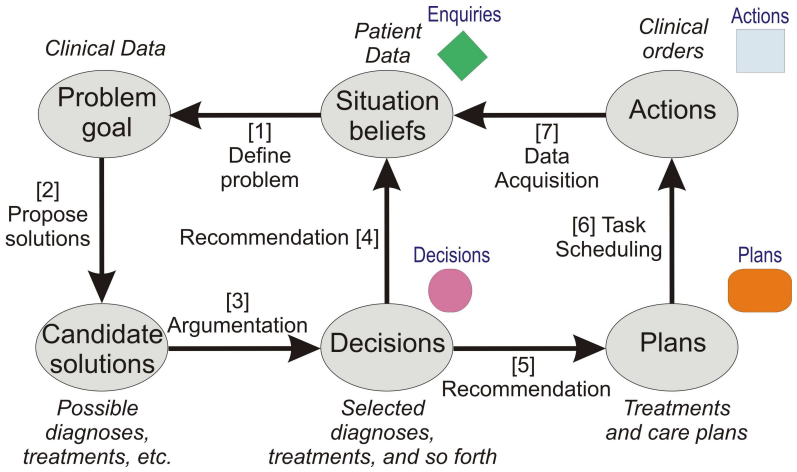


Figure 2.2: Generalised Domino model [Fox & Das 2000]

2.2 Digital Electronic Guideline Library

Digital electronic Guidelines Library (*DeGeL*) is a web-based, modular and distributed architecture, which facilitates the gradual conversion of clinical guidelines from text to a formal representation in Asbru [Shahar et al. 2004, Young et al. 2007]. It is being developed at Ben Gurion University in Israel.

Clinical guideline representation

The system maintains a repository of guidelines, and it allows the user to search, browse, retrieve and visualise all available guidelines. At the moment, the system creates guidelines using the formal language Asbru [Miksch et al. 1997], but the methodology could be extended to other languages.

One of the goals of *DeGeL* is to create formal guidelines from textual documents. The initial textual guidelines go through an intermediate layer between the textual and the final form, where experts add semantic information. The intermediate layer uses a meta-ontology that defines a hierarchy of basic concepts.

Asbru organises a clinical guideline as a library of Asbru plans created during the decomposition process performed during the specification phase. The Asbru plans in the library are interrelated in a hierarchical network of plans and sub-plans using a parent-child relationship which is encoded using control structures (*e.g.*, do in parallel) [Young et al. 2007]. Moreover, two types of plans can be distinguished: *atomic* and *composite*. Atomic plans represent a single action to be carried out (*e.g.*, administer a certain drug), whereas composite plans include a collection of atomic or other composite plans.

The system uses different standards to represent the clinical information: LOINC-3 for observations and laboratory tests, ICD-9-CM for diagnosis codes, and CPT-4 for procedure codes [Coiera 2003*a*].

Architecture

DeGeL is a modular system composed of a set of tools that support guideline classification, semantic mark-up, content-sensitive search, browsing, run-time application, and retrospective quality assessment (see the architecture of the system in Fig. 2.3).

A tool called *Uruz* allows practitioners or medical experts to create new medical guidelines. Another tool called *IndexiGuide* facilitates guideline retrieval. The run-time module is composed of several tools to test and visualise CGs. The tool named *VisiGuide* allows browsing and visualising guidelines. Another element is *Vaidurya* which allows both searching and retrieving CGs. *QualiGuide* is a tool that evaluates clinician adherence to clinical guidelines using the intentions of the guideline authors [Advani et al. 2003]. Finally, *Dipole* is a tool that assists clinicians to determine a patient eligibility and guideline applicability.

We focus the study in the *Spock* guideline execution module, which incorporates an inference engine that can retrieve data stored in a patient's medical record (see Fig. 2.4). The *Spock* system is a modular client-server application that consists of: i) a set of classes, that allow to store any guideline, ii) a parser, that interprets the content of a guideline, and iii) a specialized module, the *Controller*, which synchronizes the communication between the system layers and external services [Young et al. 2007, Young & Shahar 2005].

Spock proposes an asynchronous method to monitor all actions made in a guideline and allows to start and resume a CG execution as requested. The method, called *application log*, stores different data structures, like all state transitions of a *plan instance* (that is labelled with one of the following states: selected, activated, aborted, suspended or completed), a queue of scheduled awaiting tasks, and the list of recommended steps issued during application (see Fig. 2.4). All these data are stored in a centralised repository located on a remote server, accessible to any *Spock* client operated from a computer anywhere. Each guideline execution creates an *application-instance* for a patient, handled by a practitioner, with a CG to execute.

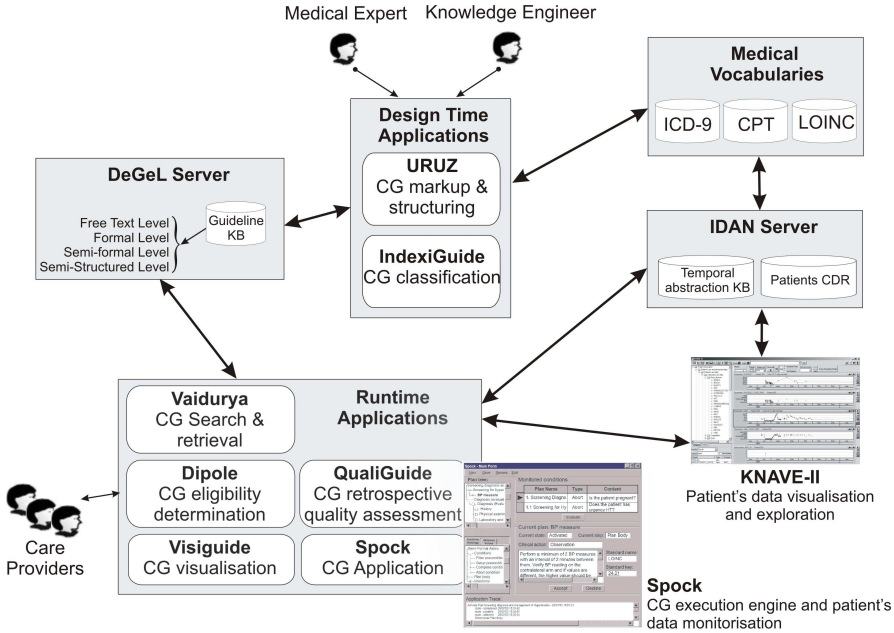


Figure 2.3: *DeGeL* general architecture [Shahar et al. 2004]

It begins with the construction of a set of *plan instances* in order to accomplish all plans included in the Asbru-coded CG, and proceeds with the *root* plan instance object [Young et al. 2007]. When this *root* plan instance object is terminated, either successfully (*i.e.*, in a completed state) or unsuccessfully (*i.e.*, in a rejected or aborted state), the application of the overall CG instance terminates. All of the plan instances are interconnected in a plan instance network (*i.e.*, parent-child relation), and changing their status according to the execution of their plan bodies.

2.3 Guideline Acquisition, Representation and Execution

GuideLine Acquisition, Representation and Execution (*GLARE*) is a system to acquire and execute clinical guidelines, developed at the Computer Science Department of the Università del Piemonte Orientale of Alessandria (Italy) in cooperation with Azienda Ospedaliera San Giovanni Battista of Torino (one of the largest hospitals in Italy) [Anselma et al. 2006, Terenziani et al. 2005, Terenziani et al. 2004, Terenziani et al. 2003].

Clinical guideline representation

Internally, CGs in *GLARE* do not use any standard representation. Their authors have defined a proprietary graph-based representation, where each action is represented by a node, while control relations are represented by arcs [Terenziani et al. 2001].

The *GLARE* designers distinguish between *atomic* and *composite* actions. *Atomic* actions are simple actions to be performed in a particular point of the guidelines. Four possible

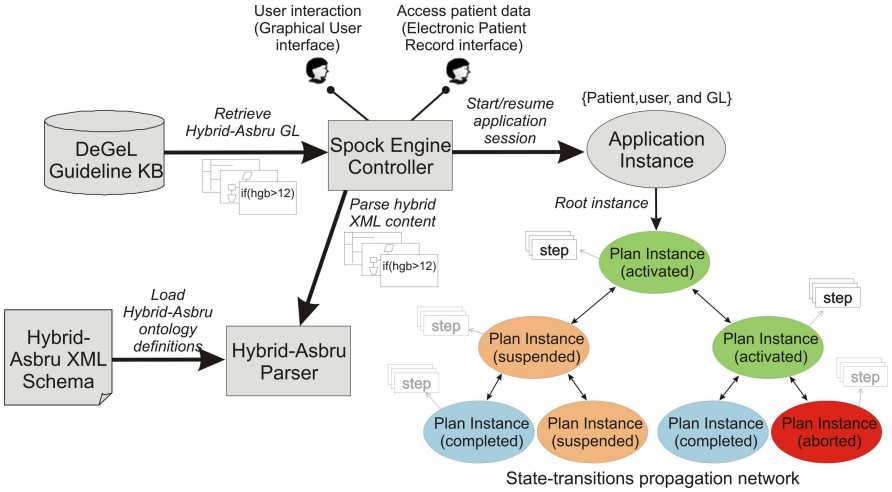
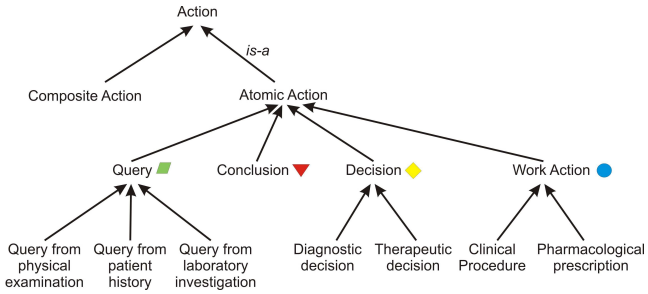


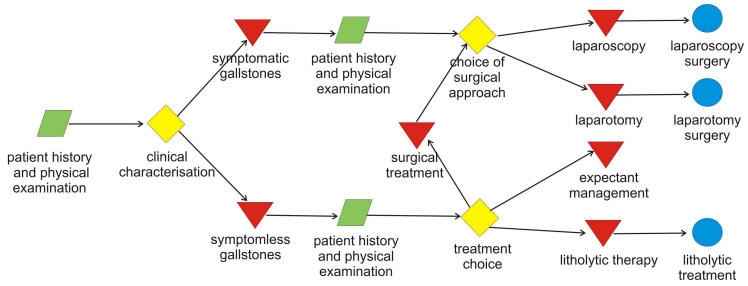
Figure 2.4: *Spock* general architecture [Young et al. 2007]

atomic actions were defined: i) *queries*, that allow to request any external information, ii) *work actions*, that represent actions to be performed, iii) *decision actions*, that embed a set of conditions to select an alternative among a set of actions that could be performed at that instant, and iv) *conclusions* that allow to describe outputs of a decision primitive. *Composite* actions are collections of atomic or other composite actions. For each action there is a set of *preconditions*, to be fulfilled before its activation, and a set of *conclusions*, that hold after the execution of the action. The *GLARE* execution engine maintains the current state of all actions and monitors all their preconditions before starting them. Fig. 2.5(a) shows the complete taxonomy of actions defined in *GLARE* as well as a specialisation of *queries*, *work actions* and *decisions*. In addition, Fig. 2.5(b) depicts a guideline example putting all the different primitives together.

Recently, the authors of *GLARE* have enriched the internal representation of guidelines in order to support temporal reasoning facilities such as consistency-checking during the creation of a guideline or checking whether actions could be performed *a posteriori* [Anselma et al. 2007]. This kind of functionalities were implemented by defining a temporal model that manages temporal constraints including constraints on repeated/periodic events [Anselma et al. 2006].



(a) Basic entities defined in *GLARE*



(b) Example of CG modelled using *GLARE*: the gallbladder stones treatment case study

Figure 2.5: *GLARE* representation overview (from [Terenziani et al. 2004, Terenziani et al. 2001])

Architecture

GLARE distinguishes between the *acquisition phase*, when a guideline is introduced in the system (*e.g.*, by a committee of experts), and the *execution phase*, when a guideline is applied by physicians to a specific situation (*i.e.*, it is instantiated on a given patient).

They define three layers called *System*, *XML* and *DBMS* (see Fig. 2.6). The *System Layer* contains the *Acquisition* and *Execution* modules. The lower level, called *DBMS Layer*, connects physically the higher levels with databases where all required data for both creating and executing guidelines are stored. There are data concerning available resources, terminology used in guidelines, information about drugs, information about all open instances of guidelines, a repository of guidelines, and medical records of patients. The intermediate *XML Layer* allows to exchange data between the *DBMS Layer* and the *System Layer* in a structured way [Terenziani et al. 2002]. The *XML Layer* defines an intermediate structure for each database that provides independence between the data and its use.

This system is focused in the management of *temporal constraints* between different actions in a CG, and the *Execution Module* allows to execute/simulate a CG using the appropriate retrieved data from each database. Each patient has his own medical record (contained in the *Patient DB*), which is updated continuously with the actions executed within a CG. The architecture is complemented with a database of available resources in a given hospital (*Resource DB*), that allows to make domain-dependent execution of guidelines. Moreover, *GLARE* allows the local adoption and update of guidelines to cope with both the need to apply them to new situations (countries, hospitals and/or departments), and with the need

to manage updates (*e.g.*, authoring, recording the history of a guideline and learning from experience) [Terenziani et al. 2005].

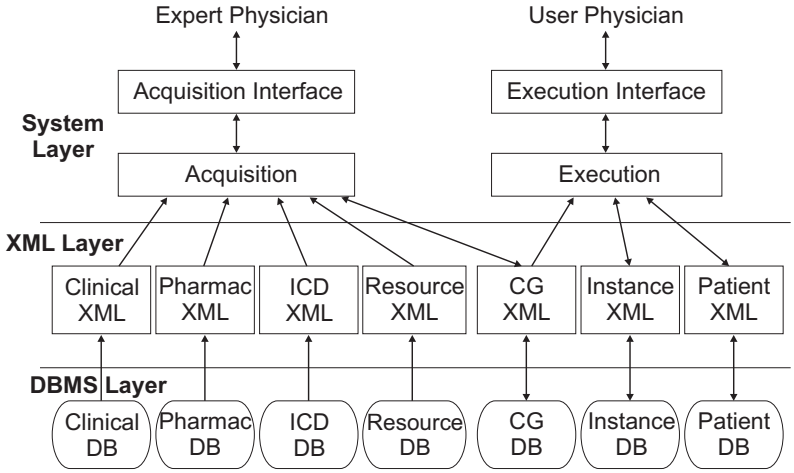


Figure 2.6: *GLARE* general architecture [Terenziani et al. 2003]

2.4 GLIF3 Guideline Execution Engine

GLEE is a tool for executing guidelines encoded in the 3rd version of GLIF (called GLIF3), which was developed across different institutions: the Department of Biomedical Informatics (Columbia University, US), the Stanford Medical Informatics Lab. (Stanford University, US), the Decision Systems Group (Brigham and Women's Hospital, Harvard Medical School, US), the Department of Management Information Systems (University of Haifa, Israel), and Eclipsys Corporation (Boston, US) [Wang et al. 2004, Wang & Shortliffe 2002].

Clinical guideline representation

GLEE handles guidelines encoded in the *GLIF3* language. *GLIF3* represents guidelines as flowcharts of temporally ordered nodes called *guideline steps* that store actions (*Action_Steps*), decisions (*Decision_Steps*), and clinical states of the patient (*Patient_Clinical_States*). There are two more types of nodes, called *Branch_Steps* and *Synchronization_Steps*, which are used for modelling multiple concurrent paths through the guideline [Choi et al. 2007]. Decision criteria are modelled using an OCL-based language (Object Constraint Language) called GELLO [Boxwala et al. 2004].

Guideline Execution by Semantic Decomposition of Representation (GESDOR) is an improvement of *GLEE*, which allows to represent clinical guidelines independently of the chosen representation language [Wang et al. 2003, Wang 2003]. The GESDOR model was tested with *GLIF3* and a subset of PRO*forma*.

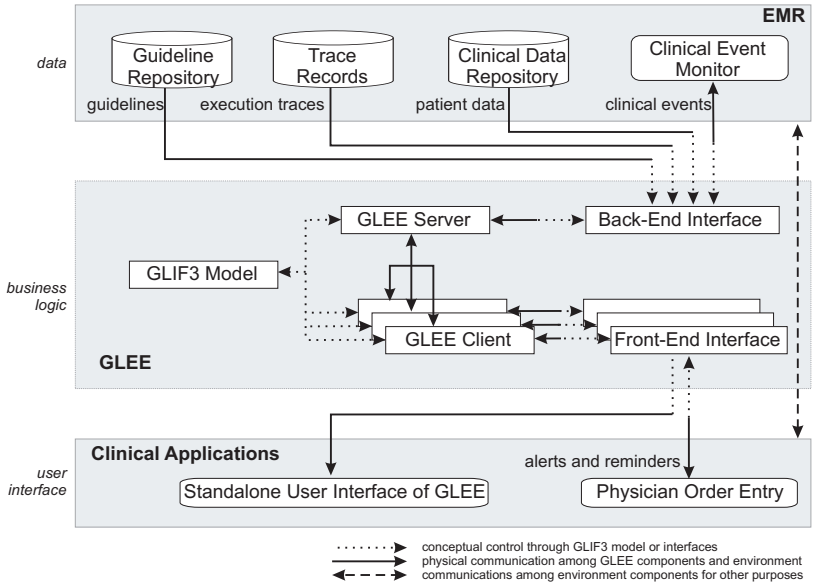


Figure 2.7: *GLEE* general architecture [Wang et al. 2004]

Architecture

As shown in Fig. 2.7, three levels of abstraction are defined in *GLEE*: *data*, *business logic* and *user interface*. The *data* level contains the EMR with a guideline repository and the *clinical event monitor*, that allows the execution (or simulation) of clinical guidelines through an event-driven model. The *business logic* level contains the GLEE execution engine, formed by a server and many clients. The server interacts with the data level, and clients interact with users (both through defined interfaces). At the bottom, we find the *user interface* level, where the clinical applications that exchange data with the upper levels are located.

The execution model of *GLEE* takes the *"system suggests, user controls"* approach. A tracing system is used to record an individual patient's state when a guideline is being applied to that patient. It can also support an event-driven execution model once it is linked to the clinical event monitor in a local environment. The tracing system allows to maintain two main views of the execution. One the one hand, *GLEE* suggests which actions can be performed and decides which actions (whose preconditions are satisfied) can change the state from started to finished. On the other hand, the user can control the process, and it can initiate, confirm or decide different transitions between actions. Fig. 2.8 shows the whole process maintained by the *GLEE* execution engine through the tracing system. Moreover, that tracing system was implemented as an external element and, to facilitate a further analysis (*e.g.*, evaluate the quality of suggestions or audit the sequence of performed actions), it stores all logs using XML (see *data* level in Fig. 2.7).

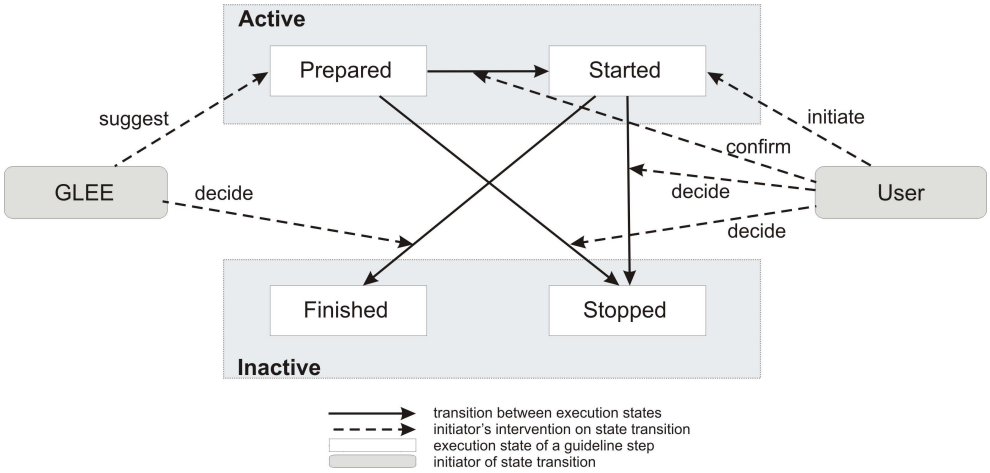


Figure 2.8: *GLEE* state-transition model [Wang et al. 2004]

In addition to serving as an interface to the *GLIF3* ([Boxwala et al. 2004]) guideline representation model, *GLEE* defines a collection of public methods (back-end and front-end interfaces shown in Fig. 2.7) to connect it with electronic medical records and other clinical applications to facilitate its integration with the clinical information system at a local institution. Concretely, two main (widely used) representations were selected to facilitate sharing information across different institutions: resource description framework (RDF) to store the guidelines, and HL7 as generic patient data model [Wang et al. 2004].

2.5 NewGuide

NewGuide is a framework for modelling and executing clinical practice guidelines developed at the Laboratorio di Informatica Medica, Università di Pavia, Italy [Ciccarese et al. 2004, Ciccarese et al. 2005].

Clinical guideline representation

Guidelines are represented using a representation language called GUIDE, which is based on Petri Nets [Quaglini et al. 2000]. It allows to model complex concurrent processes as well as temporal, data and hierarchical issues [Quaglini et al. 2001].

NewGuide uses the UMLS codification ([Lau & Shakib 2005]) to describe medical terms and procedures and a Medical Text Mark-up language called *Guideline text Mark-up* to describe tasks within a guideline [Kumar et al. 2002].

Architecture

GUIDE is integrated into a workflow management system which proposes an infrastructure that enables inter- and intra-organisational communication through a *Careflow Management System* (CfMS) that, on the basis of the available best practice medical knowledge, is able

to coordinate the care providers activities. The final goal of this architecture is to provide Health Care Organizations with technical solutions which should enable them to improve process efficiency, outcomes and quality of care.

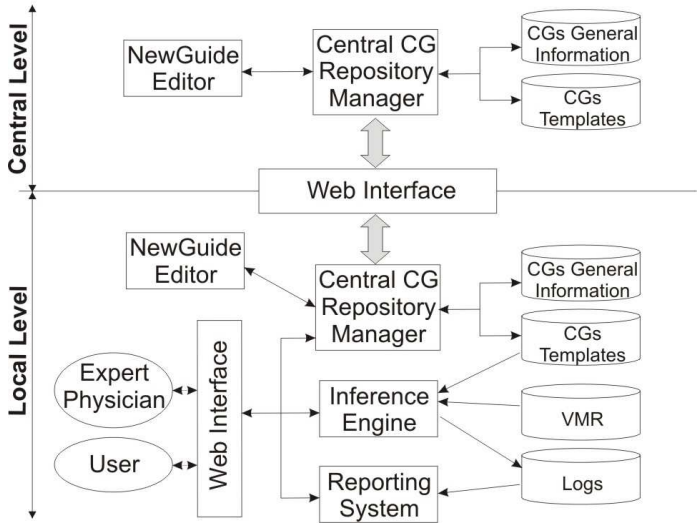


Figure 2.9: *NewGuide* general architecture [Ciccarese et al. 2004]

Fig. 2.9 shows the guideline management system proposed by *NewGuide*. First of all, the authors divide the guideline management in two levels: central and local. The former is intended to manage guidelines that cover a region, a country or several countries. These (general) guidelines are defined by some health authority or international organisation. The latter (a lower level of coverage) is localised in healthcare organisations that adopt global guidelines according to their particular requirements. In both cases, the creation and storage of those guidelines follow the same procedure: i) the CG is created/edited with the *NewGuide* graphical editor tool, ii) the *guideline repository manager* receives that information and splits it into two databases: the general information such as aim, eligibility, author, and version is stored in the first database which is used to search *a posteriori*, and the rest of the guideline (GUIDE content) is stored in a database of templates.

As shown in Fig. 2.10, the inference engine is composed of three main elements: a *general manager*, a *message manager*, and an *instance manager*. The *inference engine* is invoked by a clinician and automatically creates an instance of a CG (previously retrieved according to some criteria) for the management of an individual patient. All the steps followed in the execution of a guideline are supervised by an *instance manager*. At the same time, all instances are controlled by a *general manager*. After loading the guideline, the *instance manager* needs to collect all patient's data stored in his patient record. The execution engine goes step-by-step recommending actions, such as drug prescription or laboratory tests and, at the same time, stores that information in a logs database. All log data are used to monitor the status of a patient in the CG in another module named *reporting system*.

In addition, the communication between *NewGuide* and the external world is governed by the *message manager*, which delegates requests and responses to the web user interface

or to an external entity (through a SOAP interface) on the basis of the system configuration. The responsibility for maintaining the correct CG flow and timing is left to the external CfMS [Ciccarese et al. 2004].

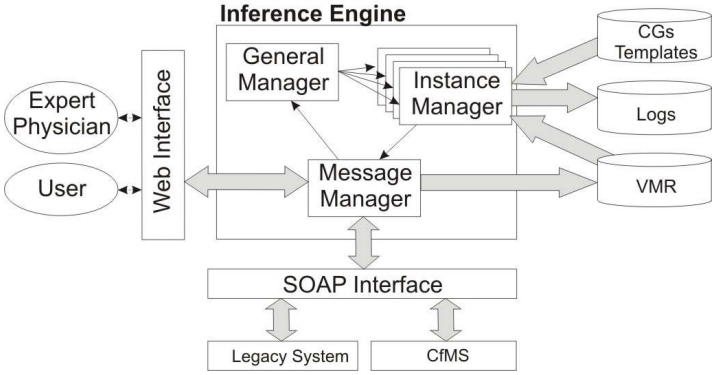


Figure 2.10: *NewGuide* inference engine [Ciccarese et al. 2004]

NewGuide authors have studied in detail the concept of *non-compliance with guidelines*. In [Quaglini et al. 2004] they analysed different factors that can cause a doctor not to follow a certain procedure; for example, a guideline does not provide the best recommendations for all patients under all possible circumstances, or a guideline can be applied in different ways depending on the clinical setting. For those reasons, guidelines should be evaluated *on the field* in order to assess both their applicability and the effectiveness of their implementation. This analysis can be a useful exercise because, according to the type of the detected non-compliance, improvements may be achieved by different interventions, such as site-specification of the guideline, users education, healthcare administrators involvement, and organisation re-engineering.

2.6 Standards-Based Sharable Active Guideline Environment

The *SAGE* project is a collaboration among research groups at six institutions in the US [Berg et al. 2004, Tu et al. 2006, Tu et al. 2004, Tu et al. 2007]. The project pursues two main goals. First of all, to create an infrastructure that allows medical experts to author and encode guidelines using a standard representation, and then, to use this infrastructure to deploy these guidelines across heterogeneous clinical information systems.

Clinical guideline representation

The internal representation of guidelines in *SAGE* is made using the EON formalism which is comprised of a set of Protégé classes and plug-ins [Gennari et al. 2003, Tu & Musen 2001].

Fig. 2.11 shows a portion of a *SAGE*-defined guideline and how the elements should react to the events in the care process. *SAGE* defines two different formalisms: *recommendation-set* and *decision-map* [Tu et al. 2006].

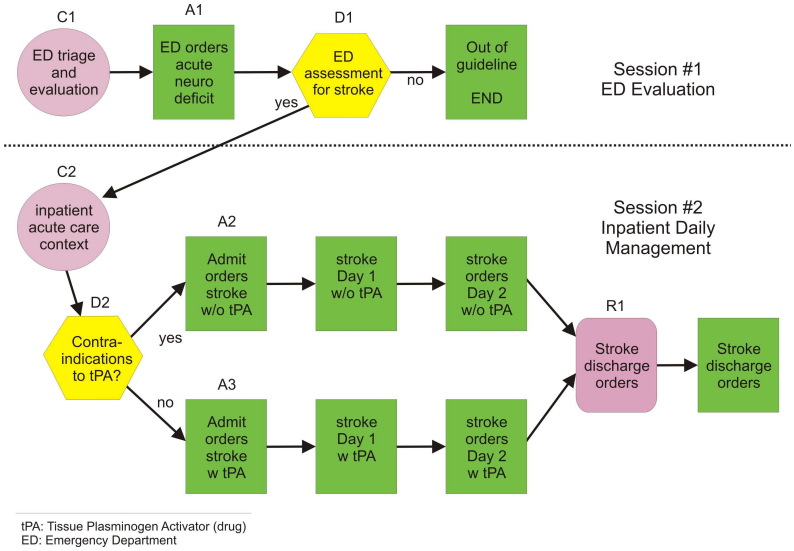


Figure 2.11: The top-level process specification in *SAGE* (adapted from [Ram et al. 2004]).

The *recommendation-set* is an activity graph composed of processes and interactions between them. Activity graphs allow the specification of computational algorithms or medical care plans as processes consisting of i) *contexts*, that are combinations of a clinical setting (*e.g.*, outpatient visit in a general internal medicine clinic), care providers to whom the recommendation is directed, relevant patient attributes (*e.g.*, patient age), and possibly a triggering event (*e.g.*, a patient checking into the clinic), ii) *decision nodes*, that evaluate conditions on variables (*e.g.*, a Boolean precondition for an action), iii) *action nodes*, that encapsulate a set of work items that should be performed either by a computer system or by a healthcare provider, and iv) *routing nodes*, that are used purely for branching and synchronization of multiple concurrent processes. In Fig. 2.11, $C1$ and $C2$ represent *context nodes*, $A1$, $A2$ and $A3$ are *action nodes*, $D1$ and $D2$ are *decision nodes*, and $R1$ is an example of a *routing node*.

Architecture

The global architecture of the system, that includes a guideline execution engine as its central component, is shown in Fig. 2.12. It is important to note the integration of this engine with current clinical applications, and also the definition of a central core of terminologies (models that detail not only medical data but also patient data, care workflow processes and the structure of health care organisations) [McClure et al. 2006].

The execution engine, called *SAGEDesktop*, is implemented as a centralised element. Given a guideline, it collects the required data from an internal repository and allows medical experts to emulate the real guideline behaviour [Berg et al. 2004]. As shown at the bottom of Fig. 2.12, the execution engine interacts with the clinical information system (CIS) via an event listener and a set of services (*terminology*, *patient record* and general *applications*). The *terminology server* was added to customise the terms used in some specific local applications. Calls to/from the execution engine and the CIS are made through a set of defined APIs, which

allow interoperability with existing systems. Services related to the EMR (invoked using *medical record calls*) allow the engine to retrieve the appropriate patient data. After that, the *action service calls* allow the engine to initiate actions within the CIS.

The *SAGE* project proposes a guideline model with the following features:

- It uses standardized components that allow interoperability of guideline execution elements with the standard services provided within vendor clinical information systems. It proposes the use of a repository of CGs in order to manage all available guidelines.
- It uses standards to represent the data (EMR and processes) such as SNOMED-CT and Health Level 7 (in particular, HL7*v3*) [Lau & Shakib 2005].
- It includes organizational knowledge to capture workflow information and the resources needed to provide decision support in enterprise settings. It proposes a methodology to develop/create medical guidelines.

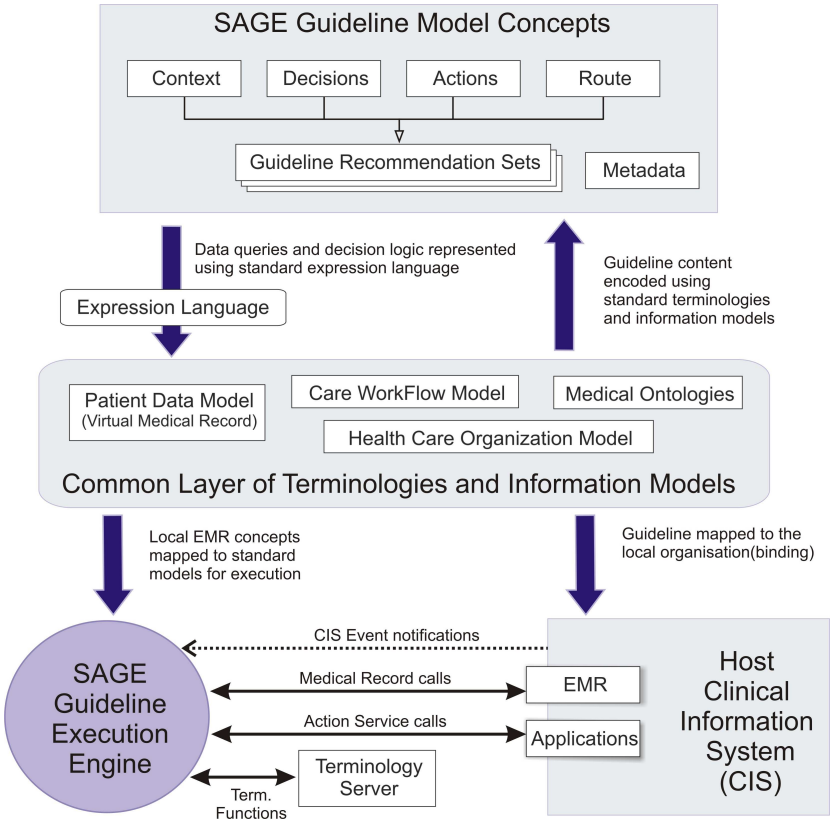


Figure 2.12: *SAGE* global architecture (adapted from [Tu et al. 2004, Ram et al. 2004])

2.7 Specification Execution and Management Plan

*Sp*ecification *E*xecution and *M*anagement Plan (*SpEM*) is a framework for supporting the management of clinical guidelines [Dube 2004, Dube et al. 2005, Dube & Wu 2006] which was developed at the Department of Computer Science, Dublin Institute of Technology, Dublin, Ireland.

Clinical guideline representation

The authors defined a model that allows to both create and execute clinical guidelines. First of all, a general purpose language called PLAN was adapted to represent clinical guidelines. That language follows an event-condition-action (ECA) approach, based on rules [Mansour et al. 2006]. Fig. 2.13 shows the defined syntax for these rules and an example. This ECA rule mechanism is mapped into an existing Database Management System (DBMS), that is at the end who performs the enactment of a specified guideline through rising and managing different triggers [Dube & Wu 2006]. The execution module embedded in a DBMS uses these rules to start a guideline and, according to the raised events, activate an specific task. An ECA rule is composed of three elements: a) an *event* part, containing a so-called transition predicate that lists all possible events which are of concern to the rule (it constitutes the situation that the rule has to monitor), b) a *condition* part, which can be an arbitrary predicate, and c) an *action* part, which is an arbitrary list of executable functions.

SpEM defines the following primitives that are required in a guideline or protocol representation model: a) an *action*, which represents any clinical or administrative task that is recommended to be performed, maintained, or avoided during the process of guideline application, b) a *decision*, that is a selection from a set of alternatives based on predefined criteria in a guideline, c) a *patient state*, which is a materialisation of a treated individual's clinical status based upon the actions that have been performed and the decisions that have been made and, d) an *execution state*, which is a description of a guideline's current state [Dube 2004].

```
%PLAN Syntax
<dynamic-rule> ::= <rule-header><rule_body>
<rule-header> ::= RULE <rule-name>,[<description>,]
<rule_body>::= ON: <event_spec>, IF: <condition_spec>, DO: <action_spec>;
<event_spec> ::= <event_name> ( [<parameter_list>] )
<condition_spec> ::= <condition> | <condition> {AND | OR} <condition_spec>
<action_spec> ::= <action> | <action>, <action_list>
<condition> ::= logical condition
<action> ::= <action_name> ( [<parameter_list>] )

%example
RULE ma1sdr1,
DESCRIPTION: rule to order test B if A result is abnormal,
ON: result_arrival('A'),
IF: A > 8.5,
DO: order_test ( 'B' );
```

Figure 2.13: *SpEM* syntax rules [Dube et al. 2005]

Architecture

As shown in Fig. 2.14, *SpEM* was designed as a layered framework. The highest layer implements the Guideline Management services and it is divided in three main components: *Specification plane*, *Execution plane* and *Manipulation plane*. The first module is able to capture clinical guidelines in a formal way. It provides methods to access, store and manipulate guidelines represented using Protocol Language (PLAN) [Dube 2004]. The *Execution plane* provides methods and tools to ease the creation and execution of patient-centred guidelines adapted from guidelines stored in the repository. The last module, the *Manipulation plane*, provides facilities to query and operate on guideline information through a high level language called TOPSQL [Dube 2004]. The lower layer implements the active rule extensions through an existing DBMS which supports the ECA mechanism. Between those two layers, the authors added an intermediate layer to support collaborative features (*e.g.*, manipulation of guidelines between several practitioners) and sharing facilities in a generic way (to allow re-use by different applications) [Dube et al. 2005].

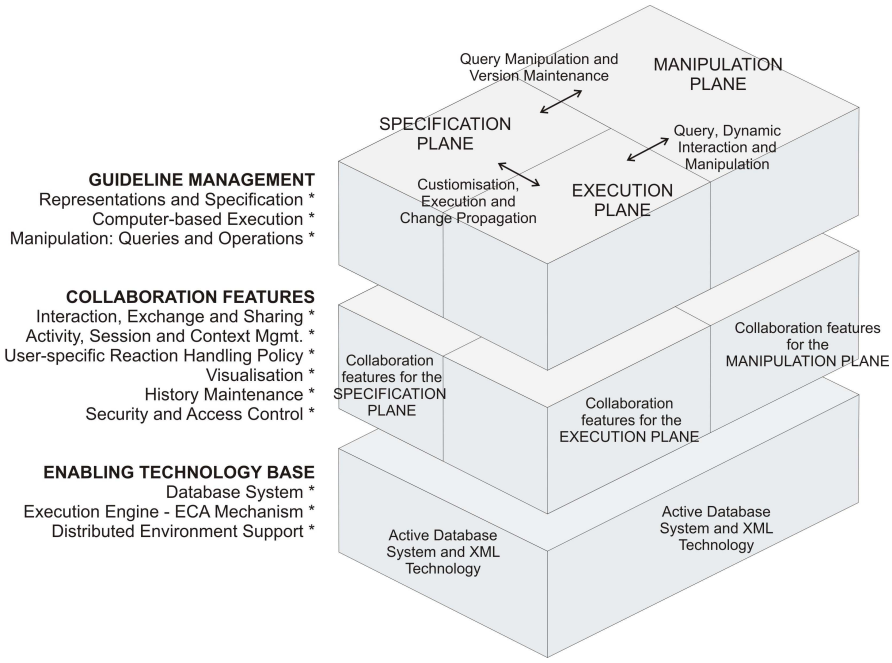


Figure 2.14: *SpEM* framework (adapted from [Dube 2004] and [Dube et al. 2005])

2.8 Comparison

In the previous section several computer-based systems that manage clinical guidelines have been briefly described. All of them have different scopes, representations and architectures, according to the research interests of each developing group. This section provides a high level comparison of these tools, focused on the items in Table 2.1. These items are the following:

a) the existence of a repository of guidelines,

b) the presence or absence of a tool offering a (graphical) editor to create and visualise its own guidelines,

c) the formal language used to represent the clinical guidelines,

d) the basic elements defined in the guideline representation language,

e) if the tool is designed to be deployed as a distributed system,

f) the presence of complex coordination elements such as parallelism, negotiation or scheduling,

g) the type of execution engine (there are different approaches to follow a guideline such as event-based (EB) and rule-based (RB)),

h) the connection of the system with an electronic medical record (EMR)[1],

i) the ability to integrate the execution engine with an existing clinical management system (CMS)[2],

h) the use of any standard terminology or representation language, and finally

i) the inclusion of security tools to preserve data integrity and authenticate the accesses to the (very sensitive) medical data exchanged in these systems.

[1] The main goal of the EMR is to store and maintain a computerised patient record. It stores all the medical history of the patient with information about results and pending tasks.

[2] A CMS is a complex system that includes the management of patients (with access to the EMR) as well as the management of resources, staff, orders and prescriptions.

Repository of guidelines

All the tools offer a repository of guidelines. It allows to use the best available guideline at each moment and/or to update them when necessary. In some cases (*GLARE*, *NewGuide*) the repository stores several versions of a CG, allowing versioning. In particular, *DeGeL* implements some tools to index the stored guidelines and to enable an automatic search by another decision support system. This feature can be used to update the CGs or to tailor a general CG to different medical centres according to the resources available in each location. Moreover, HECASE2 proposes to distinguish the knowledge available in different departments allowing different repositories for each department in each medical centre. That distribution of the knowledge has several advantages (*e.g.*, to allow a personalisation of CG to the particular circumstances of the department or the versioning maintained by the doctors of a medical centre), but it hinders the management of all available CGs.

Guideline editor

The use of an editor to create guidelines is a recommended tool to ease the visualisation and updating of guidelines. Most of the described tools offer an editor, which translates the clinical guidelines into the chosen representation language. Moreover, most of them identify several basic components (actions, decisions, and queries) which are linked together using a flowchart-based approach.

Arezzo™, *DeGeL*, *GLARE*, *GLEE*, *NewGuide* and *SAGE* have a (graphical) editor to create guidelines in their own representation. Only *SpEM* lacks this module, as it translates CGs directly to database event rules.

Even though a deep analysis of the editors is out of the scope of this manuscript, a good feature that could be implemented in these systems is a web-based edition. In addition, collaborative features to improve the creation of CGs among different medical experts would be very useful.

Table 2.1: Comparison of guideline execution engines

Tool	*CG Repository*	*CG Editor*	*CG Representation Language*	*CG basic elements*	*Agents*	*Coordination*	*Runtime Engine*	*Access to EMR*	*Access to CMS*	*Standards used*	*Security*
Arezzo™	Yes	Yes	PRO*forma*	Action, Decision, Enquiry, Plan	Yes	Yes	RB	Yes	Yes	No	No
DeGeL	Yes	Yes	Asbru	Logic statements	No	Yes	RB	Yes	No	ICD9, CPT, LOINC, XML	No
GLARE	Yes	Yes	Graph-like	Query, Work, Decision, Conclusion	No	Yes	RB	Yes	Yes	XML, ICD9	No
GLEE	Yes	Yes	GLIF3	Action, Decision, Branch, Synch., Clinical Stage	No	Yes	EB	Yes	Yes	HL7, RDF	No
NewGuide	Yes	Yes	GUIDE	Petri Net	No	Yes	RB	Yes	Yes	UMLS	No
SAGE	Yes	Yes	EON	Context, Decision, Action, Route	No	Yes	EB	Yes	Yes	HL7, UMLS	No
SpEM	Yes	No	PLAN	Event-Condition-Action	No	No	EB	No	No	XML	No

Language used to represent the computer-interpretable clinical guidelines

There are several available languages and models to represent guidelines [Clercq et al. 2004, Peleg et al. 2003, Quaglini & Ciccarese 2006]. This is an important drawback because it prevents researchers from implementing tools in the same way, and there is not any *de facto* standard language. As summarised in Table 2.1, all platforms define their own language or representation structures, according to their specific goals.

Fortunately, *Arezzo*™, *DeGeL*, *GLEE*, *NewGuide* and *SAGE* use structured and well defined languages that can be read, parsed and analysed by a program. That means that these languages can be re-used by other organisations in order to implement an *ad-hoc* guideline execution engine or to improve those functionalities (*e.g.*, to edit, to store, to validate) to allow for example a collaborative management.

Guideline representation languages define the *declarative knowledge* (know-how) of complex medical pathways. The information given in a CG represents the medical knowledge and allows to implement decision-support services at the right time. Although this information is very valuable, process-oriented knowledge (or tacit knowledge) that describes organisation goals, roles and responsibilities, and communication or coordination patterns of the care process, is also required [Peleg & Tu 2006, Stefanelli 2004]. Normally, this knowledge is represented outside of the CG (one of the most used representations is UML) although systems such as *NewGuide* include both kinds of knowledge using the same CfMS paradigm [Quaglini et al. 2000]. Developing guidelines is essentially a consensus process among medical experts. Yet, there is a gap between the information contained in published clinical practice guidelines and the knowledge and information that are necessary to implement them [Bates et al. 2003, Lenz & Reichert 2005, Shiffman et al. 2004]. Of the studied CG representation languages, PRO*forma* (used by *Arezzo*™) is the only approach that makes a distinction between a declarative language (R^2L), used during the guideline acquisition phase, and a procedural language (L_{R2L}) that is processed by a general interpreter (PROLOG in this case) in an execution engine [Fox & Das 2000]. All other approaches require a custom-developed execution engine, in which the different procedural aspects of the guideline are encoded programmatically [Clercq et al. 2004].

Basic elements defined in the guideline's representation language

As shown in the previous item, there are different languages to represent clinical guidelines, but some of them share common features. Systems such as *Arezzo*™, *GLARE*, *SAGE* and *GLEE* use similar basic elements (actions, decisions and enquiries). The main difference is related to the management of the links between those elements (*e.g.*, the use of temporal reasoning).

Those languages embed declarative knowledge using different primitives like cycling iterations, logical expressions in decisions, control structures for synchronising sequences of actions, and evaluation of pre and post conditions before and after the execution of an action.

It is feasible to think that these approaches could converge into a *common language* to cover all functionalities identified in these representations. Nowadays, the fact that there is no standardisation of representations implies that hospitals implement *ad-hoc* solutions in most of the cases. In this sense, there were several attempts ([Shiffman et al. 2004, Boxwala et al. 2001, Peleg et al. 2004]) to find a common representation format with limited success.

Agents

Although most of the papers do not use the terms *agent* or *multi-agent system* ([Wooldridge 2002]), the authors of the analysed systems propose distributed architectures (usually a client-server approach) with both data and tasks deployed around a computer network. In particular, *GLARE*, *GLEE* and *NewGuide* define different types of decentralised systems. Some authors propose an event-driven approach, similar to the communication-based approach that is the basis of multi-agent systems. Moreover, some of the proposed modules act autonomously, for instance run-time engines, and they could be easily mapped into agents. The *Arezzo*™ experience, currently being developed, show how a guideline execution system can be developed using an agent-based perspective with a formal language such as PRO*forma*, which has been designed to interact with external elements linked to agents. This perspective allows to design interoperable platforms that could be extended, in a flexible way, with more services and resources as required in a specific clinical setting. This extensibility and flexibility is especially clear in the HECASE2 system developed in the present work.

Coordination

Clinical guidelines define different tasks to be accomplished. In any run-time engine it is very important to coordinate these tasks efficiently in order to improve the global performance. For instance, some tasks to be performed can have a time constraint (deadline) that should be considered before tackling other tasks; at this point, the system must perform a booking between the patient and the resource that manages the required task. The execution of a CG requires planning, negotiation and scheduling between all entities, resolving conflicts and verifying the consistency of all partial results [Jennings 1996]. In the analysed systems, a run-time engine (central element) controls the execution of the CG. Only *SpEM* does not show coordination features and the DBMS is who simulates the enactment through the activation of events and rules.

Run-time engine

A run-time engine is required to simulate the behaviour of a clinical guideline with the patient data values. There are two approaches to perform the simulation: an event-based approach (EB in the table), such as *SpEM*, *GLEE* and *SAGE*, and a rule-based approach (RB), such as *NewGuide*, *GLARE*, *DeGeL* or *Arezzo*™.

The difference between event and rule-based approaches is how the systems are used. The former approach can be used in a continuous system and the events are handled asynchronously as they appear (*e.g.*, the arrival of a patient's result). The later should be monitored by another partner that supervises and controls the rules that can be activated in any moment in a synchronous way.

Access to an electronic medical record

As reported in Holbrook et al. (2003) and Hoyt et al. (2007), there are dozens of implementations of electronic medical records with different functionalities. In all cases a gateway between the guideline execution system and the computerised patient record is needed to retrieve the required patient's data at each moment. A knowledge base is usually employed to know exactly which attribute is required and to allow the system to find it within the EMR.

Most of the analysed tools implement an interface that enables the communication with a proprietary EMR representation. Usually, the guideline execution engines define a set of (general-purpose) interfaces that should be customised for each EMR and provide independence between the data and its use. This integration is the main problem of all guideline-based execution systems. The data definitions required in (formal) guidelines may not map to the data available in an existing EMR, and in practice an extension or modification should be made for every case [Lenz & Reichert 2005, Coiera 2003*b*]. Recently, Peleg *et al.* [Peleg et al. 2008] have integrated *GLEE* with two existing EMRs through an ontology-based interface that translates all requirements of the guideline into SQL queries to perform in the EMR.

Access to an existing clinical management system

Extending the previous item, a more general criterion to evaluate is to know whether the systems were designed to be included in an existing clinical management system [NCQHC 2006]. Normally, these systems provide general functionalities in a healthcare institution, which could be used by a guideline execution engine. These functionalities mainly comprise the management of information about the available human resources. All the suggestions made by the guideline execution engine can be used to audit and evaluate general behaviours, identify bottlenecks or dysfunctions between the guideline and the daily practice.

Arezzo™, *GLEE*, *NewGuide* and *SAGE* were designed explicitly with this gateway between those systems (the clinical management system and the guideline execution engine). *GLARE* can also be connected with an existing system but it requires to customize the XML Layer between the system layer and the databases. The rest of the systems were not designed to be embedded into existing systems, and this functionality would require to add new modules or re-design the systems.

Towards this integration, researchers have investigated the way to define and implement interfaces to include clinical decision support systems into different commercial hospital information systems (HIS). One of the first attempts in this domain was made by Müller *et al.* [Muller et al. 2001] who designed two prototypes to include a knowledge-based system into two different commercial HISs. They used XML, CORBA, Java and JDBC calls in order to implement some wrappers to allow getting and putting data into HISs. Another work in this field was made by Schadow *et al.* [Schadow et al. 2001], who designed a model to integrate guidelines in an existing electronic health record. In this case, the authors designed a HL7-based set of classes embedding logical conditions and goals, and obtained an homogeneous representation for both declarative knowledge and data. Finally, Veselý *et al.* [Veselý et al. 2006] described the integration of a GLIF-based system with an existing information system. In this case, the system was designed as an alerting system but also included a wrapper of the existing system in order to control the changes and get the required data.

Standards used

Medical terms can change significantly their meaning with little syntactic changes. A solution for that problem is the use of a standard terminology. In [Lau & Shakib 2005, Coiera 2003*b*] there is an analysis of a set of currently available approaches, including those used in the tools analysed in this paper. UMLS, which is used in several of the studied systems, links the major international terminologies (*e.g.*, SNOMED, MESH, LOINC and ICD) into a common

structure, providing a translation mechanism between them. Each medical term is labelled with a Code Unique Identifier. Medical terms (CUIs) have a semantic type (*e.g.*, diagnostic procedure, finding, body part), a definition, synonyms, and a collection of relationships with other CUIs (*e.g.*, isA, isPartOf).

The use of a well-known nomenclature is an advantage in order to facilitate the dissemination and automation of the execution of clinical guidelines under any representation. Comparing coding systems is an arduous task and it is context-dependant. Depending on the situation, a codification is more suitable than other (*e.g.*, UK terms may not perform as well in US-designed systems, or terms in primary care may differ from others used in hospitals). In all these cases, a hand-made human supervision of the used terms should be performed in any guideline acquisition process. When this supervision is made, the use of a codification establishes the basis to interact with any computerised patient record.

In addition, structured languages such as XML or RDF can be used to represent the guidelines internally. This functionality facilitates sharing and reusing of existing elements among the different entities involved in the management and creation of guidelines. *GLARE*, *GLEE*, *DeGeL* and *SpEM* use this kind of representations.

Security issues

Medical data is sensitive and has to be managed accurately. Transmissions, accesses and storage need a secure handling. To ensure secure transmissions, a ciphering of contents needs to be made [Simone 2001]. To ensure secure access, only registered (and authenticated) agents have to be permitted to exchange information with any other agent of the system. Finally, data should be stored in a secure database with authentication controls of the agents and users that want to access it. None of the analysed systems implements explicit security mechanisms. As will be shown in future chapters, this issue has been addressed in the HECASE2 system.

In addition, as reported in [Holbrook et al. 2003], one of the basic functionalities analysed for a set of EMRs, is the provision of log-on/log-off procedures and security issues. The authors of this study identified 15/40 systems as *"full"* EMRs[3]. A future line of research is to adopt these *"full"* EMRs in a guideline execution engine and to take advantage of these facilities.

2.9 Discussion

This state-of-the-art has described the basic aspects of several applications oriented towards the automation of clinical guidelines.

As a result of this survey, several limitations were identified and should be tackled in the future. One of them is the representation of computer-interpretable guidelines. The representation language is the basis of these tools, and it could be desirable to adopt one formalism as standard and promote the interoperability between different tools and systems. This standardisation seems quite feasible, as most of the representation languages commented in this

[3] A *"full"* EMR is understood as a system that implements: log-on/log-off procedures, security; patient database; charting encounters; prescriptions; chart summary/cumulative patient profile; reports; access to on-line information; referral letters; querying the database, practice research; health maintenance; patient education; laboratory orders and retrieval; and communications and productivity.

survey share the same basic components: some kind of action/decision/enquiry nodes, some mechanisms for coordination or synchronisation of actions, the ability to create sub-plans or sub-guidelines (so that different levels of abstraction can be considered when working with guidelines), and the possibility of storing the state of a guideline which is being executed.

Another important element of such automation is the existence of an electronic medical record. The problem of finding the best possible representation of an EMR has not been solved yet, and applications are made *ad hoc* to fit a certain representation. This also limits heavily the interoperability of different tools.

It is also fair to say that in most countries (even the ones considered to be more technologically advanced) health care is not yet fully computerised, and nowadays it is unrealistic (or it is hard and expensive) to include automated guideline enactment systems in real clinical settings. Most of the described systems consider as a big concern the seamless integration of the execution of guidelines with the usual workflow of activities within a medical centre, in order to make it feasible to introduce this kind of systems in daily clinical practice.

A common feature in most of the analysed systems is the internal organisation. Basically, we identified three main levels: a level with the patient's related data, an intermediate level with the execution engine, and a level that contains a set of interfaces to connect the execution engine with external devices. This approach provides transparency between the data and its use and allows to improve each element independently.

Clinical guidelines include sets of rules that a doctor can follow in a specific situation (diagnosis, treatment, or prognosis). Coordination between humans and resources according to these rules is required to follow a guideline in a coherent way (ensuring the satisfaction of all relevant constraints). In a centralised model, coordination protocols are difficult to implement or the amount of data to be exchanged could suppose a bottleneck that could hinder system performance. Due to these shortcomings, the inclusion of a distributed system (*e.g.*, multi-agent system, [Wooldridge 2002]), as proposed in this work, is a step forward in the development of guideline execution systems.

2.10 Conclusions

This chapter analysed and compared different guideline-based execution systems. A brief list of conclusions of this review is the following:

- It is widely accepted that the adoption of guideline-execution engines in daily practice would improve the patient care, by standardising the care procedures.
- A guideline stores medical knowledge (declarative) about medical procedures. It is important to use a common vocabulary and adopt one of the available terminologies to permit reusing, learning and sharing this modelled knowledge. Moreover, organizational knowledge to describe the roles and allowed actions for all actors involved during the execution of a CG, would allow to describe explicitly the *know-how* behaviour. These data should be represented outside the clinical guideline in order to facilitate its reusing among different kind of organizations (*e.g.*, different countries or different medical centres).
- The quality of a guideline depends on both its acquisition and its verification. The former has different approaches and there is not any standard or widely used language. The latter is not fully implemented and should be addressed by researchers. One of the most interesting challenges is to design guideline execution engines independent of the representation language used.
- Guideline-based systems can be embedded in a knowledge-based decision support system in order to deliver the ”right knowledge to the right people in the right form at the right time” [Schreiber et al. 1999]. In this sense, healthcare is becoming increasingly patient-centred and individualised, with the patient becoming an active subject rather than a mere object of healthcare and, these kind of systems can adopt user preferences to deliver the services [EC 2007].
- Guideline-execution systems should implement appropriate coordination techniques to perform a complex distributed task (careflow), and to minimise errors and delays between all transitions.
- The inclusion of a guideline-based system or, in a more general way, a clinical decision support system into an existing (commercial or not) electronic medical record system is hard because they are designed as closed monolithic systems without interoperability methods.
- This is an ongoing research area with numerous researchers working on it, designing and implementing useful execution systems that could potentially add some benefits to the daily practice.
- As suggested in *Arezzo*™ (and also in this work), agents can be a feasible option to optimize a guideline execution system in different levels: an efficient collection of data from different heterogeneous repositories (maybe with the addition of wrappers), the coordination of tasks during the execution, and finally, and improvement of the general throughput of the system in front of a centralised one.

Chapter 3

Multi-agent systems applied in healthcare

Healthcare is a large domain where some researchers have applied different AI techniques and algorithms. Our study is focused in the application of MAS, which allow to implement distributed systems with several benefits in front of centralised ones. Moreover, using this approach we can reuse existing applications and increase the general performance of the system.

In this chapter we study the intelligent agents paradigm and its adoption in the healthcare domain. This chapter is organised as follows. The next section describes the main characteristics of a MAS, summarising a list of features that an agent may exhibit, and concluding with a list of properties emerging from their use. Section 3.2 analyses the characteristics of the problems in the healthcare domain in order to explain the adequacy of adopting MAS to solve them. Finally, Section 3.3 reports the domains of healthcare where agent technology has been applied (*e.g.*, planning and resource allocation, decision support systems, information management), summarising the most interesting works implemented in those domains.

The application of intelligent agents in the healthcare domain is an active and interesting research area, which holds specific workshops (collocated with the most important AI conferences such as IJCAI, ECAI or AAMAS), special issues (published in journals such as IEEE Intelligent Systems and AI Communications), books (*e.g.*, Agent Technology and e-Health [Annicchiarico et al. 2008]), and research groups working around the world.

3.1 Multi-agent Systems

First of all, we will describe what is an agent and the main benefits that we can obtain in any agent-based application. An agent can be defined as follows [Wooldridge 2002]:

> *An agent is an entity that must be able to perceive the physical or virtual world around it using sensors. A fundamental part of perception is the ability to recognise and filter out the expected events and attend to the unexpected ones. Intelligent agents use effectors to take actions either by sending messages to other agents or by calling application programming interfaces or system services directly.*

The main distinguishing characteristics of intelligent agents are the following [Wooldridge & Jennings 1995]:

- *Autonomy*. Agents operate without the direct intervention of humans or others, and have some kind of control over their actions and internal state.
- *Reactivity*. Agents perceive their environment (physical world, a user, a collection of agents, the Internet, or a combination of all mentioned entities) and respond in a timely fashion to changes that occur in it.
- *Pro-activeness*. Agents do not simply act in response to their environment, they are able to exhibit goal-directed, opportunistic behaviour and take the initiative when appropriate.
- *Social ability*. Agents interact with other agents (and humans) via some kind of agent-communication language when they recognise the necessity of such communication (usually with the aim to complete their own problem solving and to help others with their activities).

MAS are applications in which many autonomous software agents are combined to solve large problems. A MAS has the following interesting properties:

- *Modularity:* the different services or functionalities may be distributed among diverse agents, depending on their complexity. In addition, a MAS allows for the interconnection and interoperation of multiple existing legacy systems. By building an agent wrapper around such systems, they can be incorporated into an agent society.
- *Efficiency:* a MAS enhances overall system performance, specifically along the dimensions of computational efficiency, reliability, extensibility, robustness, maintainability, responsiveness, flexibility, and reuse.
- *General performance:* a MAS distributes computational resources and capabilities across a network of interconnected agents. Whereas a centralized system may be plagued by resource limitations, performance bottlenecks, or critical failures, a MAS is decentralized and thus does not suffer from the *single point of failure* problem associated with centralised systems.
- *Flexibility:* agents may be dynamically created or eliminated according to the needs of the application. Negotiation and knowledge exchange allow the optimisation of shared resources.

- *Existence of a standard:* the Foundation for Intelligent Physical Agents (FIPA - [FIPA 2002*a*]) is an IEEE Computer Society standards committee that promotes agent-based technology and the interoperability of its standards with other technologies. Its main mission is to establish the rules that have to govern the design and implementation of a MAS in order to achieve interoperability among systems. Since 1997 it has been releasing specifications that have been slowly gaining acceptance and have turned into *de facto* standards in the agents community[1]. Due to this fact, any of the agents developed in this work is compatible with any other agent that follows the same specifications.

- *Existence of software development tools:* Nowadays, there are many tools to create MAS that provide some facilities (graphical tools, APIs, examples, documentation) to develop and interact with them [Luck et al. 2005, Ricordel & Demazeau 2000]. From all the available tools[2], the most well-known and used are JADE (Java Agent Development Environment, [TILab S.p.A. 2002]), FIPA-OS, Zeus, and AgentTool.

3.2 Adequacy of agent-based systems to healthcare problems

Medical informatics is a promising research area where the agent paradigm can be applied. First of all, we summarise the main characteristics of the problems found in the healthcare domain. Then, we link these characteristics with the use of agents.

Some characteristics of the problems in the medical domain are:

i) It is very usual that the knowledge required to solve a problem is spatially distributed in different locations, which adds several constraints in the planning of coordinated actions.

ii) The solution of a problem involves the coordination of the effort of different individuals with different skills and functions, usually without the supervision of a single centralised coordinator. The provision of health care typically involves a number of individuals (*e.g.*, inpatients, outpatients, physicians, nurses, carers, social workers, managers, receptionists) located in many different places. All these people must coordinate their activities to provide the best possible treatment to the patient.

iii) Health care problems are quite complex, and finding standard software engineering solutions for them is not straightforward. This complexity can be a bottleneck if it is managed with a centralised system.

iv) There is a great amount of medical knowledge available on the Internet that can be accessed and reused.

Nealon & Moreno (2003) argued that agent technology is a good option to be used in healthcare applications. The main reasons they gave in their argument are the following:

[1] All specifications can be downloaded from the FIPA repository: http://www.fipa.org/repository/standardspecs.html [last access: 25/07/2008].

[2] An up-to-date list of agent software tools can be found at the AgentLink website (http://www.agentlink.org/resources/agent-software.php). At the time of writing this chapter (June 2008), the list contains information about 129 tools.

- The components of a multi-agent system may be running in different machines, and be located in many different places. Each of the agents may keep a part of the knowledge required to solve the problem, such as patient records held in different departments within a hospital. Therefore, multi-agent systems offer a natural way of attacking inherently distributed problems.
- One of the main properties of an intelligent agent is sociability. Agents are able to communicate between themselves, using some kind of agent communication language, in order to exchange any kind of information. In that way they can engage in complex dialogues, in which they can negotiate, coordinate their actions and collaborate in the solution of a problem (*e.g.*, different units of a hospital may collaborate in the process of patient scheduling).
- When a problem is too complex to be solved in a single system, it is usual to decompose it in subproblems (which will probably not be totally independent of each other). In multi-agent systems there are techniques of distributed problem solving, in which a group of agents may dynamically discuss how to partition a problem, how to distribute the different subtasks to be solved among them, how to exchange information to solve possible dependences between partial solutions, and how to combine the partial results into the solution of the original problem. Thus, multi-agent systems can handle the complexity of solutions through decomposition, modelling and organising the interrelationships between components.
- Agents can also be used to provide information to doctors and patients. There are information agents (also called Internet agents), that are specialised in retrieving information from different sources, analysing the obtained data, selecting the information in which a user is especially interested, filtering redundant or irrelevant information, and presenting it to the user with an interface adapted to the user's preferences.
- Another important property of agents is their pro-activity; their ability to perform tasks that may be beneficial for the user, even if he has not explicitly requested those tasks to be executed. Using this property they may find relevant information and show it to the user before he has to request it. For instance, if it knows that the user has had heart problems in the past and might need this information urgently, a personal agent that also knows that the user is about to travel abroad could look for information about the medical centres in the towns to be visited that have a cardiology department.
- The basic characteristic of an intelligent agent is its autonomy. Each agent takes its own decisions, based on its internal state and the information that it receives from the environment. Therefore, agents offer an ideal paradigm to implement systems in which each component models the behaviour of a separate entity, that wants to keep its autonomy and independence from the rest of the system (*e.g.*, each unit of the hospital may keep its private data, or each hospital may use a different policy to rank the patients that are waiting for an organ transplant).

Other benefits of agents applied to healthcare are [Fox et al. 2003*b*]: a) agent technology offers advanced platforms for building expert systems to assist individual clinicians in their work, and b) distributed agent systems have the potential to improve the operation of healthcare organisations, where failures of communication and coordination are important sources of error.

3.3 Fields of application within healthcare

Agents have already been used to deal with many different kinds of problems in the healthcare domain. After collecting and analysing the related papers of this area, a list of fields was composed. Only relevant articles published (*e.g.*, in books, special issues of journals and proceedings of conferences and workshops) between 2000 and 2008 were considered.

A first classification of these fields was created by Nealon & Moreno (2003) and Foster et al. (2005), but they have been updated in order to reduce them as much as possible to give a more compact comparison.

3.3.1 Medical data management

The increasing medical information available online (Internet and other electronic sources) has led to the development of information agents that can collect, filter and organise this information. In addition, other systems use MAS to access physically distributed information sources, and build an information system that can be used to extract and combine knowledge or being used as source of other complex systems, as decision support systems.

NeLH (National electronic Library for Health) is a project that offers a portal to retrieve evidence-based medical information on the Internet [Kostkova et al. 2003]. Documents are tagged by professional experts in order to enhance high quality results. The system is composed by a network of autonomous agents that ensure that the available information is up-to-date, and that allow to manage and automate the documents review process and to respond to the user's requests. The NeLH consists of virtual branch libraries, each dedicated to a particular disease or group of diseases.

PALLIASYS was a Spanish research project aimed to improve the management of clinical data of palliative patients combining intelligent agents and IT-based technologies [Moreno et al. 2004, Riaño et al. 2004]. One of the goals to achieve was to improve the process of gathering and collecting the information of the palliative patients. Another important aim was to perform distributed and periodic actions among patients and practitioners (*e.g.*, get information on the current treatment, or change it), and implement an alarm system from the collected data. Fig. 3.1 shows the proposed agent-based architecture. A *palliative care unit* (PCU) includes agents for patients, doctors and the PCU head, and some web-based facilities to allow external accesses. A data analyser agent extracts useful information about the evolution of patients.

MEDUSA (Medical Information System using Multi-Agent Technology) is a system for improving the extensive exchange/use of information in complex organisations [Zachewitz 2004]. The system promotes the integration of heterogeneous clinical systems through wrapping existing systems and allowing communication and transmission issues. With this approach, local data warehouses can be reused.

The MIA (Medical Information Agents) project aims at providing the physician or nurse timely with relevant information during the treatment of (chronic) patients [Braun et al. 2005, Wiesman et al. 2006]. The physician or nurse obtains the information via guidelines, diagnostics or therapy supporting programs and via scientific literature. MIA focuses on the problem of how relevant information from these sources is shown to the physician at the right moment via specialised agents.

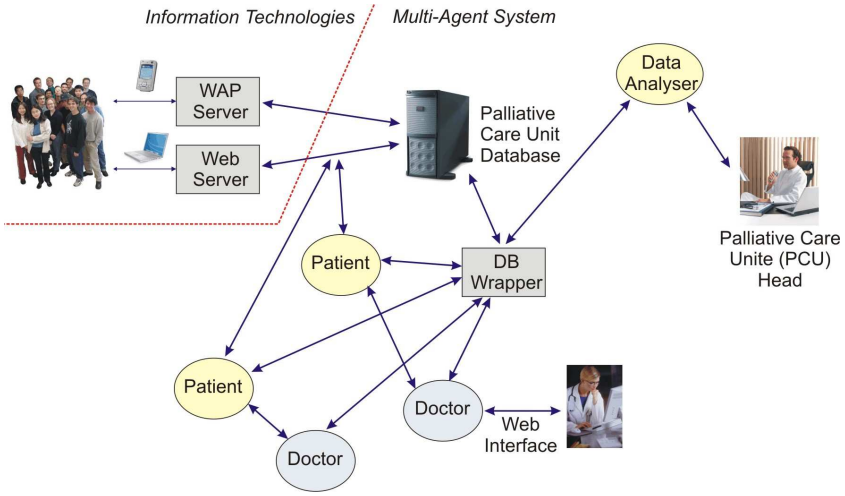


Figure 3.1: Architecture of PALLIASYS [Moreno et al. 2004]

3.3.2 e-Health for elder and disabled citizens

Agent-based systems have been applied in the coordination of all the activities that have to be performed for providing an efficient health care to citizens of a community (especially older or disabled citizens). These people are paid special attention by the European Commission, and they assistance constitutes one of the main topics included in the *e*-Health-related calls for projects.

The INCA (INtelligent Community Alarm) Project describes a distributed system that delivers community care services (referred to services delivered in the person's own home or community setting) focused on older and chronically sick people [Beer & Hill 2006, Hill et al. 2005]. INCA improves the quality of patient's care practices and its management by planning up-to-date individual care plans, monitoring patient's data, detecting emergencies, and providing an appropriate coordination between care providers.

Barrué et al. (2006) and Cortés et al. (2007) propose the integration of multi-agent systems with other existing technologies in order to build specific *e*-Tools for the persons with disabilities and for senior citizens. Fig. 3.2 shows the proposed architecture. The lowest level contains all the physical devices. The second level includes all the required hardware controllers that operate the physical devices. The third level includes the agent-based controllers that receive the information from the controllers, combine it with the knowledge they have about the current state of the system, and infer the next action to be performed (*e.g.*, actions over the controlled devices, or others). Finally, a wireless network provides connection among all layers.

From the same authors of e-Tools, an ongoing project called SHARE-IT (Supported Human Autonomy for Recovery and Enhancement of Cognitive and motor disabilities using Agent Technologies) proposes an evolution of the same idea [Cortés et al. 2008]. SHARE-IT will inform and assist the user and his caregiver through monitoring and mobility help. The internal decision support system will aid the user (typically, elder people with disabilities) by using a network of agents that plan the actions to be followed and supervise their enactment.

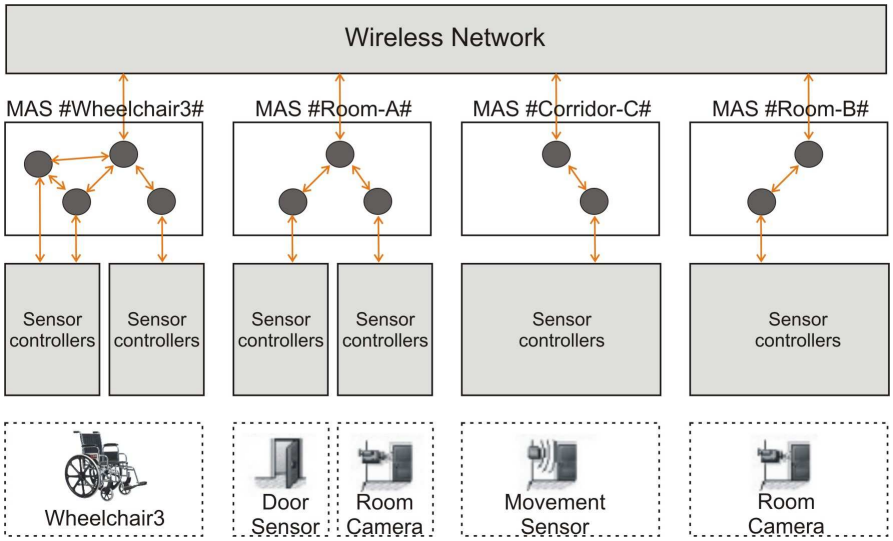


Figure 3.2: Architecture of *e*-Tools research project [Barrué et al. 2006]

In Tablado et al. (2004) an agent-based architecture for monitoring elderly people is explained. In a related area, Corchado et al. (2008) propose the implementation of an agent-based architecture that combines RFID sensors with planning in order to allow to monitor all patients of a residence, plan daily tasks and know the current position of all people.

3.3.3 Decision support systems

A distributed decision support system based on the agent paradigm can monitor the status of a hospitalised patient and help to support co-operative medical decision-making [Lanzola et al. 1999]. This kind of systems deal with different sources of data that should be collected accurately in order to satisfy their objectives.

The HEALTHAGENTS system is the result of a EU-funded research project [González Vélez et al. 2008, Lluch-Ariet et al. 2008]. It is implementing novel pattern recognition discrimination methods, in order to analyse in vivo Magnetic Resonance Spectroscopy (MRS) and ex vivo/in vitro High Resolution Magic Angle Spinning Nuclear Magnetic Resonance (HR-MAS) and DNA micro array data. HEALTHAGENTS intends not only to apply forefront agent technology to the biomedical field, but also to develop a network of agents, with a globally distributed knowledge repository for brain tumour diagnosis and prognosis (see Fig. 3.3). Different centres maintains local classifiers obtained from local patients. When a new case should be diagnosed, these local classifiers are combined to know where that case can be better classified.

Richard et al. (2004) propose the application of multi-agent systems in image interpretation (3D magnetic resonance imaging). Dedicated agent behaviours are dynamically adapted depending on the position in the image where they are assigned, of their topographic relationships and of the available radiometric information. The information collected by the agents is gathered, shared via qualitative maps, or used as soon as available by requests.

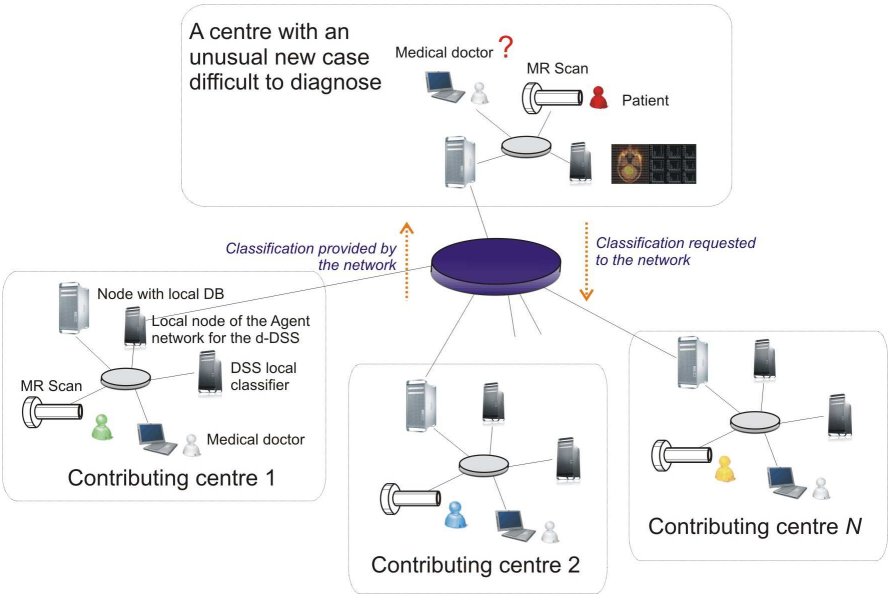


Figure 3.3: Architecture of the Clinical Distributed-DSS for brain tumour diagnosis proposed in the HEALTHAGENTS system [Lluch-Ariet et al. 2008]

Singh et al. (2005) present the architecture of an agent-based healthcare intelligent assistant on a grid environment. This assistant was designed to be used by the medical practitioners to retrieve and use various existing organizational knowledge to help solve medical cases. They use a case-based format that captures the experiential knowledge of healthcare practitioners. Using grid technologies all knowledge about the resources within hospitals can be used by the medical practitioners across a geographically distributed location.

Godó et al. (2003) designed an interesting multi-agent system to help practitioners during the prescription of antibiotics for restricted use, which are a specified set of very expensive antibiotics used only under particular circumstances. The system has a representation of drugs and constraints between them, and attaches an agent to each patient which is responsible of checking different medical aspects related to his prescribed therapy. A pharmacy agent is responsible for analysing it and suggesting alternative antibiotic treatments.

ADAM (Agent Architecture for Diagnostics and Monitoring) is an ongoing research project which aims to build a decision support system to help practitioners to extract information about data collected from measuring devices [Lhotska & Stepankova 2004, Lhotska & Stepankova 2005]. The first task of this project is to build a system that permits to extract features and its values from collected data. To achieve this task, there is a collection of agents that retrieve data, and others that allow to collect them in order to build an ontology-based structure [Lhotska & Prieto 2005]. When this stage has been achieved, post-processing reasoning methods will be added to help practitioners to evaluate the results.

An AI project applied to the healthcare domain is ASPIC (Argumentation Service Platform with Integrated Components) [Fox et al. 2007, Tolchinsky et al. 2008]. The main goals of ASPIC are to develop a theoretical ground for argumentation theory in AI, to develop

practical software components that embody standards for argumentation-based technology (inference, decision-making, dialog and learning), and develop one large scale demonstrator for organ selection and assignment. This model will allow practitioners to learn or trace all decisions made during a normative procedure.

3.3.4 Tele-medicine

Tele-medicine is a type of application of clinical medicine where medical information is transferred via telephone, the Internet or other networks for the purpose of consulting, and sometimes remote medical procedures or examinations. Remote care involves sharing knowledge, data, expertise, and services among healthcare professionals.

A project called TELECARE designed and developed a configurable agent-based framework for virtual communities focused on supporting assistance to elderly people employing tele-supervision and tele-assistance [Camarinha-Matos & Afsarmanesh 2004].

Cervantes et al. (2007) propose the combination of a multi-agent platform with an artificial neural network to create an intelligent decision support system for a group of medical specialists collaborating in the pervasive management of healthcare for chronic patients [Cervantes et al. 2007]. Three different kinds of agents allow to monitor patients remotely. An agent determines the state taking into account the physiological data. The patient agent uses the neural network to classify the symptoms into the specific medical condition. Finally, a doctor agent (in the hospital) receives the data and performs tasks such as sending the medical data to the appropriate specialist.

The SAPHIRE project aims to develop an intelligent healthcare monitoring and decision support system on a platform integrating the wireless medical sensor data with hospital information systems [Laleci et al. 2008]. It is a clinical decision support system for remote monitoring of patients at their homes. The platform uses intelligent agents to manage the available clinical patient's data, and different web services that allow to interact with the local databases.

3.3.5 Planning and resource allocation

Different examples in these areas are available. For instance, an agent-based coordination of tissue and organ transplants management across a hospital ([Vázquez-Salceda & Dignum 2003]) could provide significant improvements in the time required to pull together the resources required for a transplant operation.

OTM (Organ Transplant Management) is a project that improves the current management and use of information in organ transplant [Calisti et al. 2003]. This is done by providing a compact system that integrates patient's data management, match making and decision support procedures. In another related project presented in Moreno et al. (2001), the allocation/coordination of medical resources were managed in order to minimise the global delay. They propose an agent-based architecture that facilitates the coordination and matching of organs to patients. The general performance of the search (an important requirement in this domain) is also improved. Recently, the EU funded project PROVENANCE developed a distributed organ transplant management system [Vázquez-Salceda et al. 2008]. It allows to coordinate tasks among different healthcare institutions, and also traces all performed actions in order to identify problems or audit all the steps followed in the care process. This

project is based on a previous work called CARREL, an Agent-Mediated Electronic Institution for the distribution of organs and tissues for transplantation purposes [Vázquez-Salceda et al. 2003]. CARREL was the aim to help speeding up the allocation process of solid organs for transplantation to improve graft survival rates. Fig. 3.4 shows the organization of medical partners included in this system. There are i) the hospitals that create the tissue requests (represented through a Transplant Coordination Unit Agency (UCTx)), ii) the tissue banks, and iii) the national organ transplantation organizations, that own the agent platform and act as observers (in the figure the organizations in Spain are depicted: the Organización Nacional de Transplantes (ONT) and the Organització Catalana de Transplantaments (OCATT)). In the proposed system all hospitals, even those running a tissue bank, must make their requests through CARREL in order to distribute and to track the pieces from extraction to transplant.

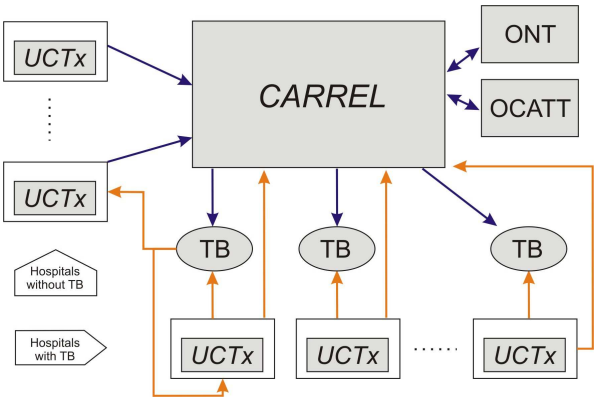


Figure 3.4: CARREL: an Agent-Mediated Institution for tissues assignment [Vázquez-Salceda et al. 2003]

Another interesting project in this field is MeSSyCo [Ciampolini et al. 2004]. MeSSyCo is a multi-agent system that integrates and coordinates heterogeneous medical services like diagnosis, clinical laboratory analysis and data retrieving, in order to improve specific services with accurate algorithms. For instance, service agents use a combination of probabilistic reasoning and abduction.

Another project in resource allocation is detailed by Becker & Krempels (2003), which implements a planner and scheduler of operation theatres. AGENT.HOSPITAL ([Becker et al. 2003, Kirn et al. 2003]), is an open agent-based framework for highly distributed applications in health care and provides different interfaces to integrate existing information systems. The framework contains numerous healthcare actors and consists of detailed partial models of the healthcare domain. It enables the examination of modelling methods, configuration problems, as well as agent-based negotiation strategies and coordination algorithms.

3.3.6 Education and simulation

Agents can help to improve medical training and education in distance-learning tutoring systems [Hospers et al. 2003]. Moreover, simulation tools can help to evaluate the performance of complex behaviours or patterns [Amigoni et al. 2003, Beda et al. 2004]. A good example of a complete system is the ANTHROPIC AGENCY. It is a biomedical control system for

the modelling and regulation of complex physiological phenomena. The authors propose an architecture based on three groups of agents that perform three basic activities: knowledge extraction, decision making, and plan generation (see Fig. 3.5). In addition, they show the implementation in two case studies related to the control of the glucose-metabolism processes in diabetic patients ([Amigoni et al. 2003]), and in the control of an adaptive cardiac pacing ([Beda et al. 2004]).

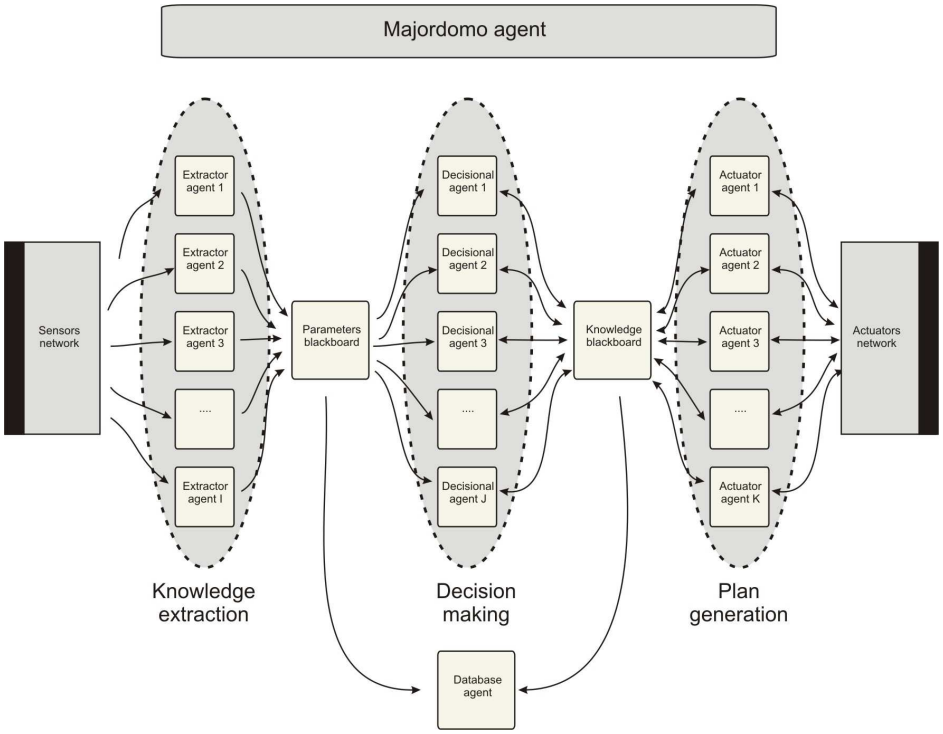


Figure 3.5: The general anthropic agency architecture [Amigoni et al. 2003]

The paper by Anderson et al. (2007) reports the construction of an agent-based model used to study humanitarian assistance policies executed by governments and non-government organizations for the health and safety of refugees. An experimental design was used to study the relationships among five factors: basic needs, food and water, sanitation, medical resources and security. The simulation demonstrates the critical role of security in providing for the health and well-being of refugees. A major strength of the model is that it allows policy makers to incorporate specific characteristics of the refugees and of the governmental and non governmental organizations that are providing humanitarian aid to the camp.

There are some simulation applications in the biology domain. Bortolussi et al. (2005) present a multi-agent based framework to simulate a protein folding process. The model identifies every biological entity (aminoacid) with a concurrent agent, which interacts with other agents, aimed at coordinating the activity of the basic processes and inducing some basic form of cooperation.

Herrler & Puppe (2005) explain the design and implementation of a framework to develop

an hospital simulator. The kit scenario, implemented using SeSam (ShEll for Simulated Agent SysteMs[3]), includes information about patients and practitioners, a time model for the simulation, clinical pathways followed by the care unit, and information about treatments and laboratory examinations. These data are used to extract the global behaviour of a department or the whole hospital, and allows preventing errors in daily care and optimising the existing medical resources.

The last project is an *e*-learning- and web-based system implemented for helping in medical teaching [Alves et al. 2006]. The system integrates different information sources into a common format. Agents, in this case, adapt the content to the user's interests, and collect and filter pro-actively the related medical documents.

The authors of simulations (as in the case of the last example) should report the accuracy of these experiments. As Amigoni & Schiaffonati (2007) report, a drawback of this kind of agent-based applications is the difficulty to evaluate its accuracy and validate the results.

[3]For more information, please visit http://www.simsesam.de/ [last visit 30/06/2008].

3.4 Conclusions

Multi-agent systems are powerful and flexible tools for modelling and regulating complex phenomena. In fact, a way to manage the complexity of a phenomenon is to decompose it in such a way that each agent embeds the control model for a portion of the phenomenon. In this perspective, the cooperative interaction among the agents results in the controller for the whole phenomenon. Since the portions in which the phenomenon is decomposed may overlap, the actions the single agents undertake to regulate these portions may conflict; hence a balanced negotiation is required.

Through the chapter, several applications in the healthcare domain that use multi-agent systems have been sketched. The agents are used in the healthcare domain because of the basic features they provide such as sociability, pro-activity and autonomy. All these properties are crucial for healthcare solutions because it is hard to apply straightforward software engineering approaches to these problems.

Some concluding remarks of the state-of-the-art in the application of agents in the healthcare domain are the following:

- Systems are mainly focused on improving decision quality rather than speed as they are either time insensitive or mid-term urgencies.
- Only a few of the systems use distributed data, but if they have this property they can do distributed data mining.
- Most of the examined systems implement a supervised system where the user gives instructions to the system and then the system gathers the required information, responds to the user and waits for further commands.
- Only some of the systems are intended to be included in complex decision support systems in which they have to operate in real circumstances or with real data.

Chapter 4

Methodological development of the MAS

Nowadays, one of the drawbacks of most implemented MAS is the lack of application of a rigorous methodology during the analysis and design phases. The use of an agent-oriented software engineering (AOSE) methodology in these preliminary stages of a MAS development provides some advantages (*e.g.*, the possibility of reusing and sharing a common vocabulary and pieces of code in an easy way), and improves the quality of the software [Bernon et al. 2005, Cernuzzi et al. 2004, Gómez-Sanz et al. 2004, Henderson-Sellers & Giorgini 2005].

One of the main problems to adopt a methodology in daily work is the wide range of available possibilities. Depending on the requirements of the project, several approaches can be applied, and the criteria for selecting one or another are difficult to evaluate. This chapter describes the whole procedure followed in order to apply a methodology to design a MAS in the healthcare domain. First of all, the selection of a methodology, and then, the construction of a prototype. The former was tackled defining clearly the requirements to be accomplished and evaluating different alternatives. The availability of a CASE tool (to ease learning and allow collaboration during the edition), and the use of a well-known agent-oriented programming language were the main criteria to perform the selection. The latter consisted in the implementation following the rules defined by the methodology and adapting our requirements to the particular representation used. In most of the cases, methodologies define an internal representation of elements with particular relationships and constraints, which can allow us (or not) to model a particular requirement in an appropriate way. The other shortcoming that can hamper the use of a methodology is the learning process. In our case, the selected approach defines a particular vocabulary adapted from UML and other software engineering techniques, and a sequence of steps and models that require the analysis of examples and tutorials. Technical support given by authors and the availability of examples have helped us during the whole development process.

4.1 Methodology selection process

As shown in the state-of-the-art of methodologies to develop MAS (see Appendix A and [Gómez-Alonso et al. 2007]), several approaches to design a MAS are available. The criteria to select one or another are very subjective because any of them covers all the requirements of the system be implemented.

A subset with the most *well-known* alternatives was considered attending to their usability, the existence of CASE tools to facilitate the implementation, the coverage of the whole life-cycle (analysis, design, and implementation), the agent-oriented programming language used during the codification, the level of formalisation and abstraction of the created documents, the expressiveness of agents and groups of them (organizations), or the availability of documentation to ease learning. The approaches that could be used in a real implementation are: EXTENDED GAIA [Zambonelli et al. 2003], INGENIAS [Pavón et al. 2005], MASE [DeLoach 2004], PASSI [Cossentino 2005], PROMETHEUS [Padgham & Winikoff 2004], and TROPOS [Bresciani et al. 2004].

After analysing all of them, EXTENDED GAIA was discarded because it only covers analysis and design, and the generated documents are difficult to translate into a programming language and to validate formally. The rest of tools offer a wide range of features and supportive elements, and it was difficult (and somewhat subjective) to select one of them. All the methodologies allow modelling agents (with formal representations), defining different roles of these agents, teams or groups of agents (organizations), and detailing communication issues (conversations and protocols). They also offer full life-cycle coverage (analysis, design and development), and provide some CASE tools. At the end, INGENIAS [Pavón et al. 2005] was selected.

The most prominent features of this methodology, which have led to its selection in this work, are:

- It allows an agent developer to analyse and design a wide range of MAS.
- It permits to describe agents in different levels of abstraction, including social and organizational issues, goals and tasks.
- It is possible to detail the environment where the agents act.
- It allows the definition of conversations and messages to be exchanged.
- It allows the definition of an agent's mental state, which includes information about facts, beliefs, events and goals, and their changes depending on the agent's current role.
- It includes several examples that ease the learning of this methodology. Moreover, several research projects have adopted INGENIAS [Cuesta et al. 2005, Soto et al. 2006].
- It provides a CASE tool called Ingenias Development Kit (IDK), which facilitates the implementation of projects as well as its learning.
- It also allows to translate the designed models into an agent-oriented programming language.

4.2 The Ingenias methodology

INGENIAS is a methodology to guide the development process of a MAS from analysis to implementation, developed at Universidad Complutense of Madrid (Spain) [Gómez-Sanz 2002, Gómez-Sanz 2003, Pavón et al. 2005, Pavón et al. 2006] (See Appendix A.3.2.2).

The first stage is gathering the requirements with interviews with the final users of the system, and interpreting the problem to be solved. INGENIAS proposes different models in the phases of analysis and design (detailed in Table 4.1 in the next section). Although INGENIAS executes its activities within the RUP, it may be necessary to iterate over different phases until the system is finalized. The process ends when there is a functional system that satisfies user requirements. These are identified with use cases at the beginning.

4.2.1 Meta-model

As shown in Fig. 4.1, the meta-model proposed in INGENIAS is quite detailed. It was defined using the GOPPRR language (Graph-Object-Property-Port-Role-Relationship) [Kelly et al. 1996, Pohjonen 2005].

The GOPPRR elements may be described as follows:

- An *object* is a thing that exists on its own. Examples of objects are a Button, a State, and an Action. All instances of objects support reuse functionality: an existing object can be reused in other graphs by using the add existing function.
- A *relationship* is an explicit connection between a group of objects. Relationships attach to objects via roles. An example of a relationship is a Transition.
- A *role* specifies how an object participates in a relationship. Examples for a Transition relationship are the roles *From* and *To*, which specify how the objects at either end of the Transition participate in the relationship.
- A *port* is an optional specification of a specific part of an object to which a role can connect. Normally, roles connect directly to objects, and the semantics of the connection is provided by the role type. Ports are defined for an object type, and all instances share those same ports.
- A *graph* is a collection of objects, relationships, roles, and bindings of these to show which objects a relationship connects via which roles. A graph also maintains information about which graphs its elements explode to.
- A *property* is a describing or qualifying characteristic associated with the other types, such as a name, an identifier or a description.

Using the GOPPRR representation, INGENIAS proposes five different concepts (see Fig. 4.1):

- *Organization*: framework where agents, resources, tasks and goals coexist. It is defined by its structure, functionality and social relationships.

  An *Organization* is an *Autonomous Entity*, which pursues a *Goal*, and can be structured in *Groups* (structural entities), and contains *Workflows* (dynamics of the organization

processes). A *Group* may consist of *Roles*, *Agents*, *Resources*, and *Applications*. Workflows define precedence relationships among Tasks, the Resources assigned to Tasks and their participants (in terms of Roles).

- *Agent*: program that follows a rational behaviour and processes knowledge. The definition of an agent requires a purpose, its responsibilities and its capabilities. The behaviour consists of its mental state (information that allows the agent to take decisions), its Mental State Manager (operations over elements of its mental state and their relationships) and its Mental State Processor (selection of a decision to execute a task).

  An *Agent* is also an *Autonomous Entity*, which plays some *Roles* and pursues *Goals*. It has a *Mental State*, which allows the agent to plan and infer facts from incoming events from the environment (*e.g.*, if a pending result of a patient is arrived, the doctor can supervise these data).

- *Tasks/Goals*: describe relationships among goals and tasks, goal structures, and task structures. They are also used to express which are the inputs and outputs of the tasks and what are their effects on the environment (using resources) or on an agent's mental state.

- *Interaction*: exchange of information or requests between agents (or agents and human users). Its elements are the actors, the interaction specification, the scope and the nature of the interaction.

  In INGENIAS, an Interaction is initiated by an *Agent*, with some *Goal* (intention). Several agents can participate in an interaction. Several formalisms can be used to describe an interaction, such as UML collaboration diagrams and Grasia! diagrams (more specific diagrams designed by the INGENIAS authors - See Appendix B -) [Pavón et al. 2005].

- *Environment*: defines an agent's perception in terms of existing elements of the system. It also identifies system resources with which the MAS interacts (resources and/or other agents), and who is responsible of their management.

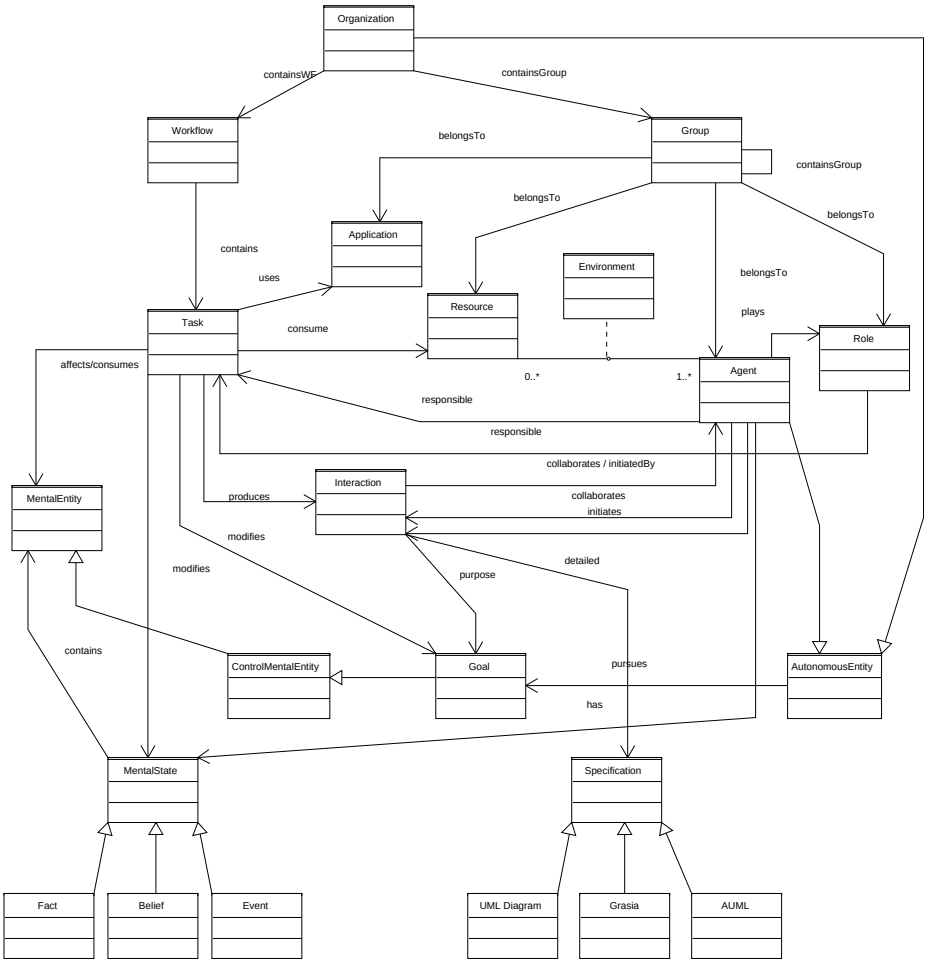


Figure 4.1: Meta-model defined in INGENIAS (adapted from [Bernon et al. 2005] and [Pavón et al. 2005])

4.2.2 Ingenias Development Kit (IDK)

INGENIAS Development Kit (IDK)[1] is the official tool developed by INGENIAS authors to allow a rapid implementation of applications as well as the verification of all designed models (see Fig. 4.2). The IDK is divided into two main parts: the *editor* and the *code generator*. The former includes a general view of the project, a graphical editor of all models, and a tree-based inspector of all individual entities. The latter is a module that facilitates the translation of all designed elements into an agent-oriented programming language.

The use of this CASE tool simplifies the complex procedure proposed by INGENIAS because it guaranties the validity of all designed diagrams, and it allows a clear comprehension and revision of all features that the system should implement at the end of the analysis and design process. Unfortunately, the used version of INGENIAS (release 2.6) has still several drawbacks or bugs that hinder the implementation, such as edition errors, loading exceptions, or the intrinsic complexity of models. The authors of IDK are currently working on a new release of this framework in order to fix reported bugs. This continuous support allow developers to reuse existing projects and improve them with new features.

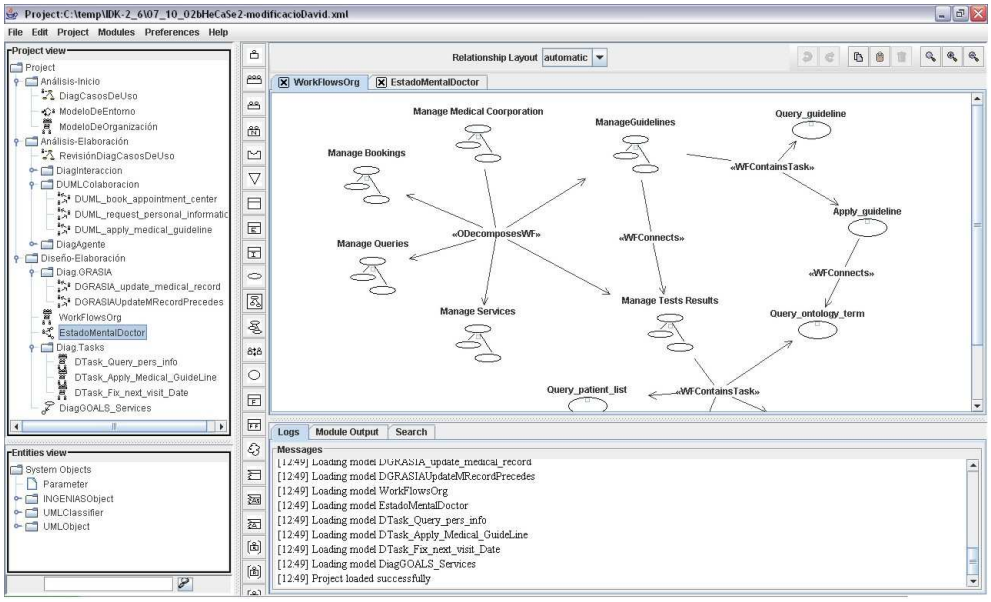


Figure 4.2: Screenshot of project created using INGENIAS Development Kit (IDK)

[1] Free to download at http://ingenias.sourceforge.net [last visit: 08/01/2008]

4.3 Applying INGENIAS to HECASE2

INGENIAS covers the *analysis*, *design* and *implementation* stages defined in the RUP lifecycle. The methodology is divided in six steps (three in the analysis and three more in the design) defining five different models (see Table 4.1). The generation of code is made when all models are completed. The IDK tool allows an agent designer to create the models described in the INGENIAS methdology. In this case, the models are created during the analysis and design stages. It also allows to generate a complete documentation in HTML reporting all data about the entities, actors, models and relationships, and also translating all models into the JADE agent-oriented programming language [Bellifemine et al. 2007].

Table 4.1: Phases and models defined in INGENIAS [Pavón et al. 2005]

REQUIREMENTS ANALYSIS	- Interpretation of all features and functionalities to be deployed. Main entities and relationships should be described accurately, as well as general organization behaviours (if they are present)		
ANALYSIS	**Inception**	**Elaboration**	**Construction**
	- Generate ***use cases*** and identify their actors - Sketch a system architecture with an ***organization model*** - Generate ***environment models*** to represent result from requirement gathering stage	- Refined use cases and interactions associated to them - Preliminary ***Agents model*** that detail elements of the system architecture - ***Workflows and tasks*** in organization models - ***Models of tasks and goals*** to highlight control constraints (main goals, goal decomposition) - Refinements of environment model to include new environment elements	- Refinements on existing models to cover use cases
DESIGN	- Generate prototypes perhaps with rapid application development tool such as ZEUS or Agent Tool or code generators from the IDK. Sometimes this prototyping limits to concrete aspects and uses a more conventional technology	- Refinements in workflows - ***Interaction models*** that show how tasks are executed - ***Models of tasks and goals*** that reflect dependencies and needs identified in workflows and how system goals are achieved - ***Agent models*** to show required mental state patterns	- Generate new models - Social relationships that regulate organizational behavior
IMPLEMENTATION	- Translation of all models of previous stages and generation of basic agents using an agent-oriented programming language		

4.3.1 Analysis of requirements

The first task in the development of a complex system is to analyse and summarise its main requirements:

The system should model a generic user (or patient) that wants to interact with a complex medical organization. The medical organization is inspired in the Catalan Health Service (CatSalut). From the user point of view, there is no direct communication with medical practitioners, and a representative of the organization is required. A broker allows the user to obtain a filtered collection of results from different medical centres. A medical organization includes medical services and doctors organized into departments. These medical services can be located in a particular department or shared by the whole medical centre. Some basic functionalities, such as booking a medical visit or searching a particular department, are offered to the user through the broker and the medical centre. An internal service used by doctors is the management of clinical guidelines. They are retrieved and executed by doctors, and can require the supervision of users to confirm pending activities. Internal tasks coded inside the clinical guidelines require coordination among doctors and medical services in order to obtain a result. In addition, the medical knowledge used by doctors is represented in a medical ontology. Moreover, an electronic health record stores all results provided by services (tests results) or by doctors (results of medical visits).

Details about the particular communication protocols used between the agents, and the definition of the ontology shared inside the multi-agent system cannot be represented accurately in this methodology. At the end of the design stage, a set of basic agent classes are generated and used to structure the internal code of the MAS.

4.3.2 Analysis

The agent developer studies the problem domain and analyses the main entities and the relationships with the environment and between them. After these evaluations, the agent designer creates the *Use Cases Diagram,* and the *Organization* and *Environment* models.

4.3.2.1 Inception

First of all, the *inception* phase is addressed to perform a deep study of the system to be developed in order to identify the involved actors and their tasks. The agent designer begins with the comprehension of the requirements in order to create a *Use Cases Diagram*, which shows the feasibility of the system. Two more diagrams (*Organization* and *Environment* models) should be sketched.

Use Cases Diagram

The following use cases were identified (see Fig. 4.3).

- *Book an appointment with a doctor of a particular department of a medical centre*. Appointment between a user (patient) and a doctor of a medical centre. The department and the centre should be provided by the user before performing the action.
- *Book an appointment with the doctor*. Appointment between a patient and a specific doctor. During a medical visit the doctor can request a new appointment with the patient. A confirmation of the proposal from the patient is required.
- *Request personal information*. A patient requests his related data stored in the electronic medical record.
- *Request medical information.* A user (patient) can obtain generic information about medical centres.
- *Request patient list*. A doctor requests the list of arranged appointments with patients. It is an internal method that does not require to access to any resource but is an step performed periodically by all doctors.
- *Apply medical guideline*. A doctor applies a guideline to a patient. This is the more complex feature and involves different partners to achieve it. The doctor must contact other colleagues or medical services that provide some results and findings. The collection of values define an evolution of a patient through the clinical guideline. Doctor and patient know exactly the current state of the treatment.
- *Update medical record*. A doctor updates the personal medical record of a patient after the medical visit.
- *Request a service*. A doctor books a required service for a patient. This required medical service can be located in the same department or in another one or can be a common medical service shared among all departments.
- *Carry out a service*. After a medical service has been completed, the patient's medical record is updated with the results.

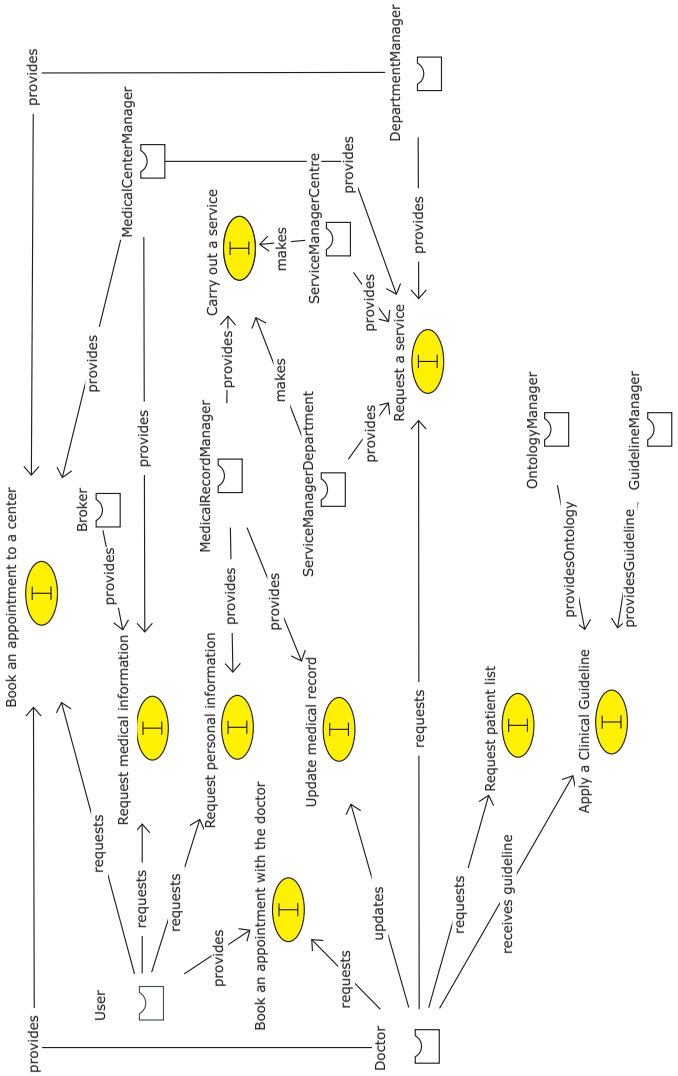


Figure 4.3: Initial *Use Cases Diagram*

Organization Model

The organizational model is the most complex and interesting model of this stage and describes how system components (agents, roles, resources, and applications) are grouped together, which tasks are executed in common, which goals they share, and what constraints exist in the interaction among agents. These constraints are expressed in the form of subordination and client-server relationships.

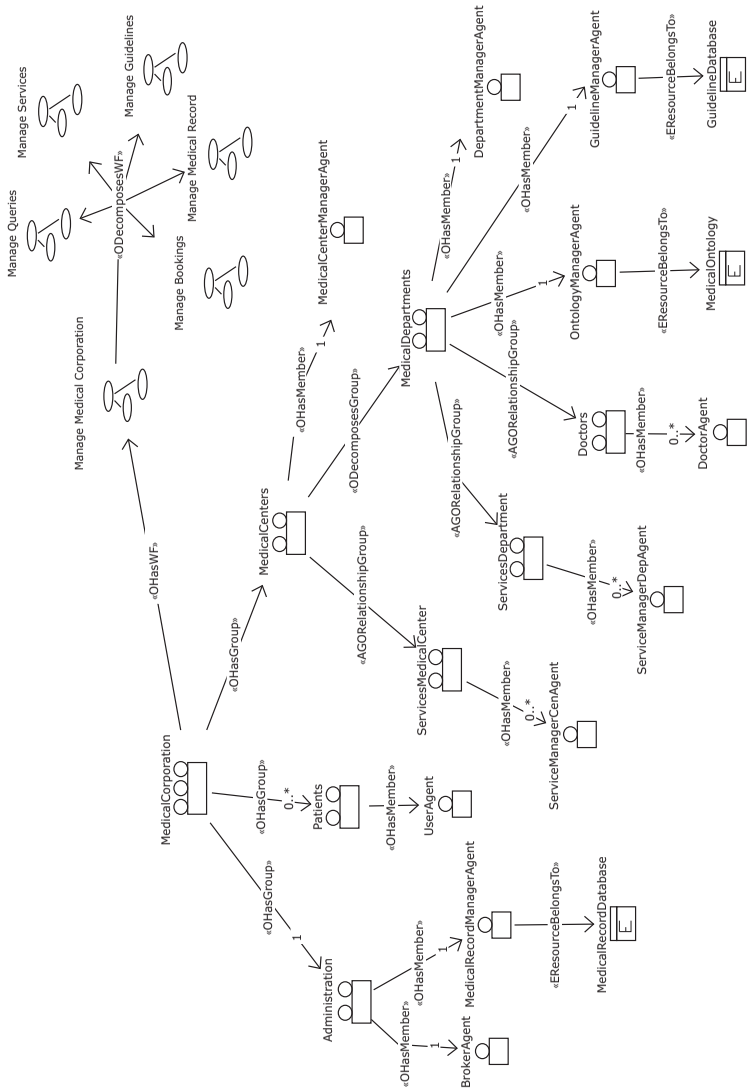


Figure 4.4: The *Organization Model*

As shown in Fig. 4.4, the model represents how the agents are grouped together and share the goals. The medical organization is structured into medical actors, administrative actors and patients. A deep analysis of a medical centre shows a hierarchical organization divided in departments. Each one includes doctors, a guideline manager and an ontology manager. In the case of medical services, they can be found at the level of the medical centre or in the department. The manager of the medical record and the broker are classified as administrative actors. Moreover, the medical organization has different workflows to manage the services, the guidelines, or the electronic medical record, which are identified in this stage. In addition, the agent designer should identify the main tasks and goals to be accomplished into the workflows and assign them to the actors which shoould pursue them (particular agent or an organization).

Environment Model

The *Environment Model* defines the agent's perception in terms of existing elements of the system (see Fig. 4.5). This model includes information about resources (consumable or non consumable), other agents, and applications (internal or external to the system). The main goal of this model is not the definition of these resources, but to detail the relationships between all elements.

In the case of study, the system is linked with three databases: one that represents the medical record accessed by the medical record agent, another to store the clinical guidelines accessed by the guideline agent, and another to save the medical ontology accessed by the ontology agent.

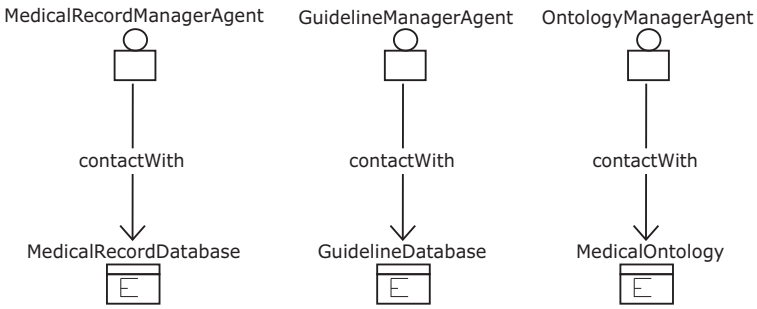


Figure 4.5: The *Environment Model*

4.3.2.2 Elaboration

During the *elaboration* phase of the analysis, the internal descriptions of the agents as well as their tasks and goals are documented in the *Agents Model* and also in the *Task and Goals Models*. This stage includes a refinement of the use cases by grouping the common features and assigning interaction models to those cases. In addition, the workflow decomposition of the organization should be reported.

Refinement of the Use Cases Diagram

During the elaboration phase, the *Use Cases Diagram* created previously (Fig. 4.3) should be refined with the information about the interactions between agents. In this case study (see Fig. 4.6), most of the use cases require an interaction between agents in order to perform complex tasks.

In addition, the diagram clusters all cases with common goals (*e.g.*, bookings, queries or guideline's management).

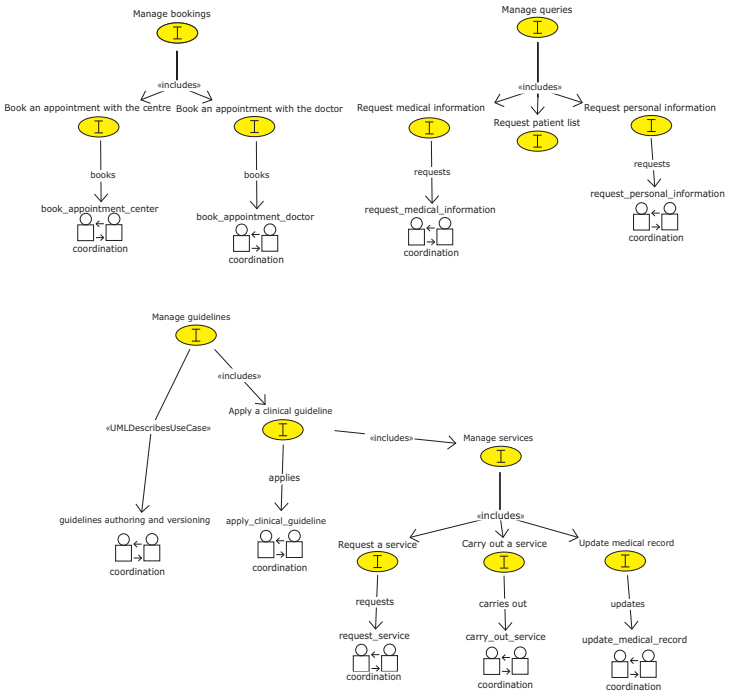
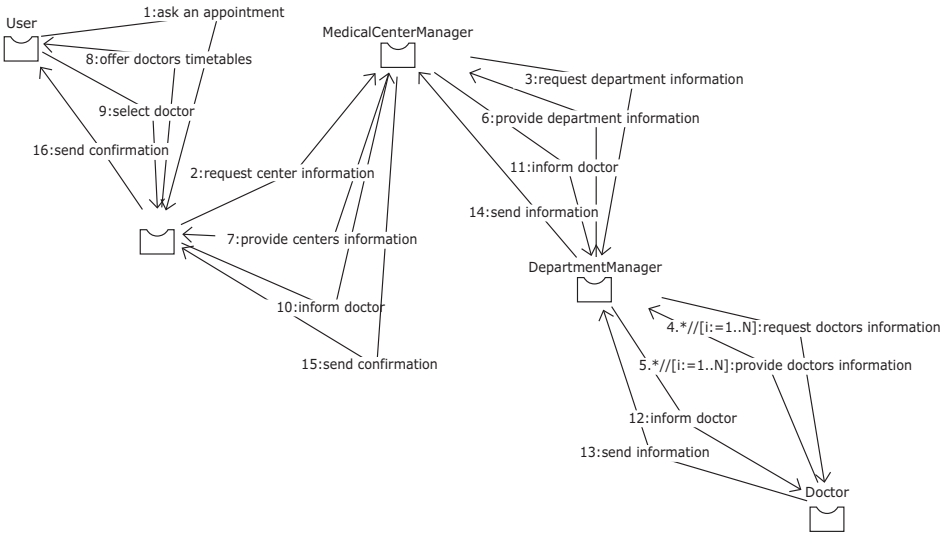


Figure 4.6: Refinement of the *Use Cases Diagram*

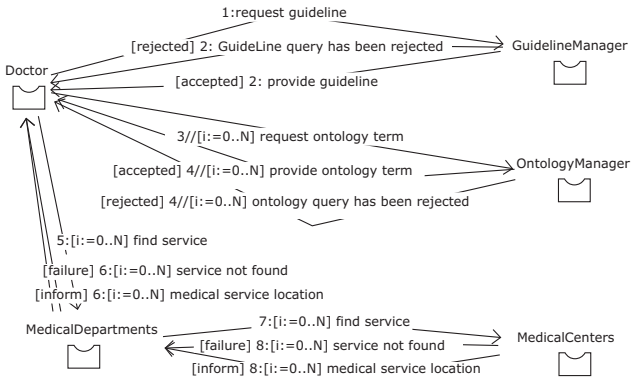
Collaboration Diagrams in UML

For all interactions defined in the previous model, it is necessary to make a preliminary specification using Collaboration Models in UML. These diagrams present the sequence of messages exchanged by all partners of the interaction. FIPA-IEEE protocols are recommended to be used but the CASE tool does not provide any library of those predefined protocols such as FIPA-Request, FIPA-Query, or FIPA-Contract Net [FIPA 2002*c*].

In the following figure, two examples of these diagrams are presented. The first one (Fig. 4.7(a)) shows the collaboration diagram for the *Book an appointment with a doctor of a particular department of a medical centre*, which involves five different roles. The second example (Fig. 4.7(b)) shows the application of a clinical guideline between three different entities: a doctor, the guideline manager and the ontology manager.



(a) Specification of *Book an appointment with a doctor of a particular department of a medical centre*



(b) Specification of *Apply a guideline*

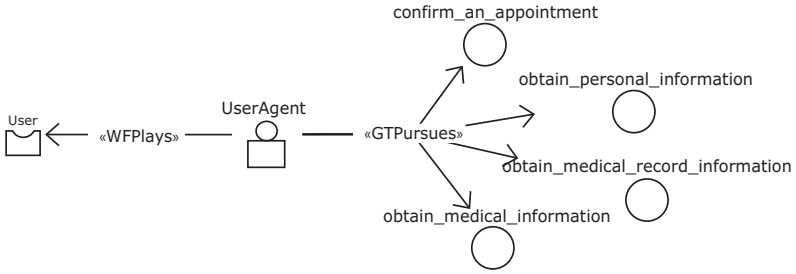
Figure 4.7: Examples of *Collaboration Diagrams in UML*

Agents Model

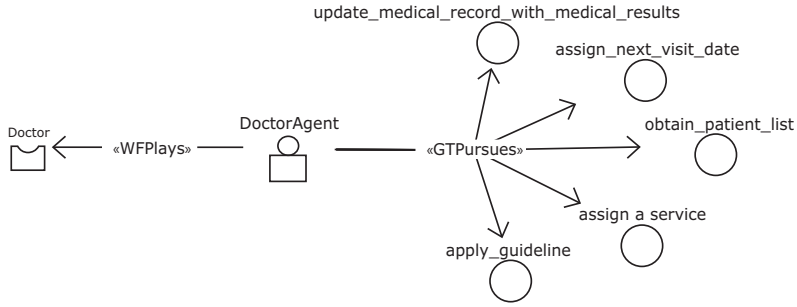
The *Agent Model* describes single agents, their tasks, goals, initial mental state, and played roles. Moreover, *Agent Models* are used to describe intermediate states of agents. These states are presented using goals, facts, tasks, or any other system entity that helps in its state description. In this way, an agent model can represent in what state be an agent that starts a negotiation should be.

At this point, we can outline the main goals of each kind of agent:

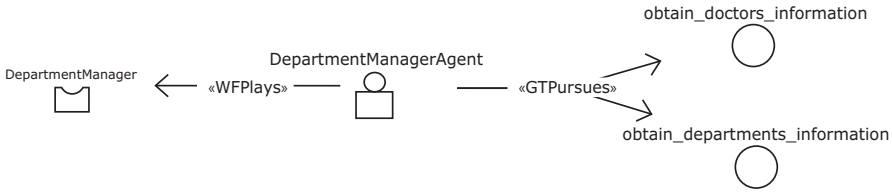
- *User*: get an appointment, obtain Personal information, obtain Medical Record information, obtain Medical information (see Fig. 4.8(a)).
- *Doctor*: include patient's data to the EHR, obtain patient list, assign a service, apply Guideline, arrange next visit date, search medical services (see Fig. 4.8(b)).
- *Broker*: obtain Medical Centers information.
- *Medical Center Manager*: Obtain Departments information, obtain Medical Center Services information.
- *Department Manager*: obtain Doctors information, obtain Department Services information (See Fig 4.8(c)).
- *Service Medical Center Manager*: execute services assigned to the medical centre.
- *Service Department Manager*: monitor staff and all assigned services within a department.
- *Guideline Manager*: obtain a guideline.
- *Ontology Manager*: obtain semantic meanings of medical terms from the medical ontology.
- *Medical Record Manager*: update the EHR (from services or doctors), obtain Medical Record.



(a) User agent



(b) Doctor agent



(c) Department agent

Figure 4.8: Examples of *Agent Model*

Goals Model

The *Goals Model* expresses the constraints and dependencies between goals, and the agent model links each goal to the agent who is able to perform it. After designing these two models, the *Tasks Model* defines all tasks in terms of goals, preconditions, postconditions, required resources, and required external modules.

The goals' data are collected from the *Agents Model* (see Fig. 4.8) during the analysis phase, and complemented with information from the *Interaction Model* (see Fig. 4.12) during the design phase. The former explains what goals are, and the latter explains how those goals are achieved.

Fig. 4.9 shows the whole *Goals Model*. One of the goals with more dependencies is referred to the execution of clinical guidelines which includes getting information about the agents who are able to provide required services, requesting services and, at the end, performing the service and adding the result to the electronic medical record. At the same time, these goals include more goals (or sub goals) like assigning a service and obtaining the description of a medical centre. Dependencies between tasks are expressed using the workflow diagrams. In this case, before applying a guideline, the actor that performs this task (doctor agent) should request it, and during the enactment, the information related to the medical terms should also be requested. Workflow and goals models together describe how an agent acts.

Tasks Model

The *Tasks Model* describes the goals, preconditions, postconditions, required resources, and required external software modules, for all designed tasks.

For instance, the application of a guideline requires to obtain a guideline, can require the results of tests, requires a coordination with other agents (doctors and services) (see Fig. 4.10(a)). As a result of the application of a guideline, an entry to the health record is made with the results. Another example of these models is given in Fig. 4.10(b) describing how a medical visit is arranged.

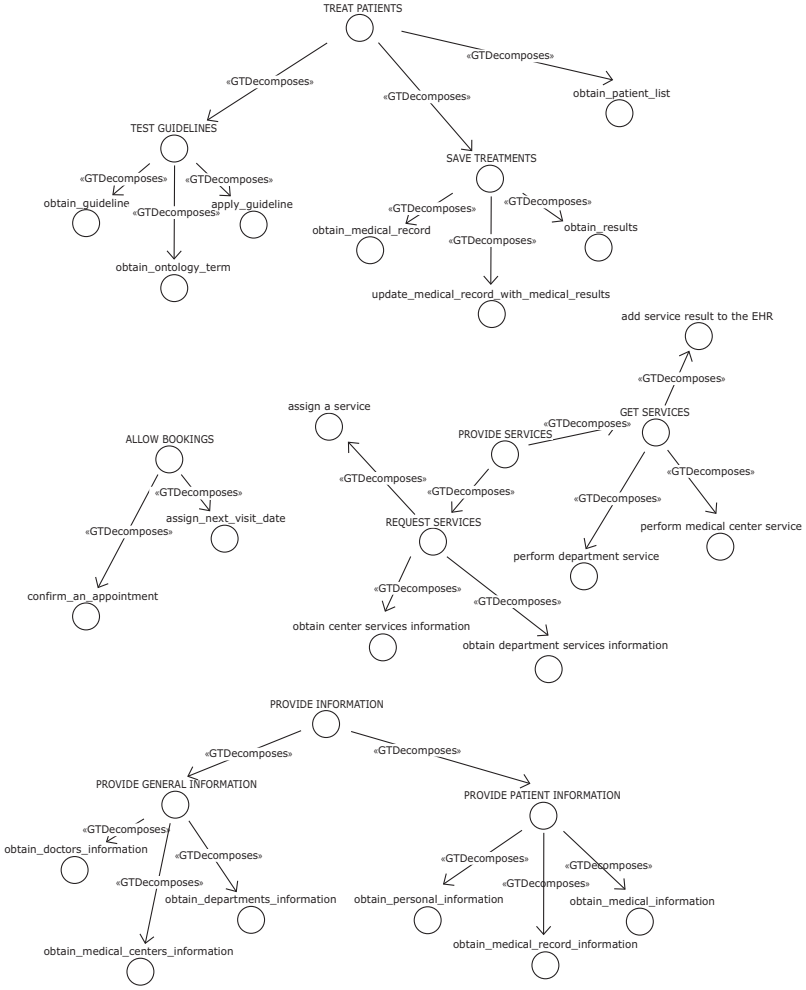
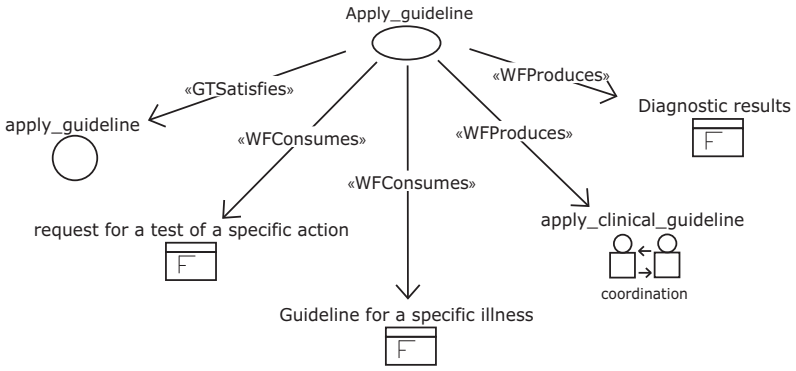
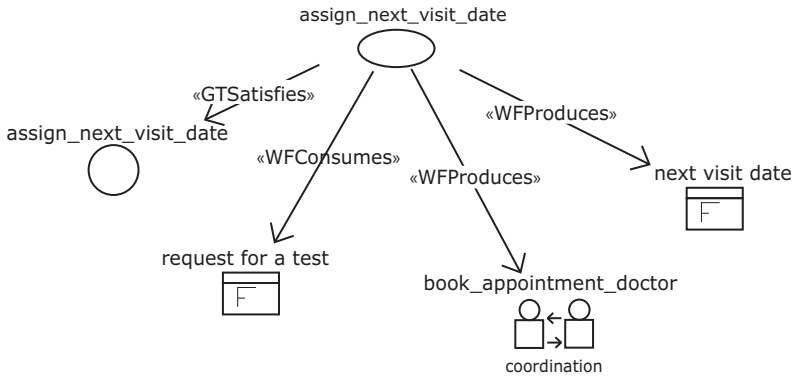


Figure 4.9: The *Goals Model*



(a) Task *Apply a guideline*



(b) Task *Arrange next visit date*

Figure 4.10: Examples of *Tasks Models*

Decomposition of workflows

In order to facilitate both the management and definition of tasks, INGENIAS incorporates a high level of abstraction called *workflow*, where tasks are included.

In our case study, the following workflows were defined (see Fig. 4.11):

- *Manage bookings*, which includes the tasks *book an appointment* and *arrange next visit date*.
- *Manage queries*, which includes the tasks *query personal information*, *request medical record information*, *request medical information*, *request medical centres information*, *request department information*, *request doctors information*, and *query medical record*.
- *Manage services*, which includes the tasks *assign a service*, *make medical centre service*, *make department service*, *add service result to medical record*, *request medical centres services information*, and *request departments services information*.
- *Manage guidelines*, that includes all tasks related with the management and execution of clinical guidelines, such as *apply guideline*, *query guideline*, *query ontology term*, and *update medical record with results*.
- *Manage medical record* which includes the tasks related to the update or selection of data from the medical record. Doctors and services can get and put data into the medical record, and both cases are distinguished in the workflow. It is related with the *manage guidelines* workflow.
- *Manage medical corporation* is a general workflow that includes the management of the relationships between all entities of a particular healthcare organization. It includes the maintenance up-to-date of all status of all contained entities (departments, services, and doctors).

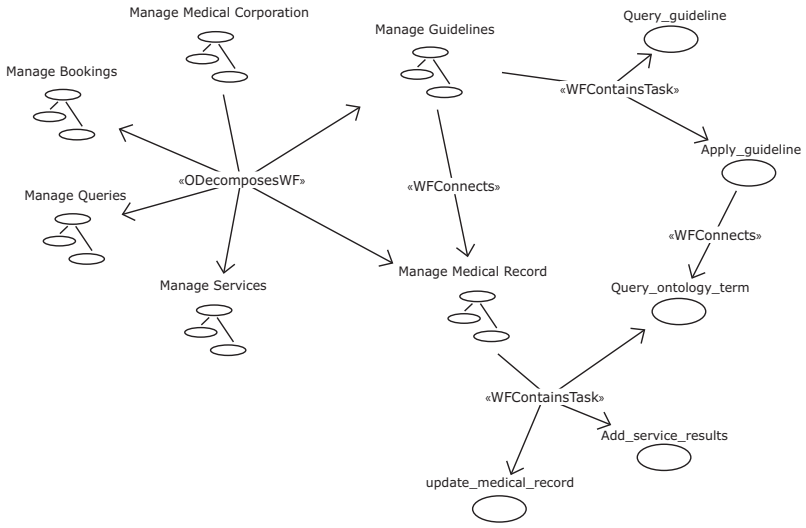


Figure 4.11: Classification and decomposition of tasks into workflows

4.3.3 Design

This stage refines the models identified during the analysis stage. These new models should not change the global view of the system (*e.g.*, addition of a module to report the doctor's work, communication between the guideline managers in order to share common guidelines). In the design phase, those new cases could require new dependencies that should be specified.

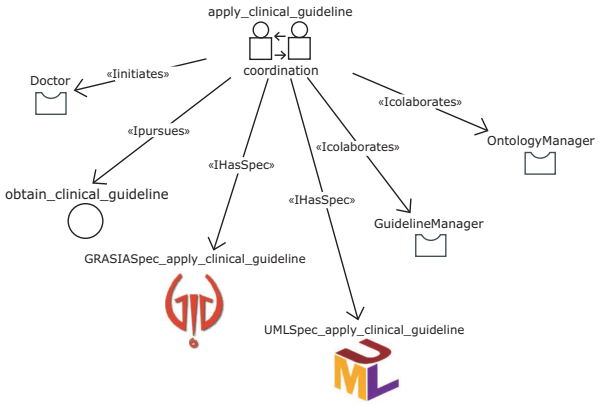
Elaboration

Basically, the results of this stage are the following: models of tasks and goals, decomposition of workflows, and mental states of the agents. This stage includes a refinement of the *Use Cases Diagram* by grouping the common features and assigning *Interaction Models* to those cases. These interaction models are defined using *UML Collaboration Diagrams*.

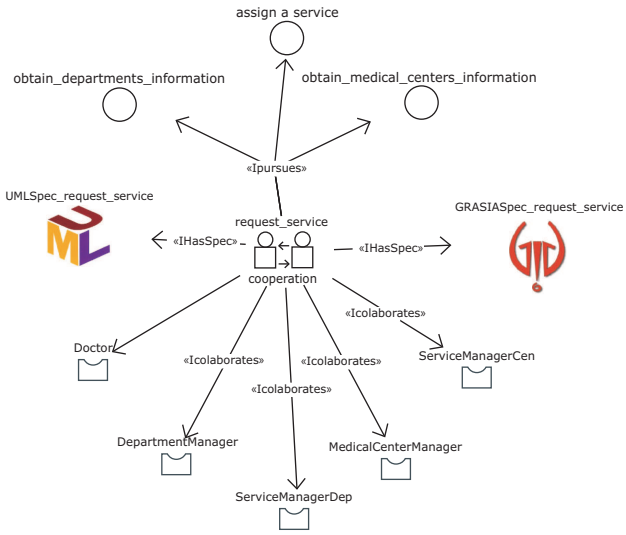
Interactions Models

An *Interactions Model* describes a communication between two or more agents. As shown in Fig. 4.6, several queries, requests, and the management of services require exchanging data between different kinds of agents.

In the following figure, two examples of these models are presented: *apply a guideline* and *request a service*. The former (Fig. 4.12(a)) involves a doctor and the guideline and ontology managers. The latter (Fig. 4.12(b)) includes all agents contained in a medical centre.



(a) *Apply a clinical guideline*



(b) *Request a service*

Figure 4.12: Examples of *Interactions Models*

The interactions defined to cover the previously identified goals are:

- *Book an appointment with a medical centre*: negotiate an appointment, obtain medical centres information, obtain departments information, and obtain information about

doctors.

- *Book an appointment with the doctor*: confirm the date of the next visit.
- *Request medical information*: obtain medical information.
- *Request personal information*: obtain personal information, obtain medical record information (user), and obtain medical record information (medical record manager).
- *Apply clinical guideline*: apply guideline (doctor), obtain medical record (guideline manager), and obtain ontology term (ontology manager).
- *Update medical record*: include medical information (doctor), obtain the medical record (medical record manager), update the medical record with the results of a service or a medical visit (medical record manager).
- *Request a service*: assign a service, obtain department services information, and obtain medical centre services information.
- *Carry out a service*: perform a service in a department, perform a service in a medical centre, and add the results to the medical record.

Specifications of the interactions model

The lowest degree of specification of interactions is represented using Grasia! diagrams. These UML-based diagrams express the exchanged messages, input/output of a shared element, remote invocations of methods, or a reference to another interaction. In addition, tasks involved in a interaction are also shown.

The diagrams for the task *update medical record*, performed by a doctor after a medical visit with a patient, are analysed in more detail. First of all, Fig. 4.13 shows the Grasia! diagram for that interaction that involves a doctor and the medical record manager. The order of the exchanged messages is described using a precedence diagram depicted in Fig. 4.14.

Mental states

A *mental state* is a representation that is able to build a plan (through a state-transition graph) for executing tasks in order to accomplish a goal. The *mental state* responds to events from the environment and infers something. For instance, Fig. 4.15 shows how a doctor can infer that if a result of a pending test of one of his patients arrives, he can obtain the current state of this patient and, after observing these results, he can include an evaluation in his electronic patient record. The set of mental states and their embedded conditions are monitored by the *mental states processor*.

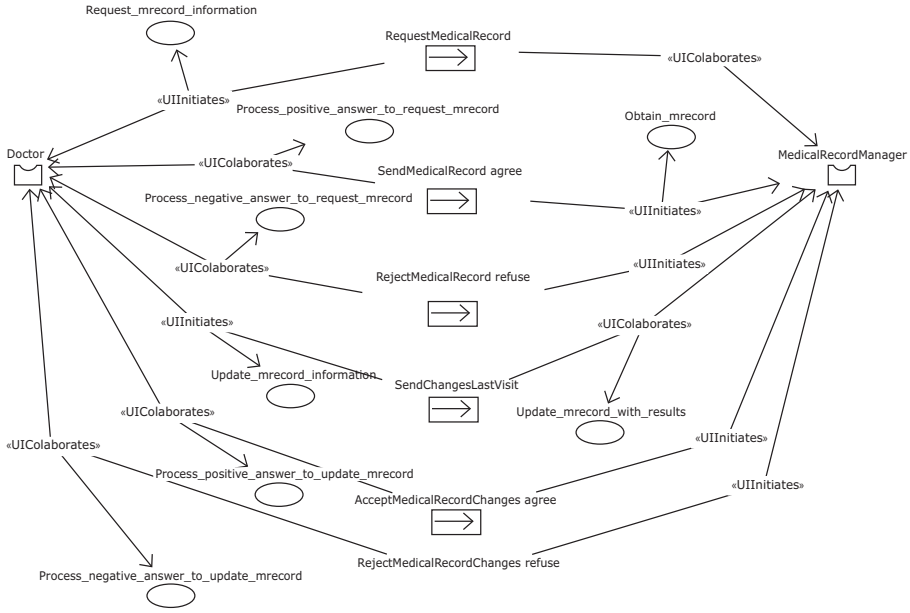


Figure 4.13: Grasia! specification of the task *update medical record*

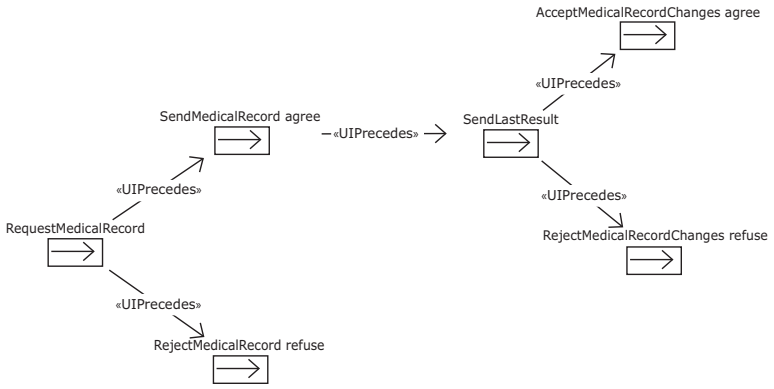


Figure 4.14: Precedence graph of entities in the task *update medical record*

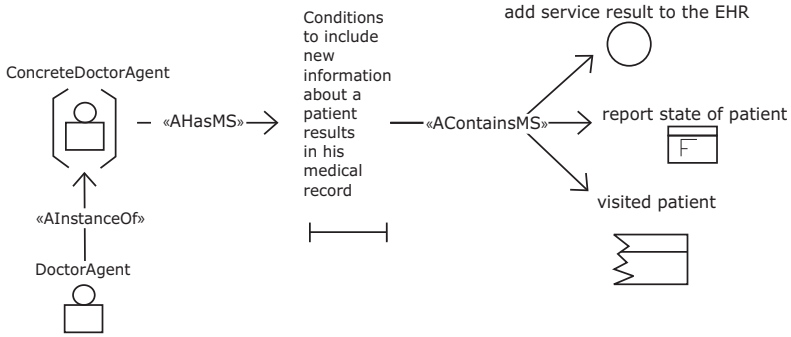


Figure 4.15: Example of mental state

4.3.4 Implementation

The generation of source code is one of the most interesting features of INGENIAS. The IDK has two modules to generate a complete documentation in HTML as well as a prototype using the agent-oriented programming language JADE [Bellifemine et al. 2007]. Even though the internal codifications used during the analysis and design stages are independent of the final language, JADE is a good option to translate all specifications. Agents include several behaviours (internal threads) that handle all goals to accomplish. If the goal includes different tasks and dependencies between them, INGENIAS authors set up a finite state machine behaviour, where nodes are goals, and transitions conditions are achieved through conversations. Particular communication protocols like FIPA-Query and FIPA-Request are allowed in INGENIAS and are translated directly to the corresponding initiator and responder classes. In addition, INGENIAS allows to include external applications and use them through calling the corresponding API methods, and embed Java pieces of code if a particular method has to be specified.

4.4 Conclusions

Following the adoption of a methodology to develop a MAS, a prototype using an agent-oriented language was implemented. This created prototype contains a general skeleton for all agents with the conversations patterns, functional features applied to a particular organization (topology of agents), internal roles of agents according to a set of defined goals, sequences of general tasks included into workflows, and the translation of all interaction models into communication protocols.

The prototype does not cover the full functionalities of HECASE2, and other features such as the definition of an ontology in the communications between agents or security issues were added *a posteriori* [Isern & Moreno 2008*a*]. However, the generated skeleton is very useful to organize a complex MAS, and changes in the organization or in the agents' roles can be implemented more easily than in an *ad hoc* implementation. Concretely, the identification of bottlenecks or reusing pieces of software are easy with this approach and, in addition, the generated models are used to document the functionalities of the final software in terms of agents, communication protocols, tasks and goals.

From the designed skeleton, the complete multi-agent architecture of HECASE2 is presented in Fig. 4.16. This is an open architecture and, depending on the situation, there will be more or less agents of each type, and more or less interaction between them. At the top, the patients are represented by *User Agents* (UA). All UAs can talk with the *Broker Agent* (BA). The BA is the bridge between users and the medical centres, and it is used to discover information. The BA knows about the medical centres located in a city or in an area. A *Medical Centre Agent* (MCA) monitors all of its departments, represented by *Department Agents* (DAs), and a set of general services (represented by *Service Agents* (SAs)). Each department is formed by several doctors (represented by *Doctor Agents* (DRA)) and more specific services (also modelled as SAs). Moreover, in each department there is a *Guideline Agent* (GA) that performs all actions related to guidelines, such as looking for a desired CG, storing and/or changing a CG made by a doctor, etc. This GA contains only CGs related to the department where it is located but, if it is necessary to use another guideline (*e.g.*, when treating a patient with multiple pathologies), the GA could request it from other GAs.

Each department also contains an *Ontology Agent* (OA) that provides access to the designed medical ontology and complements the information provided by the GA. At the bottom of the architecture there is the *Medical Record Agent* (MRA) which controls the access to a DB that stores all patient health records (PHR) in the medical centre. This agent provides a secure transmission of sensitive data.

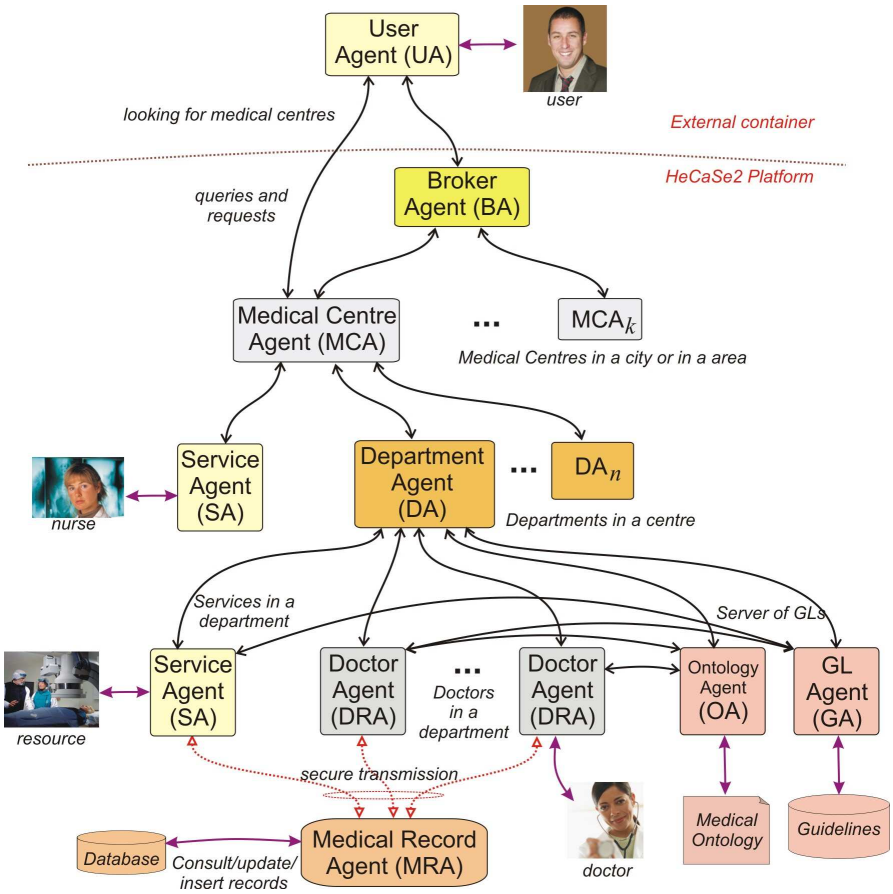


Figure 4.16: Complete architecture of HECASE2

Chapter 5

Provision of personalised medical services

The new information technologies allow people to access efficiently a wide range of data and services, improving health care delivery [Singer et al. 2001]. These initiatives make it necessary to collect information that was not needed in the past. This information can be used to improve continuity care (track patients in different care settings across the health system), to inform people about enrollees before and during medical care, and to enable the assessment of new technologies. In general, the technology that predated the 1990s was proprietary to particular institutions and therefore incompatible with technology in other institutions. This situation makes difficult not only the transfer of information across institutions but also the modernisation of existing technology. Converting or replacing existing systems requires substantial investment. The proposal described in this manuscript changes substantially this point-of-view and goes an step forward towards the openness and interoperability of existing information systems, enabling the citizen as an active actor in the health care delivery. Concretely, this chapter describes a distributed system that provides different information-based services (*e.g.*, it looks for health care providers), looks up the citizen's personal medical record, and negotiate and recommend medical bookings with a doctor taking into account the personal preferences of the user.

Health care is becoming increasingly patient-centred and individualised, with the patient becoming an active subject rather than a mere object of health care. This kind of systems store user preferences to deliver the personalised services [EC 2007, MHP 2006]. Patient-centred health care means that the system should be designed and delivered to address the needs and preferences of patients so that health care is appropriate and cost-effective [IAPO 2006]. Even though these approaches should be implemented with high level governmental policies, there are different issues where the inclusion of information about patients preferences' can improve the current management of the services delivered to citizens (*e.g.*, adoption of personal treatments, filtering medical data during the diagnosis or treatment).

This chapter is mainly divided into two parts: provision of information-based health care services to citizens (Section 5.1), and personalisation of those services (Section 5.2). The former explains the agent-based platform that implements information-based services to citizens. The personalisation of medical services includes the representation of a user's profile,

the exploitation of user's preferences to make recommendations and, the implicit adaptation of the user's profile over time.

5.1 Home Care Services

One of the goals of this dissertation is to provide a general and distributed health care model based on intelligent agents. The model has different actors such as practitioners, nurses and patients. The basis of that model was proposed under the EU-funded project AgentCities [Willmott et al. 2001], which described a user as a citizen (with his own personal interests) that is moving abroad and accesses some information-based services that cover an area (with one or more cities) through a mobile device (traditionally a PDA or a mobile phone). With these basic rules, several European research groups designed services like restaurants information retrieval and recommendation [López et al. 2007], tourist information services [Bajo, Botti, Corchado, Ilarramendi, Ilarri, Julián, Carmona, Marsá, Mena, Moreno, Pavón & Valls 2007], and e-Government [Palau et al. 2004].

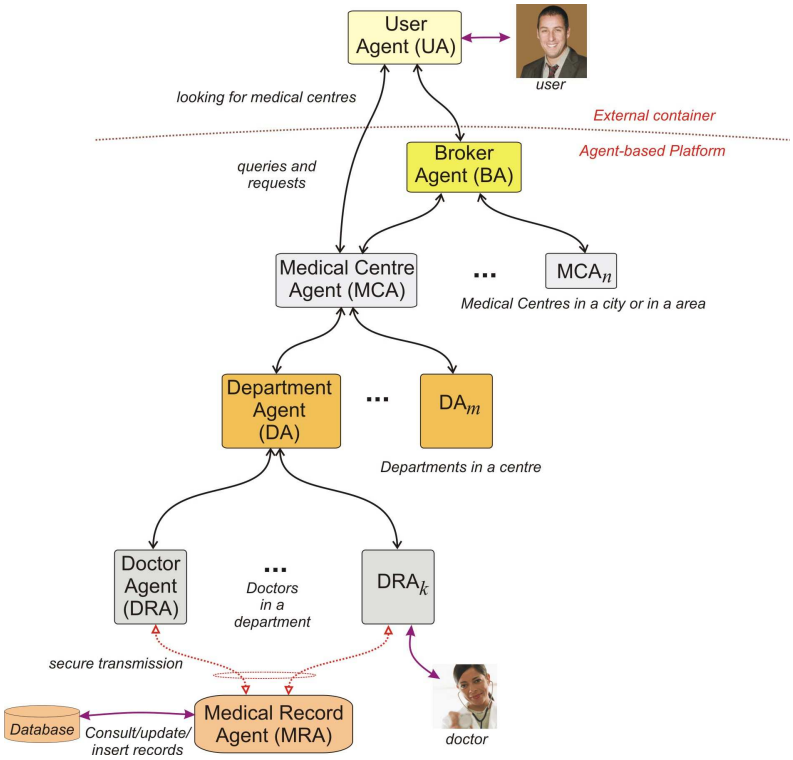


Figure 5.1: Architecture of the multi-agent system HECASE [Moreno, Isern & Sánchez 2003, Moreno, Valls, Isern & Sánchez 2003]

In this project, the HECASE system was presented in [Moreno, Isern & Sánchez 2003, Moreno, Valls, Isern & Sánchez 2003, Isern & Moreno 2004*b*, Isern & Moreno 2004*a*, Isern et al. 2003*b*]. Its main goals were:

- The main aim of our work is to develop a set of agents that coordinate and communicate to offer to the citizens and visitors of a city not the usual leisure-oriented services but health-care related services.
- The user of HECASE may request information about all the medical centres available in a particular geographical area.
- The internal structure of the medical centres modelled in the system is analogous to the structure of medical centres in Catalonia where each centre has a set of departments, and each department has a set of doctors.
- It should be possible to book a visit to be examined by a doctor.
- The user must be given access to his medical record. A doctor should also be able to consult and update the medical record of a patient during a visit.
- It must assured that nobody can access the private medical information of the users of the system without proper authorisation.

5.1.1 Searching the appropriate medical centre

The *user agent* is an interface agent that allows a citizen to interact with the system (see Fig. 5.2). One of the basic services delivered by the platform is the search for medical centres that satisfy some requirements.

As shown in Fig. 5.2, in this request the user has to select the broker agent that is aware of a set of medical centre agents. This agent will consider the specified user's constraints to perform the search.

The requirements that the user can specify are:

a) *Name*. The user may provide a string that must appear somewhere in the name of the medical centre (*e.g.*, "Hospital", "central", "St. John", "memorial").

b) *Centre type*. The user may search for a particular kind of medical centre. The application differentiates between these kinds of centres: "Primary Assistance", "Clinic", "Consultory" and "Hospital".

c) *Department*. The user may detail that he is interested in centres having a particular department (*e.g.*, "cardiology", "general medicine").

d) *Origin city*. The user may be interested only in those centres that are located in a particular town.

e) *Destination city*. In this option, the user may provide his current location to the system. This information is used to sort the results according to the distance from the user's location.

All constraints may be freely combined by the user, and the broker will filter and/or sort all received results. With this approach, the user can request from the simplest query "*give me the details of all hospitals*" to more complex ones like "*give me the details of all hospitals that contain a Cardiology department, and sorted from my current location in Tarragona*".

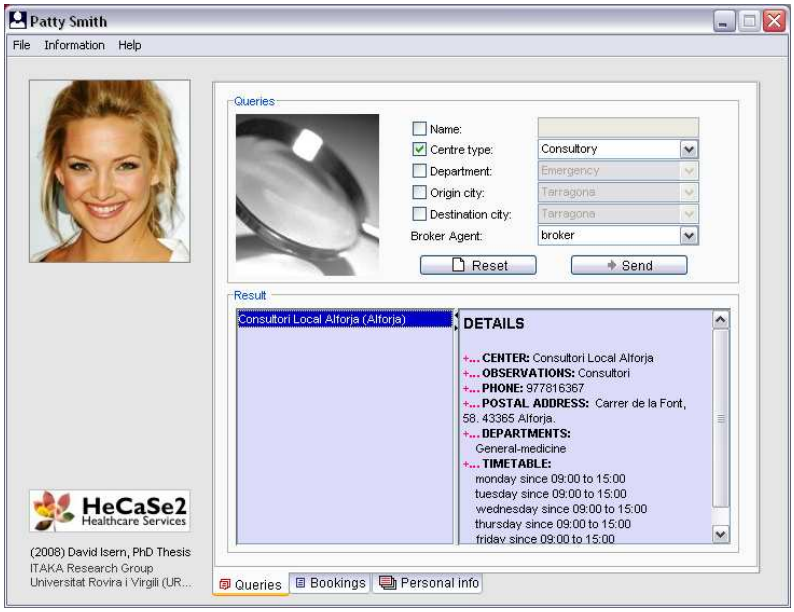


Figure 5.2: Screenshot of the user agent searching for an appropriate medical centre

Fig. 5.3 depicts the protocols implemented in this search process. The procedure is a particular case of the FIPA Brokering Interaction Protocol [FIPA 2002*b*]. The use of brokerage agents can significantly simplify the interaction between agents in a multi-agent system. Additionally, brokering agents also enable a system to be adaptable and robust in dynamic situations, supporting scalability and security control at the brokering agent. In our case, the initiator and the broker implement a FIPA Query Protocol [FIPA 2002*e*], and the broker also uses this protocol with the final receivers of this brokering service.

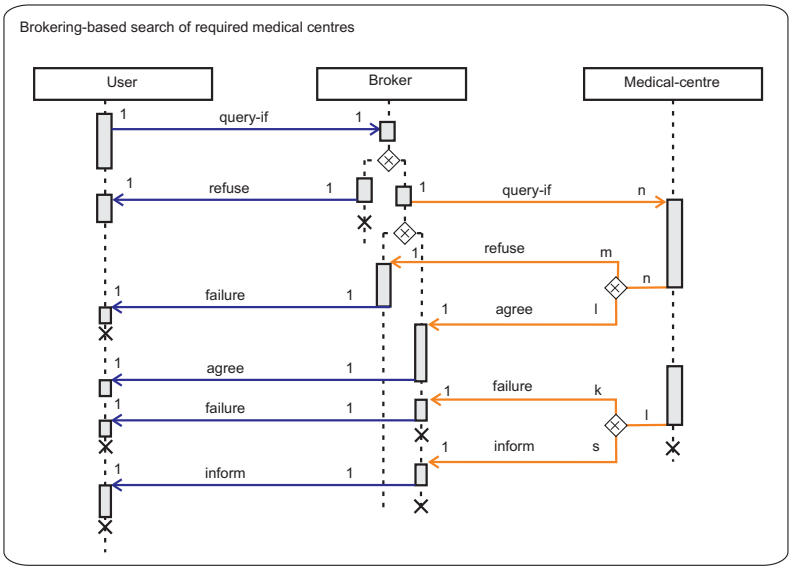


Figure 5.3: AUML diagram of the brokering-based search of medical centres

5.1.2 Access to the Electronic Health Record

Another service offered to the user the possibility of accessing his personal medical record in a secure fashion [Moreno, Sánchez & Isern 2003].

All data related to patients is transmitted through a *medical record agent*. This agent is a wrapper of the database with the personal data and results collected from the patients of the covered area. Fig. 5.4 shows the agent-based protocols able to interact with this agent. The first stage, implemented with the FIPA-Query protocol, aims to query the data related to a patient; this task is only allowed for doctors or the own patient. The second task is intended to update the medical record by adding a new entry (*e.g.*, result of a medical visit, result of a medical test); this case is implemented with the FIPA-Request protocol.

All agents that are able to communicate with this agent use ciphering techniques to protect all data and authenticate the users in order to prevent unauthorised accesses[1]. In addition, integrity[2] and non-repudiation[3] facilities were also provided.

The implementation was made combining different facilities such as the JADE-S plug-in, SSL transmissions, and ciphering techniques.

JADE-S is an add-on of JADE that allows to restrict the allowed actions for a certain type of agents. Thus, it is possible to assign permissions to parts of the code and to its executers, restricting the access to certain methods, classes or libraries depending on who wants to use them. An entity can only perform an action (send a message, move to another container) if the Java security manager allows it. The set of permissions associated to each identity

[1] *Confidentiality* is the property that ensures that only those that are properly authorised may access the information.

[2] *Integrity* is the property that ensures that information cannot be altered. This modification could be an insertion, deletion or replacement of data.

[3] *Non-repudiation* is the property that prevents some of the parts to negate a previous commitment or action.

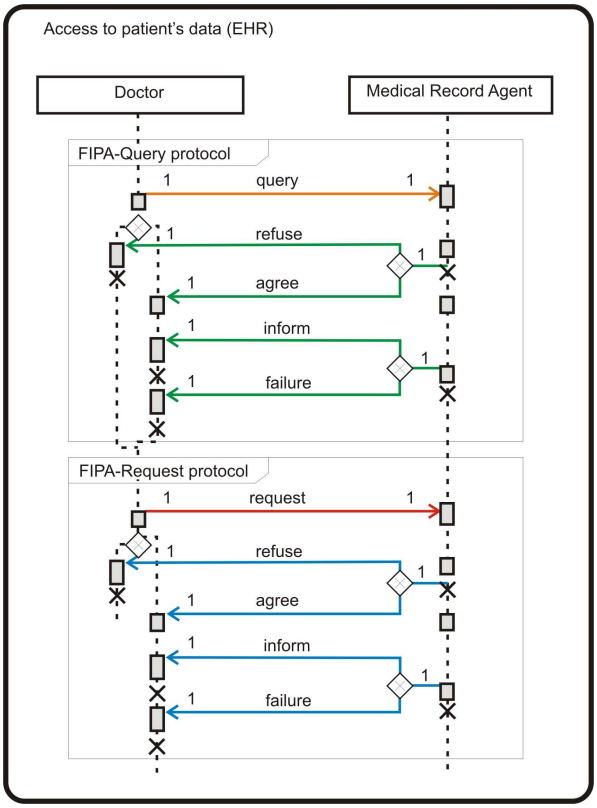


Figure 5.4: AUML diagrams to access the EHR through the medical record agent

is stored in the access rights file of the platform (which is unique and is loaded when the platform is booted). To assess the authentication, JADE-S provides a Certification Authority used to sign the certificates of all the elements of the platform. To do that, it owns a couple of public/private keys so that, for each certificate, it creates an associated signature by ciphering it with its private key (which is secret). Then, when the identity of an entity has to be checked, the signature may be unencrypted with the public key of the Authority (which is publicly known) and we can check that the identity that the entity wanted to prove matches the one provided by the Authority. The secure platform JADE-S provides a Certification Authority within the main container. Each signed certificate is only valid within the platform in which it has been signed.

The permissions associated to the different agents in the platform are the following:

- All the internal agents (agents that are executed within the main container of the platform such as the *broker*, *medical centres*, *departments*, *doctors* and the *medical record agent*) do not have any constraint on the actions they can perform. This decision was made to facilitate the interaction among them (they can send and receive messages from any agent, or register in the DF or the AMS). As they are internal to the platform (they execute in the main container) and they have been programmed by us, they will

not perform any malicious action (kill other agents, deregister other agents, fake the identity of another agent, etc.).

- The external agents (they can be executed in an external container in another host such as the *user agents*) can neither access the main container (to join the other agents) nor access the DF (to modify the information of other agents). They can only communicate with the internal agents through the broker; therefore, they cannot pretend to have the identity of other agents or kill other agents. All these constraints are necessary because we do not have any control over these agents, and they may have been programmed to perform dangerous actions.

In order to provide a secure communication between agents located in different hosts or containers, the SSL protocol (Secure Socket Layer) was used to provide privacy and integrity for all the connections established in the platform. This is a way of being protected against network sniffers. Even though it is only necessary to encrypt those messages between the main container and the external container (between external agents and the broker) that contain confidential information (medical records), the activation of SSL may only be made globally: we can only cipher all messages or none of them. Thus, by activating the appropriate option in the JADE initial configuration files in the client (user-side) and in the server (platform), all the communications between the agents of the system are ciphered and, therefore, we can safely send the encryption keys or the medical records. This mechanism works if we have previously obtained an identity certificate for the server side (the one that boots the platform), so that SSL may implement authentication. This transparent service prevents from sniffing but it remains to assure that only the owner of a patient's record is authorised to access it, and neither SSL nor JADE-S provide information to perform this comparison. Basically, it is not possible to access the information about the identity of the agent from the program.

Finally, at the lowest level of security mechanisms, an authentication mechanism was added. Classical authentication is based on public key algorithms (in our case RSA), so that each user owns a public key and a private one (which is only known by the agent itself and by the certification authority that generated it). In our case, the keys (managed internally by the *medical record agent*) are associated to the user from the personal data when he joins the system the first time. These keys allow on agent to sign the messages that it sends, so that its identity may be checked (from the *user agent* to the *medical record agent*, and vice versa).

When the user joins the system, the medical record agent generates the keys and sends them through the *broker*. This agent stores the public key of each registered user and sends both keys to the user agent. It is not necessary to control in the program the identity of the agent that sends the message (*medical record agent* or *broker*) with identity certificates, because it is controlled at a lower level (using SSL and JADE-S users management). When the user wants to send a critical message in which it has to prove its identity (*e.g.*, when it wants to access the medical record or request a visit to a doctor), it encrypts the message with its (secret) private key. When the broker receives this message, it can check the identity of the sender by using the public key associated to this agent. If the unencrypted content is valid, the identity is deemed correct and the request is processed. If the key does not match the one of the agent or the content of the message has been modified, the result of the unencryption will be wrong and will provoke an exception during the interpretation process, which will cause a denial of the request (see Fig. 5.5).

The last secure-based facility offered in this system is the non-repudiation. It is easy to audit all the transactions made through the platform due to all the security measures implemented in different transmissions levels.

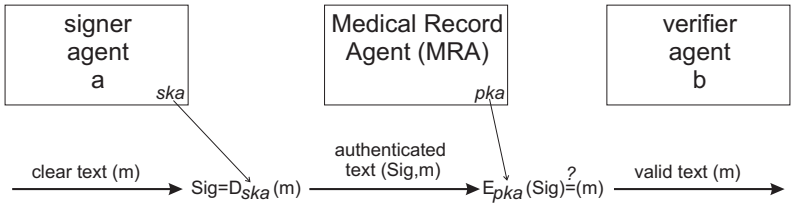


Figure 5.5: Signing a message with a public key

5.2 Personalisation of medical services

In the previous section the architecture of the agent-based platform was introduced and basic services such as searching for information about medical centres and getting the medical record, were explained. This section describes in depth the most complex service, which is the personalisation of the arrangement of medical bookings between a patient and a service.

When the doctor decides that some medical test must be performed, usually it is the patient who has to find an appointment with an external medical unit that can perform this test, and this is a problem that requires a lot of time and effort from the patient. As it is shown in Fig. 5.6, HECASE2 proposes another brokering-based protocol between the user and a service through the doctor (acting as a broker in this case). The doctor negotiates a free slot with a service (implementing a FIPA Contract Net protocol), and the doctor should confirm one of the received proposals by interacting with the user (using a FIPA Request protocol) [FIPA 2002*c*].

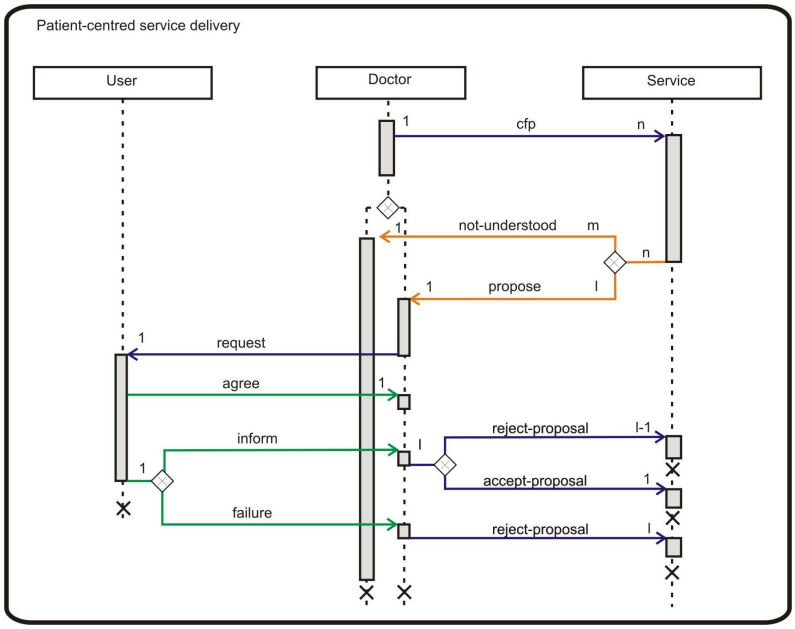


Figure 5.6: AUML diagram of the agent-based recommendation procedure

The whole procedure is achieved by combining the use of intelligent decision-making techniques and learning algorithms. The proposed algorithm for making recommendations and performing a dynamic management of the user profile was designed to be included in the HECASE2 system in order to provide a user-centred patient care [Isern et al. 2006*b*, Isern et al. 2006*a*, Bajo, Corchado, Fernández, Fuentetaja, González, Isern, López & Valls 2007].

When the user needs to arrange a meeting with a doctor, there exists a set of personal constraints and preferences that can be used to guide the search process, and hence, performing a patient-oriented service delivery (see Fig. 5.7). Using the patient preferences on different criteria, a multi criteria decision ranking process is applied to rate and rank the list of poten-

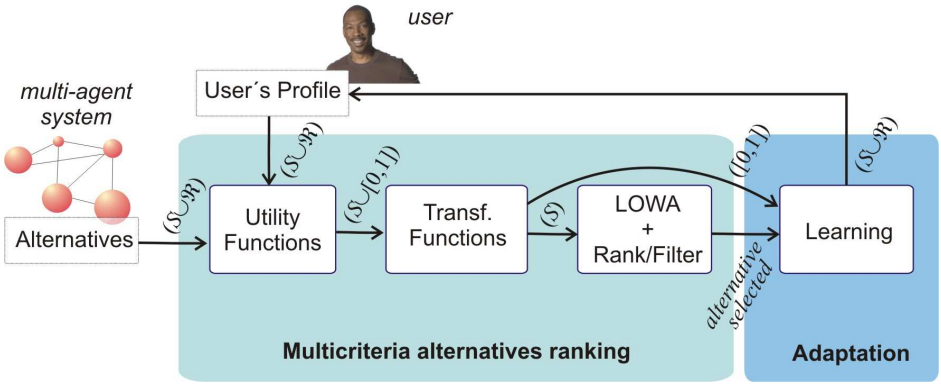


Figure 5.7: Patient-oriented delivering of information services

tial appointments (see Section 5.2.1.4). If the patient does not select the first proposal, the automatic learning process to adapt the user's profile is performed (see Section 5.2.2).

The recommendation process requires to represent the user's preferences into a profile, which consists in a tuple of criteria that stores specific values for each user. The type of criteria and their semantic meaning has been defined for this particular application; Section 5.2.1 describes how these data are represented and handled.

5.2.1 Multiple criteria decision ranking process

HECASE2 implements a decision making process that consists in rating and ranking different alternatives proposed by the system. This process is called *decision ranking*. In general, this kind of problems involves six components [Beliakov et al. 2007]:

1) A *goal* or a set of goals the decision maker wants to achieve.

2) The *decision maker* or a group of decision makers involved in the decision making process with their *preferences* with respect to the evaluation criteria.

3) A set of *evaluation criteria* (objectives and/or physical attributes).

4) The set of decision *alternatives* (actions).

5) The set of *outcomes* or consequences associated with each alternative and criteria pair.

In our case, the goal is to rank a set of alternatives from the best to the worst according to the preferences of the decision maker.

A typical scenario considers a decision situation in which a finite set of alternatives (actions) A is evaluated on a family of n criteria $g_1, g_2, ...g_i, ...g_n$, with $g_i : A \rightarrow D$ for all $i \in G = \{1, 2, ..., n\}$. We assume, without loss of generality, that the greater $g_i(a)$, the better is alternative a on criterion g_i, for all $i \in G$. D is the common domain selected by the decision maker to compare the ratings of the alternatives.

Each criterion must be able to establish preferences among the alternatives, and to allow its comparison. The preference function can lead to different relations among the alternatives:

alternative $a \in A$ is preferred to alternative $b \in A$ (denoted $a \succ b$) if only if $g_i(a) \geqslant g_i(b)$ for all $g_i \in G$, with at least one strict inequality; moreover a is indifferent to b (denotation $a \sim b$) if and only if $g_i(a) = g_i(b)$ for all $g_i \in G$; hence, for any two alternatives $a, b \in A$, one of four situations may arise in the weak dominance relation: $a \succ b$, $a \prec b$, $a \sim b$ and $a?b$, where the last one means that a and b are incomparable.

5.2.1.1 Multiple criteria decision analysis

Traditional *decision aid* is the activity of the person who, through the use of explicit but not necessarily complete formalised models, helps to obtain responses to the questions posed by a stakeholder in a decision process [Roy 2005]. These elements work towards recommending, or simply favouring a certain alternative. At the end, the decision maker is completely free to behave after the recommendation is made.

Even when *decision aiding* is provided for a single decision maker, it is rare for him to consider a single clear criterion. *Multiple criteria decision analysis* (MCDA) deals with various points-of-view (criteria). MCDA techniques can be used to identify a single most preferred option, to rank options, to list a limited number of options for subsequent detailed evaluation, or to distinguish acceptable from unacceptable possibilities.

Before analysing the designed preference model, the kind of data handled should be considered. There are two main kinds of data, depending on the measuring scale [Torra & Narukawa 2007]:

a) *Ordinal scale*. Scale such that the gap between two degrees does not have clear meaning in terms of distance between preferences; this is the case with:

   - a verbal scale when nothing allows us to state that the pairs of consecutive degrees reflect equal preference differences all along the scale;
   - a numerical scale when nothing allows us to state that a given difference y between two degrees reflects an invariant preference difference when we move the pair of degrees considered along the scale

b) *Measurement scale*. Numerical scale whose degrees are defined by referring to a clear, concrete defined quantity.

Now, let us consider two potential actions or alternatives a and b together with their respective performances on n criteria. The problem is how a *comprehensive judgement* between those actions should be performed. This problem is usually called the *aggregation problem*, and is solved through the definition of a *consensus* function $\mathbb{C}$ defined in a given domain D [Torra & Narukawa 2007]:

$$\mathbb{C} : D^n \rightarrow D$$

Let us consider the set of alternatives A as:

$$A = (a_1, ..., a_m)$$

where each alternative is composed by some criteria G taking values in a domain X, such as:

$$a = (x_1, x_2, ...x_n)$$

Then, as an example, we can consider the weighted mean as one such aggregation operator:

$$\mathbb{C}(a) = \mathbb{C}(x_1, ..., x_n) = \sum_{i=1}^{n} \frac{w_i * x_i}{n}$$

or the harmonic mean:

$$\mathbb{C}(a) = \mathbb{C}(x_1, ..., x_n) = \frac{n}{\sum_{i=1}^{n} \frac{1}{x_i}}$$

At present, there exists a large number of aggregation operators applicable to a broad range of data representation formalisms. For example, aggregation operators on the following formalisms have been considered in the literature: numerical data, ordinal scales, fuzzy sets, belief functions, dendrograms, and DNA sequences, among others. In fact, any kind of data representation formalism is adequate for applying fusion techniques because the plurality rule (mode or voting) can be applied to data of almost any type [Torra & Narukawa 2007].

5.2.1.2 User's profile

In the HECASE2 system, the user's profile includes five criteria. Three of them are linguistic variables and two are numerical:

- *day_of_week*: indicates the preference of the user to consider alternatives in all days of the week. The user should provide his preference to consider alternatives in one day or in another. For instance, if the citizen works during the week, he might prefer an appointment during the weekend.
- *centre*: destination medical centre. The user can indicate preferred medical centres. This criterion allow the user to select a set of preferred medical centres; if the system receives an alternative for other centres, this criterion will not contribute (neither positive nor negative) to the final rate of the alternative.
- *period_day*: with the following values *morning*, *afternoon*, and *night*. The user indicates his preferences per each of these periods.
- *distance*: kilometres from the origin centre to the destination. In this case, the citizen indicates the distance that could travel from the medical centre where is treated to another, in order to perform a required test. This criterion allow to distribute the patients among different medical centres.
- *delay_days*: days to wait before the test. This criterion is used to describe the days that the citizen is able to wait for a required medical test.

Table 5.1 shows an example of the user's profile with certain values for those criteria.

During the booking process shown in Fig. 5.6, the doctor collects a set of proposals from services. Each proposal contains some attributes as the date of the appointment, location (medical centre) and the time. The doctor agent (DRA) (performing the role of broker), after receiving all proposals, completes each of them with the distance between medical centres, translating the time to the period of the day where the test will be performed and calculating the number of days that the patient will have to wait for the test. After these evaluations the DRA builds a tuple, with the following structure:

$$a = \langle \textit{day\_of\_week, centre, period\_day, distance, delay\_days} \rangle$$

At this point, the user's profile is used to evaluate the different alternatives in terms of the user' preferences and this information is used to compare, rank and sort the alternatives. We assume that there are not dependencies between the input variables. This process is explained in the following sections.

User: Mr. Jones		
Criteria	*Preferred value*	
day_of_week		
	Monday	High
	Tuesday	Perfect
	Wednesday	Perfect
	Thursday	Perfect
	Friday	Perfect
	Saturday	Low
	Sunday	Very low
centre		
	Hospital Santa Tecla	Very low
	Hospital Sant Joan	Perfect
	Hospital Joan XXIII	Low
period_day		
	Morning	Very low
	Afternoon	Perfect
	Night	Low
distance		
	1	(in kilometers)
delay_days		
	2	(in days)

Table 5.1: Example of user's profile

5.2.1.3 Mapping the user's preferences

The user's profile is stored in each *user agent* (UA). The user's profile could be initialised by the user the first time that he logs in the system or initialised with default values selected from basic profiles. In both cases, the goal of our approach is that the profile evolves on time.

The user's profile stores the utility functions that allow to map the values of the alternatives into an appropriate preference domain ($profile = \{\mathcal{U}_{atr_h}, h = 1..n\}$). As noted in the first step in Fig. 5.7, we need some functions to translate all the information to two different domains: a) for numerical attributes, to the $[0, 1]$ range, and b) to a common linguistic domain called S, in the case of linguistic attributes.

In the case of numerical attributes, a utility function $\mathcal{U}^N_{atr_i}$ was designed (see Eq. 5.1). This function compares the numerical value of the i-*th* attribute in a certain alternative ($g_i(a) = r$) with the preferred value stored in the user's profile for this attribute (r_{user}). This function evaluates the difference between the stored value and the alternative value. There are different available possibilities to consider this difference, such as a linear function, logarithmic function, or polynomial function, but in our case we need to distinguish well small differences between both values and for this reason we implemented an exponential function. The value of the constant k for an attribute i was fixed experimentally from the range of the variable as $k_i = \frac{10}{(max_{atr_i} - min_{atr_i})}$. It is important that the utility function has an inverse one, because

in the learning process that we shall explain later we will need to recover the result of a value r' from an specific value of the function $\mathcal{U}^N_{atr_i}(r')$.

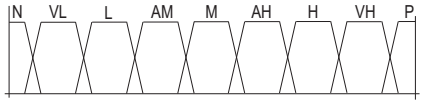
$$\begin{aligned} \mathcal{U}^N_{atr_i} : \mathbb{R} &\longrightarrow [0,1] \\ r &\longrightarrow (e^{k_i|r_{user}-r|})^{-1} \end{aligned} \tag{5.1}$$

On the other hand, we consider categorical attributes by using linguistic values. We will denote $S = \{s_i\}$, with $i \in \{0, \ldots, T-1\}$ a finite ordered set of T linguistic labels whose semantics is given by fuzzy sets. Each label s_i is defined by a 4-tuple (x_0, x_1, x_2, x_3), where x_1 and x_2 indicate the interval in which the membership function value is 1, and x_0 and x_3 are the bounds of the definition of a trapezoidal fuzzy membership function. For instance, Fig. 5.8 shows two examples of sets T considered in our tests.

Coming back to the user's profile representation, the linguistic domain S is used to represent each possible value of the linguistic variables. For instance, in the case of the *period_of_day* criterion, we need to evaluate all allowed values: *morning*, *afternoon* and *night*. As we show in Eq. 5.2, the utility function $\mathcal{U}^L_{atr_j}$ associates each possible value of the categorical attribute atr_j to a label in S, indicating its preference score.

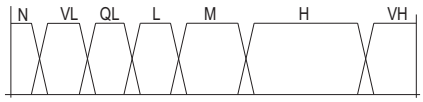
$$\begin{aligned} \mathcal{U}^L_{atr_j} : String &\longrightarrow S \\ str &\longrightarrow s_i \end{aligned} \tag{5.2}$$

P=Perfect (0.925,0.95,1.0,1.0)
VH=Very_High (0.8,0.825,0.925,0.95)
H=High (0.675,0.7,0.8,0.825)
AH=Almost_High (0.55,0.575,0.675,0.7)
M=Medium (0.425,0.45,0.55,0.575)
AM=Almost_Medium (0.3,0.325,0.425,0.45)
L=Low (0.175,0.2,0.3,0.325)
VL=Very_Low (0.05,0.075,0.175,0.2)
N=None (0.0,0.0,0.05,0.075)



(a) A uniformly distributed ordered set of nine labels with its semantics

VH=Very_High (0.8125,0.8626,1.0,1.0)
H=High (0.5125,0.5625,0.8125,0.8625)
M=Medium (0.4375,0.4875,0.5125,0.5625)
L=Low (0.325,0.375,0.4375,0.4875)
QL=Quite_Low (0.1875,0.2375,0.325,0.375)
VL=Very_Low (0.05,0.1,0.1875,0.2375)
N=None (0.0,0.0,0.05,0.1)



(b) A non-uniformly distributed ordered set of seven labels with its semantics

Figure 5.8: Examples of linguistic terms

5.2.1.4 The decision making process

The first goal of the recommendation process is to rate and rank all alternatives received from different agents. By applying an aggregation-based method, we shall obtain a label for each alternative that will be used to rank them. That method allows to handle both linguistic and numerical attributes. Fig. 5.7 depicts the process followed in that stage, which has the following steps:

(*1*) Transforming all the values contained in the received alternatives into a common vocabulary by using the utility functions described in Section 5.2.1.3. Numerical attributes are transformed into the [0,1] domain and linguistic attributes into the S domain.

(*2*) The numerical preferences in $[0, 1]$ are transformed into the linguistic domain S by means of a particular numerical-linguistic transformation function defined in [Delgado et al. 1998] (linguistic preferences are left without changes), obtaining the transformed vector $a_i = (s_1, ..., s_n)$ with $s_i \in S$ (assuming that n is the number of criteria).

(*3*) An aggregation operator ϕ is applied to all a_i in order to obtain a linguistic rating for each of them.

Finally, all alternatives can be ranked using the rating values. Then, a filtering is performed to show to the user only the best alternatives, so that he can confirm one of them.

The problem of aggregating information has been widely studied [Ahn 2006, Grabisch et al. 1999, Torra 1997]. There exist several methods to aggregate numerical values as well as linguistic terms. The family of OWA operators are in the class of mean operators, because they are *idempotent*, *monotonic* and *commutative* ([Torra 1997, Yager 1988*b*]). The LOWA aggregation operator ϕ was defined in [Herrera & Herrera-Viedma 2000], and it is an extension of the OWA operator to deal with linguistic variables [Yager 1988*b*].

Definition of the linguistic aggregation operator ϕ The operator ϕ aggregates a set of labels $a = \{x_1, ..., x_n\}$, where $x_i \in S$, with respect to a set of weights $W = \{w_1, ..., w_n\}$ such that $w_i \in [0, 1]$ and $\sum_i w_i = 1$. Those weights specify the decision-maker policy.

$$\phi(a) = \phi(x_1, \ldots, x_n) = W \cdot B^T = C^n\{w_k, b_k, k = 1, \ldots, n\}$$

$$= w_1 \odot b_1 \oplus (1 - w_1) \odot C^{n-1}\{\beta_h, b_h, h = 2, \ldots, n\}$$

where $\beta_h = w_h / \sum_2^m w_h, h = \{2, ..., n\}$ and $B = \{b_1, ..., b_n\}$ is a permutation of the elements of a, such that $B = \sigma(a) = \{x_{\sigma(1)}, ..., x_{\sigma(n)}\}$, where $x_{\sigma(j)} \leq a_{\sigma(i)}\ \forall i \leq j$. $\mathcal{C}^m$ is the convex combination operator of m labels.

If $n = 2$, then $\mathcal{C}^2\{w_i, b_i, i = 1, 2\} = w_1 \odot s_j \oplus (1 - w_1) \odot s_i = s_k,\ s_i, s_j \in S, (i \leq j)$ such that $k = min\{|S| - 1, i + round(w_i \cdot (j - i))\}$.

If $w_j = 1$ and $w_i = 0$ with $i \neq j$, then $\mathcal{C}^n\{w_i, b_i, i = 1, n\} = b_j$.

Selecting the weight vector The weight vector W characterises how the aggregation operator will work. Depending on the values, we can emphasize different values based upon their position in a decreasing ordered set. Thus, if we place most of the weights near the top of W, we can emphasize the higher scores, while placing the weights near the bottom emphasizes the lower scores in the aggregation [Yager 1988*a*]. In [Yager 1988*b*] were identified a set of different fuzzy majority-based policies such as "*most*", "*mean*", "*at least half*" or "*as many as possible*".

In order to compare the performance of one or another weight vector, the function *orness* (see Eq. 5.3) characterises to which degree the aggregation is like an *or* or an *and* operation [Yager 1988*a*, Ahn 2006]. Another measure that can be used to compare different weight vectors is the *dispersion* (see Eq. 5.4) that evaluates the degree to which we use all the attributes in the argument. Table 5.2 summarises some of the tested weight vectors with their *orness* and *dispersion* values. Ahn (2006) analysed the performance of different weight vector patterns according to their *orness* and *dispersion* values, but the concrete selection depends on the problem requirements.

$$orness(W) = \Omega = \frac{1}{n-1}\sum_{i=1}^{n}(n-i)w_i \tag{5.3}$$

$$disp(W) = -\sum_{i=1}^{n} w_i ln(w_i) \tag{5.4}$$

Entry	*Weight*	*Meaning*	*Orness*	*Disp*
1	$W_5 = (.200, .200, .200, .200, .200)$	*mean*	0,50	1,61
2	$W_5 = (.457, .257, .156, .090, .040)$	*fixing orness to 0,75*	0,75	1,34
3	$W_5 = (.399, .399, .200, .001, .001)$	*at least half*	0,80	1,07
4	$W_5 = (.001, .200, .399, .399, .001)$	*most*	0,45	1,07
5	$W_5 = (.001, .001, .200, .399, .399)$	*as many as possible*	0,20	1,07
6	$W_5 = (.996, .001, .001, .001, .001)$	*or-like*	0,99	0,03
7	$W_5 = (.001, .001, .001, .001, .996)$	*and-like*	0,01	0,03

Table 5.2: Features of weight vectors

In our domain, we need to deal with heterogeneous data coming from different attributes. As we mentioned previously in Section 5.2.1.3, the user's profile contains information about the preferred medical centre, the distance that the user is able to travel for a test, a desirable delay to wait before a medical appointment, and the most preferable period of the day and the day of the week. When service agents send a set of proposals to the user, it is very difficult to satisfy all the user's preferences at the same time. In most of the cases, only one or two of the variables are well rated and the rest have worst values.

The other parameter that affects the final result in combination with the vector W is the selection of the set S. Different policies are available ([Herrera & Herrera-Viedma 2000]) and were analysed and compared in [Isern et al. 2006*a*]. Fig. 5.8(a) shows one of the best alternatives for our domain, which allows to distinguish nine semantic meanings. Other possibilities that were considered contained seven labels and different membership functions (with symmetric and non symmetric distributions (see Fig. 5.8(b))).

Several configurations with different settings were simulated and, at the end, we selected the weight policy called *as many as possible* (Table 5.2, entry 5) with the linguistic set S consisting on nine labels distributed symmetrically (Fig. 5.8(a)).

5.2.2 User's profile adaptation

The profile stores the preference information about the criteria that describe the alternatives. If those preferences may change over time, it is desirable to update them in an explicit or implicit way in order to maintain the current user's interests. Since the users are not usually willing to provide information, implicit techniques for learning preferences are required.

In our scenario, as previously described, when the user receives a set of alternatives to consider for an appointment, the list of proposals is rated and sorted according to his preferences stored in his user's profile. Then, the list of recommended proposals is presented to the user. Now, the patient, through his personal agent, selects the most appropriate alternative for him. If he selects the first option, it means that our algorithm to rate the alternatives works fine, but if he selects another alternative, it means that for some reason the algorithm has rated too low the most appropriate alternative to the user. Using this information, the main goal now is to adapt the user's profile with this information (implicitly). If the same situation is repeated in the future, the alternative selected by the user in the past will be rated better.

5.2.2.1 Learning Algorithm

Our method is based in the following statement: if the user has selected alternative a_i with a rate s_r ($s_r \in S$), we can use the subset of alternatives $a_h,\ h \in (0, ..., i-1)$ that received better ratings (and, therefore, have better positions in the ranking) to update the user's profile.

The update of the user's profile is performed by comparing the alternative a_i with a vector that represents the subset of alternatives $(0, ..., i-1)$. This vector is calculated by the application of clustering methods, and the adaptation is performed in a similar way the the LOWA operator aggregates the information contained in the criteria.

Having that each alternative a_h includes n criteria, we propose the following algorithm to adapt the user's profile after his selection:

(1) All the linguistic preference values of the alternatives from position $j = 0$ to $j = i$ are translated to the numerical domain $[0, 1]$ by means of a particular linguistic-numerical transformation function defined in [Delgado et al. 1998] (numerical preferences are left without changes), obtaining the transformed vector called $v_j = \{c_{j\,k}\}$ ($c_{j\,k} \in [0, 1]$) (See Fig. 5.9(a)).

(2) Using an unsupervised clustering method (*e.g.*, k-means), generate c clusters from the set of alternatives $V = \{v_h, h = 0, ..., i-1\}$. Then, we calculate the prototype of each cluster, $R_j, j = 0, ..., c-1$ (See Fig. 5.9(b)).

(3) Find the distance between v_i (the alternative selected by the user) and all the prototypes R_j. Let R_{min} be the closest prototype to v_i, so that its distance[4] is $\{min(dist(R_j, v_i))\ \forall j = 0, .., c-1\}$.

(4) Let A be the vector v_i, and B be the vector R_{min}. For each of the n criteria, calculate the difference of its value in A with respect to B, $d_j = (a_j - b_j)$. The criteria for which the difference d_j is greater than a given threshold are marked to be changed in the user's profile.

(5) To update the preference values of the criteria marked in the previous step, we propose a method based on the LOWA aggregation operator described in Section 5.2.1.4.

Being A the vector v_i, B the vector R_{min}, and m the criteria to be updated, an intermediate value α_m is calculated, so that $\alpha_m = a_m + w_m|b_m - a_m|$, where the weight w_m is indicating the degree of change that we want to apply to the criterion m. We propose to use the difference of the actual value and the desired one, that is $w_m = b_m - a_m$.

(6) Using the value α_m, adapt the user's profile. For numerical variables, the new value of the m^{th} variable in the user's profile is $\beta_m = (U_m^{N^{-1}}(\alpha_m))$.

For linguistic variables, that value β_m is obtained by transforming the number α_m into its corresponding term in S, with the same function applied in *Step 1* and defined in [Delgado et al. 1998].

[4] Any distance measure can be applied. We chose the Euclidean Distance, $dist(P, Q) = \sqrt{(\sum_i (p_i - q_i)^2)}$, given two vectors $P = (p_i, i \in 1, .., n)$ and $Q = (q_i, i \in 1, .., n)$.

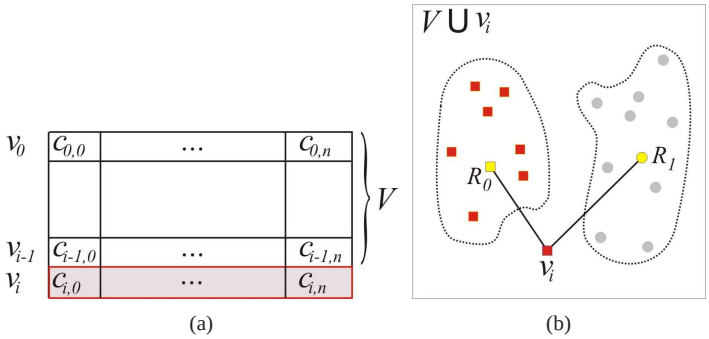


Figure 5.9: Clustering of alternatives in the learning process

5.2.2.2 Some Comments on the Learning Method

In this section we want to present some alternatives to some of the steps of the learning method we have proposed, and argue why we have chosen this configuration and not another. Let us proceed step by step.

In step 2 we use the *k*-means algorithm because it is a fast and well-known method, but other non-supervised clustering techniques could be applied [Witten & Frank 2005]. The number of clusters that are generated must be carefully studied. Depending on the application domain, we could consider to have more than two clusters. However, we must think that if the profile and the decision making methods are correct, the user should not select an alternative far form the initial positions of the ranking, so the number of alternatives to cluster should not be more than ten. With this assumption, building more than three clusters does not seem necessary.

The rationale behind step 3 is that we can have sets of alternatives with common features, and we must select one of the prototypes to become our ideal alternative, that is, the one we want to get closer to. If we select the more distant prototype, we are very optimistic and want to make all the changes necessary in our preferences to arrive to the best positions. A more conservative approach is to consider the closest prototype. If we have more than two clusters we can select any other cluster in between.

The threshold defined in step 4 allows us to restrict the number of changes in the user's profile. Depending on the number of criteria that change, the profile is adapted more or less smoothly.

In step 5, we calculate the new value for each criterion in the profile, β_m. The weight w_m indicates the degree of change we will apply to the actual value in the profile. It is based on the difference between the actual value, a_k and the desired one b_k. However, other options or more parameters could be used. For example, we could allow bigger changes at the beginning of the use of the recommendation system and, after some time, take a more conservative approach, reducing the amount of change per iteration. Note that the value of β_m can be greater or lower than zero allowing displacement in both ways from a current value of an attribute.

Fig. 5.10 shows the iterations and the evolution of a profile. This example considers five criteria and all the time the algorithm selects the alternative with the value of variable 2 best rated. Initially all the variables are rated in the middle, usually named *medium* because the

user does not know anything about the environment. At the end of these nine iterations, different variables have changed their value, and the variable 2 has suffered a bigger change. As the example shows, one transition of the algorithm can suppose the change of more than one variable, as it is the case on step 4.

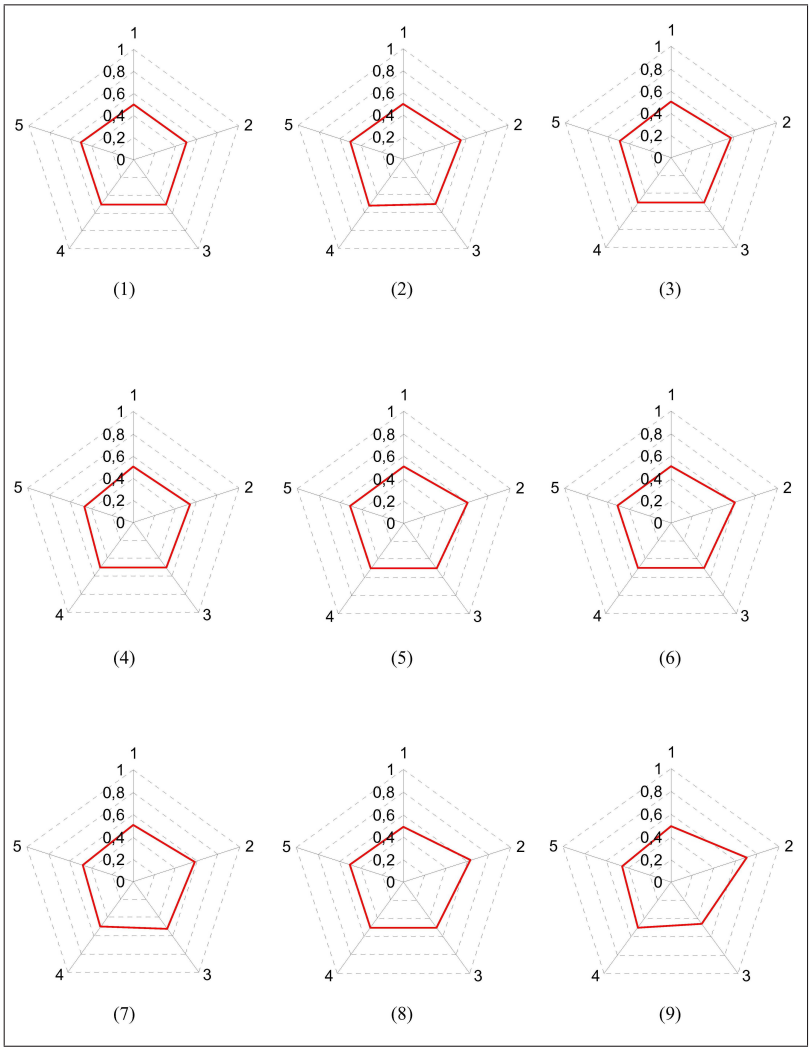


Figure 5.10: Example of evolution of a profile

5.2.3 Patient-oriented personalisation of medical services: example

This example considers the variables described in Section 5.2.1.2. Fig. 5.8 shows two possible linguistic domains S. After the study made in [Isern et al. 2006*b*], we selected a vocabulary with seven linguistic labels non symmetrically distributed (see Fig. 5.8(b)) to have more precision to indicate different bad degrees of preference.

For our example, let us consider that the profile of Mr. Smith is this one:

delay_days	$\langle$0.0$\rangle$
distance	$\langle$0.0$\rangle$
centre	$\langle$(MCBona,N)(MCBorges,M) (MCConst,H) (MCMorell,VH) (MCGimb,M) (MCHospi,L) (MCJaume,QL) (MCLlib,L)$\rangle$
day_of_week	$\langle$(Sun,VL) (Mon,QL) (Tue,M) (Wed,H) (Thu,M) (Fri,M) (Sat,L)$\rangle$
period_day	$\langle$(Morning,VH) (Afternoon,QL) (Night,VL)$\rangle$

Mr. Smith needs a tooth X-ray to check which is the origin of his recurrent toothache. Each of the proposals of appointment that the system finds is initially considered a valid alternative. Each proposal is a 5-tuple $p_i = \langle$*delay_days, distance, medical_centre, day_of_week, period_day*$\rangle$. The values on each alternative are evaluated using Mr. Smith's utility functions stored in his profile, in order to know the corresponding linguistic preference values (following [Herrera & Herrera-Viedma 2000]).

Let's consider the we have found six possible appointments P for Mr. Smith:

$$p_0 : \textit{(2.0,1.2,MCBorges,Wed,Morn)} \xrightarrow[transf]{U} \text{(H H M H VH)}$$
$$p_1 : \textit{(1.0,8.0,MCConst,Mon,Aft)} \rightarrow \text{(H VL H QL QL)}$$
$$p_2 : \textit{(4.0,9.0,MCBona,Thu,Aft)} \rightarrow \text{(L VL N M QL)}$$
$$p_3 : \textit{(5.0,1.2,MCBorges,Sat,Night)} \rightarrow \text{(QL H M L VL)}$$
$$p_4 : \textit{(10.0,1.2,MCBorges,Fri,Morn)} \rightarrow \text{(N H M M VH)}$$
$$p_5 : \textit{(9.0,17.0,MCHospi,Fri,Aft)} \rightarrow \text{(VL N L M QL)}$$

In the next step, the LOWA operator is applied to rate the proposals. An important parameter to be set in this stage is the weight vector W. As it has been mentioned before, weights specify different aggregation polices. In this application we have good results with the policy *"as many as possible"* [Yager 1988*b*], so weights are: $W = (.0, .0, .2, .4, .4)$. After applying the LOWA operator, all alternatives are linguistically rated and can be ranked and presented to the user for a selection. In the example, the ranked list of appointments for Mr. Smith is the next one:

$$p_0 \succsim p_4 \succ p_1 \succsim p_3 \succ p_2 \succsim p_5$$

$$p_0 : \text{(H H M H VH)} \rightarrow \text{(H)}$$
$$p_4 : \text{(N H M M VH)} \rightarrow \text{(H)}$$
$$p_1 : \text{(H VL H QL QL)} \rightarrow \text{(M)}$$
$$p_3 : \text{(QL H M L VL)} \rightarrow \text{(M)}$$
$$p_2 : \text{(L VL N M QL)} \rightarrow \text{(L)}$$
$$p_5 : \text{(VL N L M QL)} \rightarrow \text{(L)}$$

Let us suppose that Mr. Smith is not suffering from toothache at the moment, and that he needs some time to arrange his schedule to include the medical appointment. Thus, he selects option p_5, although it is in the sixth position of the ranking. Therefore, we assume that his profile is not completely accurate and we will try to modify it. In fact, if we observe the profile, we will see that the most preferred option is to wait for 0 days, but this point has not been important in the choice made by Mr. Smith. Now, we start the learning process to adapt the user's profile to the selection made.

The set $V = \{v_0, v_4, v_1, v_3, v_2\}$ is built, and the two clusters obtained using the *k*-means algorithm are evaluated. To apply this method we need to have all the information in a numerical scale. To do this we apply the linguistic-numerical transformation function described in [Herrera & Herrera-Viedma 2000] (see Fig. 5.7). As a result of that transformation, the alternatives are described as follows[5]:

v_0 : (H H M H VH) $\leftrightarrow$ (0.6065 0.7408 0.4999 0.6875 0.9259)
v_4 : (N H M M VH) $\leftrightarrow$ (0.0821 0.7408 0.4999 0.4999 0.9259)
v_1 : (H VL H QL QL) $\leftrightarrow$ (0.7788 0.1353 0.6875 0.2812 0.2812)
v_3 : (QL H M L VL) $\leftrightarrow$ (0.2865 0.74082 0.4999 0.4062 0.1437)
v_2 : (L VL N M QL) $\leftrightarrow$ (0.3679 0.1054 0.03055 0.4999 0.2812)
v_5^* : (VL N L M QL) $\leftrightarrow$ (0.1054 0.0143 0.4062 0.4999 0.2812)

The k-means algorithm generates two clusters $\{v_1, v_2, v_3\}$ and $\{v_0, v_4\}$, with their corresponding centroids, called R_0 and R_1, respectively.

R_0 : (0.4777 0.3272 0.4060 0.3958 0.2354)
R_1 : (0.3443 0.7408 0.4999 0.5937 0.9259)

[5]To improve legibility, all numerical values have been rounded to four decimals. Internal operations maintain more precision.

Now, we measure the distance between the selected alternative (v_5^*) and the two prototypes. We apply the Euclidean Distance obtaining:

$$dist(v_5^*, R_0) = 0.2234 < dist(v_5^*, R_1) = 0.4512$$

Considering a *conservative* approach, we decide that we should adapt the user's profile to be closer to R_0.

In the next step, we calculate the difference d between v_5^* and R_0, obtaining:

v_5^*: (0.1054 0.0143 0.4062 0.4999 0.2812)
R_0: (0.4777 0.3272 0.4060 0.3958 0.2354)
$|d|$: (0.3723 0.3129 0.0002 0.1042 0.0458)

At this point, we have to establish a threshold to choose which attributes are appropriate to be considered in the profile's update. If we choose a low threshold, too many attributes will be changed simultaneously and the profile modification will be bigger and more difficult to control. On the other hand, a threshold too high will not allow the system to react to the changes in the preferences of the user. Let us suppose that we take a threshold[6] of 0.35; in that case, we only have to change the first attribute: the *delay_days*.

The adaptation of that variable is made using to the aggregation-based updating function defined in the proposed learning algorithm. According to that, $\alpha = 0.1054 + w(0.4777 - 0.1054)$ where $w = 0.3723$, resulting $\alpha = 0.2440$. This means that the preference of the value proposed in v_5 should have been 0.2440 instead of 0.1054. Therefore, to change this preference utility function, we can only change the number of days considered to be the most preferred by the user. In this example, we initially assumed that Mr. Smith wants to wait for 0 days, but from the scenario we have noticed that this is not true. Applying the inverse function to obtain the number of days from the preference value 0.2440, we find that the most preferred number of days is set to 5.3, which is a good approximation of the real desired value (it is an intermediate value between the selected proposal and the value stored in the user's profile).

[6]This value has been found from different tests and it depends on the application domain.

Now, we can check what happens with this new profile. So, we start again the ranking process with the same alternatives. The results obtained now are shown below. Notice that the change in the utility function of the $delay\_days$ variable has affected the preference values of all the proposals.

p_0 : (**L** H M H VH) $\rightarrow$ (H)
p_1 : (**QL** VL H QL QL) $\rightarrow$ (L)
p_2 : (**H** VL N M QL) $\rightarrow$ (M)
p_3 : (**VH** H M L VL) $\rightarrow$ (H)
p_4 : (**QL** H M M VH) $\rightarrow$ (H)
p_5 : (**L** N L M QL) $\rightarrow$ (L)

Our goal was to increase the preference value of p_5 smoothly. In fact, it can be observed that now p_5 is considered of L (*low*) preference in its first attribute, instead of VL (*very-low*). Moreover, the changes suffered by the first variable have some effects in the ranking, changing the positions of some of the alternatives. Some alternatives with a delay close to the new preferred value have increased its global utility value such as p_3. However, in this particular example this change is not sufficient to modify the overall ranking of the proposal selected by Mr. Smith. Note that the method does not aim to improve automatically all variables in one step (that would be too drastic). Moreover, there are several parameters that affect the degree of change, such as the threshold taken in the adaptation or the centroid chosen to adapt the selected proposal.

5.2.4 Related work

5.2.4.1 Patient-oriented services

The change towards patient-centred policies has been investigated in several projects, with some results and prototypes. One of the first attempts was made by Curé (2003). His proposal was designed to educate citizens (patients) in health care issues. In this case, the system allows to exchange data (in a structured way) with computerised patient records, to look for information related with drugs, symptoms, or opinions of health care experts, and finally, to include the patient into a community (of specific diseases, geographical) that allows to share experiences about a common issue or problem. Ghinea et al. (2004) proposed an architecture for a distributed collaborative e-health multimedia application that incorporates an intelligent mechanism for obtaining a priority order of low-level QoS parameters, which ensures that expected user quality is maintained at an acceptable level across dynamically varying network conditions. This system was designed to facilitate the user care through an effective remote application, avoiding medical visits to the general practitioner.

Recently, Xanthos (2007) analysed the current delivery of care in Barbados, and how the services that are handled in a provider-focused fashion can be translated into a patient-focused perspective. She concludes that while the concept of patient-focused care has gained some recognition in Barbados over the last years, several practitioners considered health care delivery was in general not patient-focused. She also identified three kind of applications:

patient-focused, provider-focused and intermediate. This classification can be extended to more countries. Flatley (2007) introduces some patient-focused computer systems and depicts some benefits to both citizens and practitioners. Applications such as health-related websites, consumer health informatics tools, patient portals to hospital records and clinical resources, and palm-top reminders for medications and disease management, help lay people and their caregivers better understand their health challenges, participate in health care choices, cope with the implications of disease and injury, and maintain contact with their clinical care providers. These innovations help reduce health disparities and increase knowledge of, and involvement in, their own health care processes, and provide ongoing, point-of-living monitoring of complex health problems.

An ongoing project called *K4Care* is developing an infrastructure to adopt general intervention plans for each patient in a semi-automatic way (see also Section 7.1 and Campana et al. (2008)). This approach will implement accurate treatments to patients that usually suffer more than one disease at the same time. In fact, the execution of personalised clinical guidelines to each patient has several works with numerous representations and tools addressing this issue [Leong et al. 2007].

PIPS (Personalised Information Platform for Health and Life Services) is an e-Health EU-funded research project, which aims to create novel health care delivery models by building an environment for Health and Knowledge Services Support [Domínguez et al. 2006]. The environment integrates different technologies in order to enable health care professionals to get access to relevant, updated medical knowledge, and European citizens to choose healthier lifestyles. In order to accommodate in the system many types of devices and users, with different roles and needs, a multi-agent system approach was selected as the natural choice for the design of the core system component, the decision support layer.

5.2.4.2 Classification and recommendation of alternatives

As has been shown throughout this section, all recommender systems have two main elements: (*i*) the profile representation and maintenance, and (*ii*) the method for exploiting this profile in order to evaluate and rank a set of alternatives. First of all an overview of recommendation methods will be given, followed by an study of the available methods for the profile learning. Then, several comments about the reasons to select one method or another will be given.

User's recommendation With respect to how the recommendation method provides a personalised answer to some decision problem, different approaches can be considered. For the particular problem faced in this dissertation, we have proposed the use of a multi-criteria decision making (MCDM) technique [Figueira et al. 2005]. MCDM methods have their foundations in Philosophy, Economics, social choice or even game theory. Note that aggregating the opinion of preferences of voters or individuals of a community into collective or social preferences is quite similar to devising comprehensive preferences of a decision-maker from a set of conflicting criteria.

The two main approaches to MCDM are *outranking* methods and *multi-attribute utility theory* (MAUT) [Figueira et al. 2005]. One the one hand, outranking methods seek to establish the strength of evidence favouring selection of one alternative over another, on the basis

of pairwise comparison of alternatives. The result of those comparisons is represented by an outranking binary relation S defined on the set of alternatives A, such that aSb if there are enough arguments to decide that a is at least as good as b, whereas there is no essential argument to refute that statement. On the other hand, MAUT is based on assigning a global utility value to each alternative in A. This global utility is a combination of the marginal utilities that each criterion assigns to an alternative [Linkov et al. 2006]. MAUT is a simple approach that is quite used in MCDM. Besides those two main approaches, other methods have been developed; a revision of them can be found in [Figueira et al. 2005].

As it has been presented before (Section 5.2.1), preferences can be represented with different types of values, mainly numerical or linguistic. In general, aggregation operators can be classified according to the data type (numerical, fuzzy, qualitative, heterogeneous) or according to their mathematical properties.

The main families of aggregation operators are ([Beliakov et al. 2007, Torra & Narukawa 2007, Yager 1988*b*]):

- Means (averaging functions), like arithmetic mean, weighted mean, geometric mean, or harmonic mean.
- Medians, which try to find a value that is more representative of a typical value than the mean. It essentially discards very high and very low values.
- Ordered weighted averaging functions (OWA), which are also averaging aggregation operators which associate weights not with a particular input, but rather with its value. According to the nature of the data, numerical or linguistic, OWA or LOWA operators can be defined, respectively.
- Choquet and Sugeno integrals, which are two classes of averaging functions defined with respect to a fuzzy measure. They are useful to model interactions between the criteria.
- Conjunctive and disjunctive functions, like the so-called triangular norms and conorms respectively. Minimum and maximum functions, product and probabilistic sum, Lukasiewitch norms, or drastic sum and product, are several examples of these aggregation functions that are used in fuzzy set theory and fuzzy logic.
- Mixed aggregation, used in situations where high input values are required to reinforce each other, whereas low values pull the output down. In this case, the aggregation function has to be disjunctive for high values, conjunctive for low values, and perhaps averaging if some values are high and some are low. The classical expert systems MYCIN and PROSPECTOR used this type of aggregation [Hájek & Valdés 1994].

Learning preferences Methods for learning preference models and predicting preferences are among very recent research trends in fields like machine learning and knowledge discovery. Approaches relevant to this area range from learning special types of preference models, such as lexicographic orders, over collaborative filtering techniques for recommender systems and ranking techniques for information retrieval, to generalisations of classification problems such as label ranking. Like other types of complex learning tasks that have recently entered the stage, preference learning deviates strongly from the standard problems of classification

and regression. It is particularly challenging as it involves the prediction of complex structures rather than single values. Moreover, the acquisition of preferences is not always an easy task. Therefore, not only are modelling languages and formalisms needed, but also methods for the automatic learning, discovery and adaptation of preferences.

In our approach, the proposed algorithm allows to deal with dynamic preferences by changing the user's profile values in an implicit way (taking profit of the user's selection of an appointment), avoiding the explicit interaction with the user. Other recent works in this area are:

- Refining linear constraints on multi attribute utility functions, allowing a set of Pareto optimal decisions to be identified. Some procedures have been defined for incremental elicitation of utility functions that attempt to reduce minimax regret with as few equations as possible [Wang et al. 2003].
- Learning a decision maker's utility function from the decision maker's observed behavioural patterns, by means of finding a utility function which (together with a domain model) can explain the user's behaviour [Nielsen & Jensen 2004].
- Preference learning for adaptative interaction in intelligent assistants (that give you reminders, requests for permissions, etc.) [Weber & Pollack 2008].
- Automatic construction of a user interest hierarchy that represents a user's interests at different abstraction levels, which is learned from the contents (words or phrases) in a set of web pages bookmarked by a user [Kim & Chan 2008].
- Using feedback about activities done during a trip to acquire implicit knowledge about the user's interests [Bajo, Botti, Corchado, Ilarramendi, Ilarri, Julián, Carmona, Marsá, Mena, Moreno, Pavón & Valls 2007].

Discussion According to the characteristics of the problem presented in this dissertation (the ranking of appointments) the most suitable approach is multi-attribute utility theory. In this model, it is assumed that each criterion is directly associated to a measurable attribute.

Moreover, some conditions are necessary and sufficient for preference criteria to satisfy the *utility hypothesis* ([Belton & Stewart 2001]). Assuming that there exists an ordering axiom between preferences, it requires that a preference relation $\succsim$ fulfils,

- *completeness*, *i.e.*, for all $p, q \in S$, $p \succsim q$ or $p \precsim q$, and
- *transitivity*, *i.e.*, for all $p, q, r \in S$, if $p \succsim q$ and $q \succsim r$ then $p \succsim r$, and
- the assumption of *mutual preference independence* must hold, and
- *continuity of preferences*; an assumption of non-continuity of preferences implies the impossibility of ordering the decision maker's preferences by a monotonic numerical representation or utility function.

The criteria considered in this work satisfy these conditions, so they can be represented by means of utility functions.

In contrast, the outranking approach is more appropriate for situations with the following characteristics ([Roy 1991]): there are more than five criteria, some alternatives are evaluated

in an ordinal scale or interval scale (scales that are not suitable for comparison of differences), strong heterogeneity exists among criteria, which makes it difficult to aggregate them into a unique scale, compensation of the loss on a given criterion by a gain on another one may not be acceptable, and finally, for at least one criterion, small differences of evaluations are not significant in terms of preferences, while the accumulation of several small differences may become significant. For this domain, most of these conditions are not fulfilled.

MAUT approach was selected due to the flexibility to describe the user's profile by a set of utility functions as described in Section 5.2.1.3. In this case, the nature and the number of criteria are not a constraint to apply this approach. The lack of compensation between criterion values can be handled by the family of operators, called ordered weighting averaging (OWA) [Yager 1988*b*]. Particularly, linguistic OWA operator (LOWA) allow to work with a rational domain of data (qualitative) and easier to use by human beings [Herrera et al. 2002].

5.3 Conclusions

The first part of the chapter has introduced the main features of the HECASE system, by describing the main services delivered by the platform. First of all, searching for information about medical centres satisfying certain properties. This service is implemented through a broker agent that intermediates between the user and the provider of the information, and provides transparency between the services implemented inside the platform from external user agents. The management of the patient's health record, has also been explained. In this case, the most important feature is the addition of security measures around the medical record agent in order to provide authentication, integrity and confidentiality to these sensitive data. The final and most complex service is the implementation of a patient-oriented booking of appointments between the user and the care providers.

The overall patient-oriented delivery of medical services is made in two main stages. The first one automatically collects a set of proposals received according to a service to be delivered, and ranks them with the support of a user's profile. To achieve this stage, an aggregation-based process is performed. To work with linguistic criteria a LOWA algorithm was selected. After evaluating all the items, a filtered ranking is performed in order to avoid low rated options and offer to the user the best ones (best rated according to his preferences). Different parameters to achieve this aggregation have been set up and explained deeply through Section 5.2.1. The second stage observes the selection made by the user and adapts the profile accordingly (Section 5.2.2). The proposed learning algorithm estimates which criterion or criteria, should be adapted, and changes their values accordingly. The selection of the candidate to change and its own change are the most difficult tasks, because the algorithm should estimate why the user selected one option in front of the others, and the underlying reason of the user is not known. In the method proposed in this chapter, some parameters can be adjusted to tune the behaviour of the learning process, with the experience of the real use of the system.

In conclusion, we have designed and implemented an unsupervised method for providing patient-oriented services in the health domain.

Chapter 6

Ontology-driven execution of clinical guidelines

A *clinical guideline* indicates the protocol to be followed when a patient is diagnosed a certain illness (also called *know-what*). They provide very detailed information concerning the resources needed in the treatment of a patient [Boxwala et al. 2001].

A hard task to be accomplished is the inclusion of a guideline execution engine in the daily work flow of practitioners. This chapter deals with the representation of how the tasks embedded into a CG are managed (also called *know-how*). This knowledge has been designed as an element (ontology) external to both the agents beliefs and the CG codification.

With that approach, care is improved at least in four ways:

i) Ontologies provide a common understandable semantic framework to execute clinical guidelines. Consequently, all the entities and concepts involved in that execution can be explicitly defined according to their relations and attributes.

ii) Agents can understand what they must perform at any moment and negotiate or coordinate their activities with the appropriate partners.

iii) Ontologies provide a high level abstraction model of the daily work flow. That model can be adapted to each particular organisation, without the agents having to change their internal behaviour. In that sense, any organisation can have an ontology adapted to its particular circumstances.

iv) There are different representations to code *clinical guidelines* but most of them share a common set of elements and features. Ontologies can be used to define the structure of a generic CG that can be used to share the information contained in CGs coded in different representations.

Ontologies They define terms and relations comprising the vocabulary of a topic area as well as the rules for combining terms and relations to define extensions to the vocabulary [Neches et al. 1991].

Different knowledge representation formalisms exist for the definition of ontologies. However, they share the following minimal set of components (see Fig. 6.1):

a) *Classes*: represent concepts. Classes in the ontology are usually organised in taxonomies through which inheritance mechanisms can be applied.

b) *Individuals*: are used to represent real world entities.

c) *Properties*: they represent binary associations between ontological entities. On the one hand, *object properties* establish relationships between pairs of individuals. On the other hand, *data type properties* relate an individual to a data value (integer, string, float, etc.); they can be considered attributes.

There exist different representation languages for ontologies, such as XML and RDF. Nowadays, one of the most used languages is Web Ontology Language (OWL) [McGuinness & Harmelen 2004]. There are three different flavours of OWL with varying levels of expressiveness: OWL Full, OWL Lite and OWL DL. For our purposes we need the maximum level of expressiveness but maintaining a standard structure (classes and properties) to allow inference. For these reasons OWL DL was used.

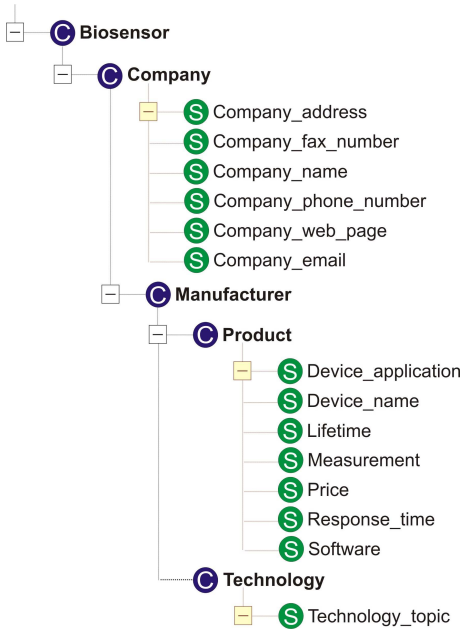
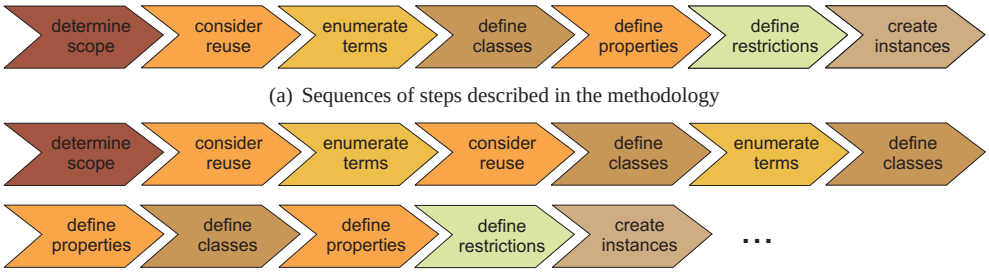


Figure 6.1: Example of ontology in the biotechnology domain; classes and properties are shown. [Sánchez et al. 2006]

Methodologies for building ontologies There does not exist a unique way for modelling ontological knowledge [Corcho et al. 2003]. From the ontology engineering point of view, several methodologies and guides have been designed in the past for aiding the ontology construction process such as METHONTOLOGY and ON-TO-KNOWLEDGE [Gómez-Pérez et al. 2003]. From all of them, the *101 ontology development method* has been selected ([Noy & McGuinness 2001]) due to both its flexibility and independence from the final language description. It divides the ontology construction process in several iterative steps, covering from the definition of the scope to the specification of each ontological entity (see Fig. 6.2(a)). It also provides golden rules about how an ontology should be defined. Each step can be executed as many times as desired and in any particular order, allowing to create the final ontology in a flexible and incremental way (see Fig. 6.2(b)).



(a) Sequences of steps described in the methodology

(b) Iterative deployment of the steps

Figure 6.2: Sequences of steps described in the *101 ontology development method* [Noy & McGuinness 2001]

The rest of the chapter is organised as follows. First of all, Section 6.1 gives an overview of the main applications of ontologies in medicine. Then, Section 6.2 explains the ontology-based representation of CGs used to manage CGs coded in different languages using a common representation among agents. Section 6.3 details another ontology designed to represent medical and organizational knowledge used during the execution of a CG; the HECASE2 system needs to complement the procedural knowledge contained in CGs with declarative knowledge stored in this ontology. Section 6.4 describes a case study using all these ontological tools. Finally, the conclusions section summarises some concluding remarks of this chapter.

6.1 Use of ontologies in medical applications

The use of ontologies in medicine has been shown to suppose an important advantage. Gong et al. (2007) designed an ontology, called *medical error*, to improve patient safety and reduce medical errors. The ontology allows healthcare professionals to report, in a structured way, medical errors. These data are used for detecting patterns of (erroneous) behaviours, and discovering underlying factors, in order to propose solutions.

In addition, Dixon et al. (2007) created a taxonomy of health IT terms. In this case, the taxonomy has been included in a web site portal in order to facilitate the citizens to find information.

Kumar et al. (2003) studied the implementation of a task ontology named Context-Task Ontology (CTO) in order to map the knowledge required in the implementation of CGs. They implemented the CTO using DAML+OIL and intended to create CGs through an independent structure that stored all relations and concepts in a unique way. They noted that this approach had some drawbacks, such as the difficulty to define and know exactly which relations are required, as well as the requirement of expert's intervention. The same authors later described the use of ontologies to define clinical guidelines by adding a hierarchy of classes to represent medical procedures and plans [Kumar et al. 2004]. However, this implied a high level of complexity as compared to flow-chart-based representations. Abidi et al. (2007) also designed an ontology-based representation of CGs (in this case, focused in the breast cancer domain) used to translate the data contained in CGs into a set of rules, which are handled by a rule-based execution engine.

Serban et al. (2007) proposed the use of an ontology to guide the extraction of medical patterns contained in CGs in order to reconstruct the captured control knowledge. These works suggest the use of UMLS as a central corpus.

Ciccarese et al. (2004) introduced an architecture that linked a care flow management system and a guideline management system by sharing all the data and ontologies in a common layer. They proposed to represent medical and organisational information in those ontologies, but they did not use non-taxonomic relations in the ontologies; all the information is stored in the flow chart-like representation used to code CGs (see the complete description at Section 2.5).

Moreover, Davis & Blanco (2005) suggested the use of taxonomies to model the clinical life cycle knowledge. They also described a state-based data flow model to represent all dependencies between enterprise entities.

The web-based application BioPortal[1] allows to access the Open Biomedical Ontologies (OBO) library (see Fig. 6.3). This library contains a large collection of ontologies in biomedicine as well as biology, chemistry, anatomy, radiology, and medicine. It permits users to browse individual ontologies and provides a suite of tools for developers to integrate its functionality into their own applications. A preliminary version of this site was Knowledge Zone [Supekar et al. 2007].

[1]The website is maintained by The National Center for Biomedical Ontology that is part of the National Centers for Biomedical Computing supported by the NIH Roadmap, USA. URL: http://www.bioontology.org/tools/portal/bioportal.html [last access 26/04/2008]

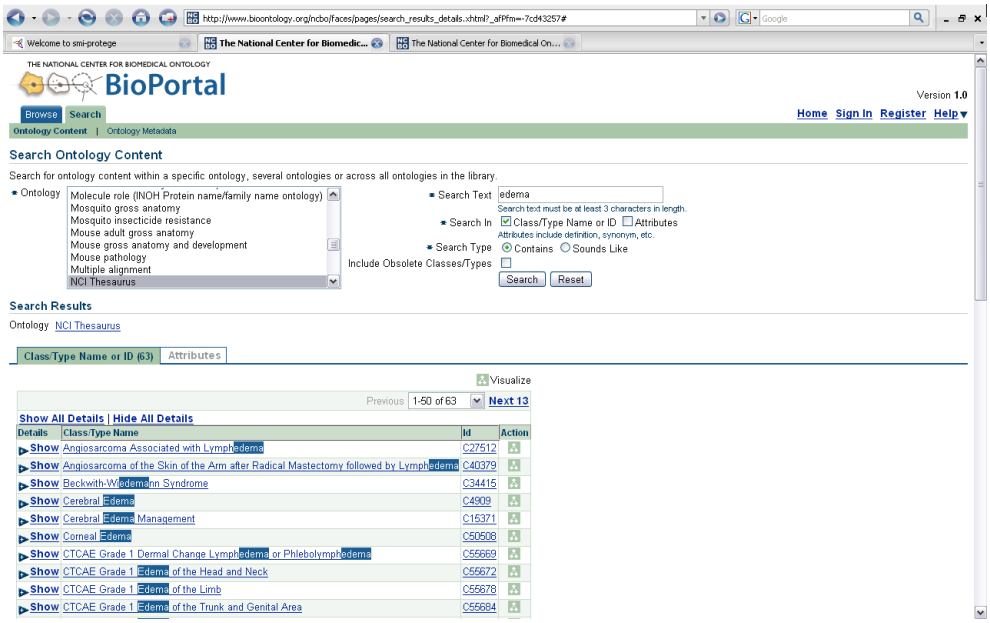


Figure 6.3: Screenshot of BioPortal, a web-based repository of medical ontologies

6.2 Ontological representation of clinical guidelines

As said in the introduction of this dissertation and studied in Chapter 2, one of the barriers for adopting clinical guidelines in careflow management systems is that there are different representation languages of clinical guidelines, and none of them is widely used. Nowadays, the authors of guideline-based execution engines have to select one of the available languages, and implement and *ad hoc* tool [Isern & Moreno 2008*b*].

Meanwhile there is no created a standard, guideline-based execution engines should select one of the available languages, and implement a language-depending tool.

The aim of this section is to present an approach to create a representation of clinical guidelines that includes properties from existing languages to allow both to obtain information about CGs and, at the same time, to maintain the original sources of these guidelines. This structure stores information about the steps to follow, and uses an interface with general methods to allow enacting the guideline (these methods allow getting and putting the required data and knowing the current state of the patient into the treatment). These facilities have been implemented as an external module and they can be attached to an actor. Particularly, they can be attached to two different actors: the doctor agent, or the guideline agent. We selected the doctor because this module requires values of tests results or findings obtained from the medical record, and these data are only available (authorised) for a doctor. Moreover, with this approach, the DRA uses this module when required and only for his patients.

After this brief introduction of the requirements and goals to accomplish, a description of the implemented module and the way it handles the CG knowledge is done in the following sections.

6.2.1 Clinical guideline ontology: motivation and features

The *clinical guideline ontology* has been developed after an exhaustive study of the main available languages to represent clinical guidelines. Unfortunately, different attempts to create a standard to represent CGs such as GLIF or GELLO, had limited success, and none of those proposals was successfully adopted by researchers and widely used in guideline-based execution engines [Clercq et al. 2004].

As shown in [Isern & Moreno 2008*b*], there are several languages to represent guidelines but, at the same time, as [Peleg et al. 2003] and [Clercq et al. 2004] noticed, they share common functionalities and elements that permit to establish a set of common characteristics. Following this assumption, the designed ontology is based on PRO*forma* [Sutton & Fox 2003], SDA* [Riaño 2007] and SAGE [Tu et al. 2007]. Other languages such as Asbru [Young et al. 2007], GLIF3 [Boxwala et al. 2004], and EON [Shankar et al. 2002] were analysed, but the differences between them, the lack of tools to manage them, and the limited availability of CGs coded using those languages, were the reasons to discard them.

6.2.1.1 PRO*forma*

PRO*forma* is an executable process modelling language that has been successfully used to build and deploy a range of decision support systems, guidelines and other clinical applications [Sutton & Fox 2003].

It has a declarative format defining four basic types of tasks (*plans*, *decisions*, *actions* and *enquiries*) as well as logical and temporal relationships between them. An *action* is a procedure to be carried out (usually by an external element like a doctor or a medical resource). A *plan* is the basic building block of a clinical guideline and represents a container for a number of tasks, including other plans. A *decision* is a task that represents an option in terms of different logic commitments to be accomplished. An *enquiry* is a request for further information or data required before proceeding with the application of the guideline.

To support the entire decision life cycle, the proposed model combines argumentation with ideas from modal logic, logic programming, and agent theory. In this approach, a decision process is defined as a special kind of object (a task) whose attributes include: a) a situation (the logical conditions requiring a decision), b) a goal (what the decision is intended to achieve), c) candidates (a schema for proposing candidate decision options), d) arguments (propositional rules or first order schemas that specify how to argue for or against competing candidates), and e) commitments (rules and procedures for selecting the most preferred candidates).

6.2.1.2 SDA*

SDA* is based on flowcharts and enriched with specific health care elements [Riaño 2007] (see Section 7.1.2.2). The basic elements of SDA* structures are *states*, *decisions* and *actions*. *States* represent patient conditions, situations, or statuses that deserve a particular course of action which is totally or partially different from the actions followed when the patient is in other state. *Decisions* allow the integration of the variability a treatment has depending on the available information about the patient. An *action* constitutes the proper health care activity in the treatment. Between those elements, directed edges define the direction of the steps and

can be used to define periodic or temporal constraints. This codification allows the medical experts to represent sequences, loops or concurrence between tasks.

6.2.1.3 SAGE

The internal representation of guidelines in SAGE is made using the EON formalism implemented as a Protégé add on [Tu & Musen 2001]. Using SAGE, a medical expert can encode computable guideline content as recommendation sets using only standard terminologies and standards-based patient information models. The SAGE Model supports encoding large portions of guideline knowledge as re-usable declarative evidence statements and supports querying external knowledge sources.

SAGE distinguishes two formalisms: *recommendation-set* and *decision-map* [Tu et al. 2007]. The *recommendation-set* is an activity graph composed of processes and interactions between them. Activity graphs allow the specification of computational algorithms or medical care plans as processes consisting of i) *contexts*, that are combinations of a clinical setting (*e.g.*, outpatient visit in a general internal medicine clinic), care providers to whom the recommendation is directed, relevant patient attributes (*e.g.*, patient age), and possibly a triggering event (*e.g.*, a patient checking into the clinic), ii) *decision nodes*, that evaluate conditions on variables (*e.g.*, a Boolean precondition for an action), iii) *action nodes*, that encapsulate a set of work items that should be performed either by a computer system or by a healthcare provider, and iv) *routing nodes*, that are used purely for branching and synchronization of multiple concurrent processes.

6.2.1.4 Common features

As said above, there are several similar features between the selected languages. Table 6.1 summarises the main topics considered to design our ontology.

All languages share the same elements: a method to introduce required values (*e.g.*, a finding, a result, a comment), a function to make decisions according to the values of some attributes considered in a certain point of the guideline, a method to describe actions to perform according to treatments, and requests of needed clinical tests. Temporal constraints between elements are also considered because all languages permit this functionality. Finally, one of the goals is to maintain the original source of guidelines in order to reuse existing run-time engines. SDA* and PRO*forma* provide this facility, but in the case of SAGE, an interpreter of the XML documents was implemented.

Topic	PRO*forma*	*SDA**	*SAGE*
Basic elements	Plan, enquiry, decision, action	State, decision, action	Context, decision, action, routing
Codification	R2N	XML	XML
Available CGs	Yes	Yes	2 (public workbench)
Execution engine	Yes	Yes	No
Temporal constraints	Yes	Yes	Yes

Table 6.1: Comparison between PRO*forma*, SDA* and SAGE

6.2.2 Clinical guideline ontology

The clinical guideline ontology has been coded in OWL DL, and edited using Protégé-2000 release 1.3[2]; Fig. 6.4 depicts the whole view. It has been implemented following the *101* methodology explained above (Fig. 6.2).

The designed ontology has the following parts:

- *Guideline*. This is the abstract part of the ontology and it is used to share the main features of a guideline. It includes information about the language used to code the guideline, the guideline itself, a general description, a set of constraints used to filter this guideline (*e.g.,* symptoms associated to a guideline that are used as pre-conditions during a search), and a slot that describes the operation that an agent wants to perform (*e.g.,* getting, updating).

- *Content*. This is the main part of the clinical guideline ontology. It includes information about the tasks (*e.g.,* enquiry, decision and action), and data definitions (different data descriptions allowed by the guideline execution engine). The former defines a set of common slots to all tasks used by all of them. All tasks include information about pre- and post-conditions, descriptions, and captions (*e.g.,* descriptions showed in the visualiser tool). In the case of decisions, logical arguments are handled; in the case of enquiries, the required data sources; and, in the case of actions, the action block embeds the information about the action (unstructured and textual).

The *content* part of the clinical guideline ontology shown in Fig. 6.4 includes more specific classes named: *Expression*, *Assertion* and *Parameter*. These classes are described in Fig. 6.5. An *Expression* is a general class that allows to explain descriptions, goals, and other abstract terms. An *Assertion* allows to describe a logical expressions in terms of arguments and conditions. A *parameter* is used to describe task definitions as the root plan. An auxiliary class used during the enactment of CGs is the *Action block* class. In this case, this class stores information about who is able to run a task (this information is contained in the SDA* codification and it is stored in another ontology designed to follow clinical guidelines explained in next sections).

[2]The Protégé ontology editor is freely available at http://protege.stanford.edu/ [last visit 12/07/2008]

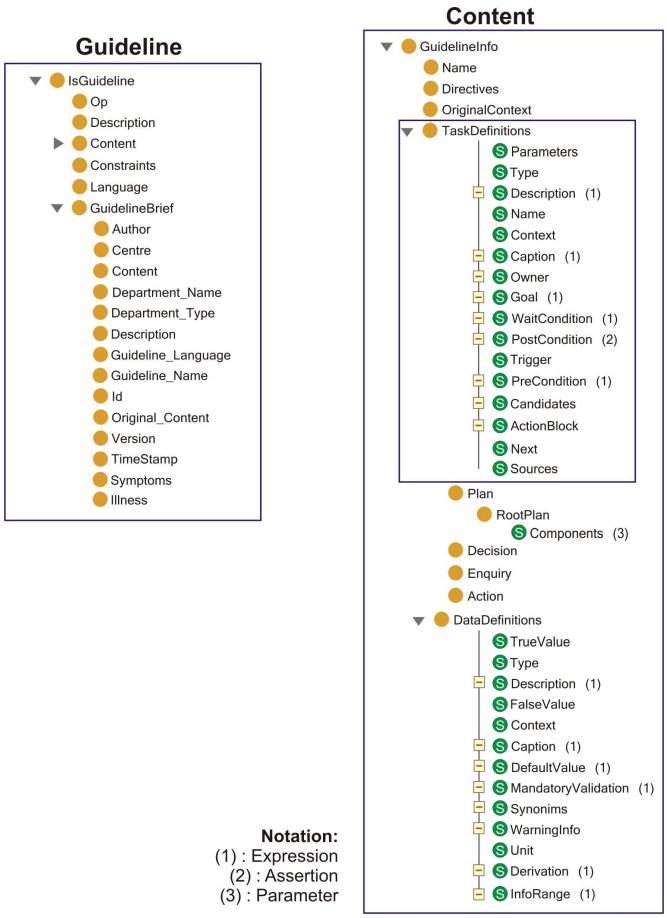


Figure 6.4: Clinical guideline ontology (*part 1*)

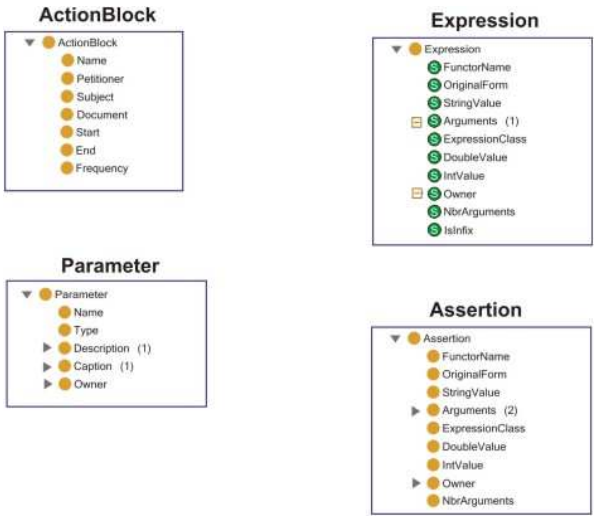


Figure 6.5: Clinical guideline ontology (*part 2*)

Using this ontology, the basic information contained in a CG can be shared across several entities, particularly this structure is used in the guideline agent in order to transmit the results of queries received by doctors. For instance, the following message shows the result for query for a specific guideline from a centre:

```
(=
     (all ?x
       (IsGuideline getGuideline
         (GuidelineBrief
           :centre ConsultoriLocalAlforja
           :department_name General-medicine
           :guideline_name miquelSDA_kidneyProblemsFictitious.xml))
     (set
       (GuidelineInfo
         :GuidelineInfoTaskDefinitions
           (set
             (TaskInfo
               :TaskInfoType 2
               :TaskInfoName DS2
               :TaskInfoNext (set A3))
             (TaskInfo
               :TaskInfoType 3
               :TaskInfoName ES2
               :TaskInfoNext (set A3))
             (TaskInfo
               :TaskInfoType 2
               :TaskInfoName DKidney_problems
               :TaskInfoNext (set Kidney_status))
             (TaskInfo
               :TaskInfoType 3
               :TaskInfoName EKidney_problems
               :TaskInfoNext (set Kidney_status))
             (TaskInfo
               :TaskInfoType 2
```

```
  :TaskInfoName Kidney_status
  :TaskInfoNext (set A4 A4 A4)
  :TaskInfoSources
    (set "low functionality" "low functionality"))
(TaskInfo
  :TaskInfoType 1
  :TaskInfoName A3
  :TaskInfoActionBlock
    (set
      (ActionBlock
        :Name "prescripts a non-pharmacological treatment"
        :Petitioner PC
        :Subject PC
        :Start "Tue Apr 29 12:46:22 CEST 2008"
        :End "Tue Apr 29 12:46:22 CEST 2008"
        :Frequency 0))
  :TaskInfoNext (set))
(TaskInfo
  :TaskInfoType 1
  :TaskInfoName A1
  :TaskInfoActionBlock
    (set
      (ActionBlock
        :Name "Prescribes assistive devices"
        :Petitioner PC
        :Subject Nurse
        :Start "Tue Apr 29 12:46:22 CEST 2008"
        :End "Tue Apr 29 12:46:22 CEST 2008"
        :Frequency 0))
  :TaskInfoNext (set A2))
(TaskInfo
  :TaskInfoType 1
  :TaskInfoName A4
  :TaskInfoActionBlock
    (set
      (ActionBlock
        :Name Send_message
        :Petitioner PC
        :Subject Patient
        :Document "Message to Patient"
        :Start "Tue Apr 29 12:46:22 CEST 2008"
        :End "Tue Apr 29 12:46:22 CEST 2008"
        :Frequency 0))
  :TaskInfoNext (set))
(TaskInfo
  :TaskInfoType 1
  :TaskInfoName A2
  :TaskInfoActionBlock
    (set
      (ActionBlock
        :Name "Prescribe nursing care"
        :Petitioner PC
        :Subject PC
        :Start "Tue Apr 29 12:46:22 CEST 2008"
        :End "Tue Apr 29 12:46:22 CEST 2008"
        :Frequency 0)
      (ActionBlock
        :Name "authorizes nursing care"
```

```
        :Petitioner FD
        :Subject FD
        :Start "Tue Apr 29 12:46:22 CEST 2008"
        :End "Tue Apr 29 12:46:22 CEST 2008"
        :Frequency 0)
      (ActionBlock
        :Name "perform an intravenous therapy"
        :Petitioner PC
        :Subject Nurse
        :Start "Tue Apr 29 12:46:22 CEST 2008"
        :End "Tue Apr 29 12:46:22 CEST 2008"
        :Frequency 0)
      (ActionBlock
        :Name "write a follow up report"
        :Petitioner Nurse
        :Subject Nurse
        :Start "Tue Apr 29 12:46:22 CEST 2008"
        :End "Tue Apr 29 12:46:22 CEST 2008"
        :Frequency 0))
  :TaskInfoNext (set)))))))
```

6.2.3 Guideline-based execution module

The module that implements the guideline-based execution engine is presented in the Fig. 6.6. The doctor agent has an instance of this module and accesses it through a set of API calls. The first call is to load a certain guideline, and a set of methods allows to know the current status of a patient over his treatment, query a requested data value, set a certain value, and trace all the steps followed in each step (*e.g.*, tasks accomplished, in progress, finished) as well as the decisions made.

The module allows a continuous enactment of a guideline, or "to freeze" the execution. The agent stores this information (current status of the execution and all values related to the execution of this guideline) into the patient's record.

The module uses the particular execution engines translating the data from/to each run-time engine. This approach allow to maintain the original representation of the CGs with a common set of methods.

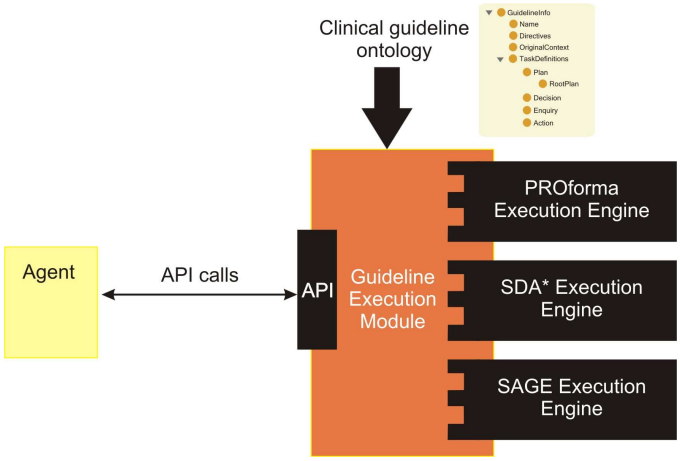


Figure 6.6: Guideline execution module

Moreover, the listener that defines all events that can be managed by the agent is the following:

```
public interface CGListener
{
/**
 * Thrown when an event related to time is detected
     * @param event Information related to the event
     */
    public void TimeEvent( CGEvent event );
    /**
     * Thrown when a document has been written
     * @param event Information related to the event
     */
    public void documentWritten(CGEvent event);
    /**
     * Thrown when an action has been reached
     * @param event Information related to the event
     */
```

```
    public void ActionReached( CGEvent event );
    /**
     * Thrown when a state has been reached
     * @param event Information related to the event
     */
    public void stateReached(CGEvent event);
    /**
     * Thrown when a jump is detected
     * @param event Information related to the event
     */
    public void jumpReached(CGEvent event);
    /**
     * Thrown when a decision has been reached
     * @param event Information related to the event
     */
    public void decisionReached(CGEvent event);
    /**
     * Thrown when the execution of the graph has been started. Then
     * the listener must choose which entry point wants.
     *
     * @param event Information related to the event
     */
    public void chooseEntryPoint(CGEvent event) throws CGException;
    /**
     * Thrown when the execution of the graph has been finished. This
     * means when the graph hasn't any route to continue, or when a
     * prefixed number of cycles has been done.
     *
     * @param event Information related to the event
     */
    public void graphFinished(CGEvent event);
    /**
     * Thrown when the execution of the graph has been finished because
     * there has produced so many cycles, and is considered that the
     * execution has entered into an "infinite" loop
     * @param event Information related to the event
     */
    public void tooMuchCycles(CGEvent event);
}
```

Internally, the guideline execution module should adapt the execution to the particularities of all language, and also, to the particularities of each execution engine. In our case, the SDA* and PRO*forma* engines are very similar and is easy to adapt the event-based enacting by translating and calling the appropriate methods. In the case of SAGE, the engine is under developing in the same way than the previous ones.

6.3 Ontological representation of medical and organizational knowledge

The scope of the ontology presented in this section covers all the relations established in the multi-agent system associated to a healthcare organisation. The ontology is divided in three main groups of concepts : *a*) description of all health care entities with their relations, *b*) linkage of semantic categories to the medical concepts, and *c*) representation of all medical terminology used by all partners [Isern, Sánchez & Moreno 2007*a*].

6.3.1 Description of health care entities

The *Agent* hierarchy of classes includes all main concepts related with the internal organisation of the multi-agent system (see Fig. 6.7). In this hierarchy there are *Departments*, *Patients*, *Practitioners*, *Medical centres* and *Services*. All these elements have internal relations, such as *Cardiology* `is-a` *Department* that `belongsTo` *Medical-center* (see Table 6.2 for a complete list of all non-taxonomic relations defined). More complex relations between doctors and services are also mapped, such as *Nurse* `belongsTo` *Department* because a nurse can be located in any department, or *Family_doctor* `belongsTo` (*General_medicine* $\cup$ *Emergency* $\cup$ *Paediatrics*) that means that an instance of family doctor could belong to any instance of these three departments.

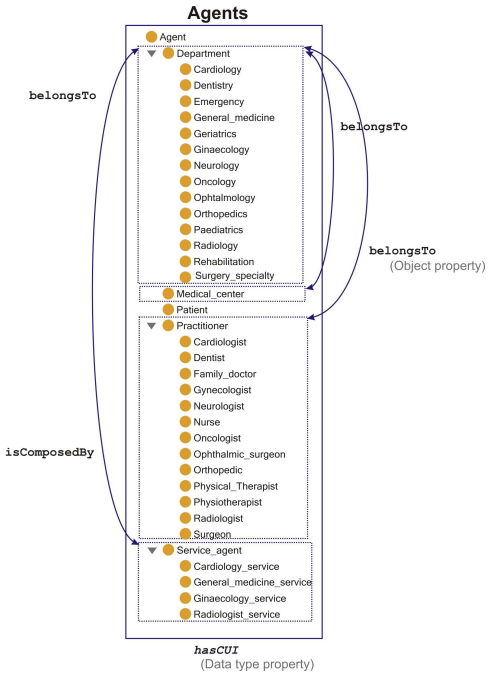


Figure 6.7: Medical ontology: organizational part

Relations between *Agent* subclasses are inspired in usual healthcare organisations. The inverse relations are also available to know which kind of doctors compose a department or which kind of services are located in a department or medical centre.

Although most of the departments are similar in medical centres, it is possible to represent different variations. In those cases, a specialisation of the ontology could be made. For instance, the oncology department is different in a hospital or in a primary attention centre that covers a part of a city or a set of villages. In these cases, two subclasses of the oncology department would be created. The parent class would keep all common features and the two siblings would contain the features or resources for each one.

6.3.2 Description of semantic types of medical concepts

The next set of classes concerns the different semantic types of the medical concepts. There are two main hierarchies, named *Entity* and *Event*, which were picked from UMLS Metathesaurus[3]. Currently, UMLS defines 135 different semantic types divided in two groups: meanings concerned with healthcare organisations or entities, and meanings related with events or activities in a daily care flow. Both hierarchies are organised as a taxonomy with `is-a` relations between concepts, such as *Disease_or_Syndrome* `is-a` *Pathologic_function.*

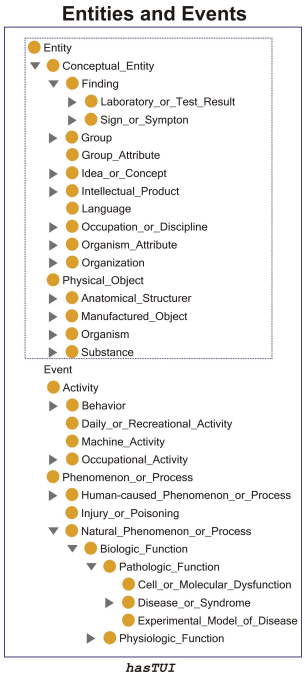


Figure 6.8: Medical ontology: semantic types

[3] *Unified Medical Language System* (UMLS) is a repository of the US National Library of Medicine that comprises the most widely used medical terminologies and nomenclatures, like MedLine, MeSH or LOINC. Web site: http://www.nlm.nih.gov/research/umls/ [last visit 11/11/2008].

All this information is used by agents to *know exactly which is the function of any required concept* and further connections with others. For instance, if a concept is a *Finding*, and a *Finding* `isResponsibilityOf` a *Practitioner*, the agent knows that a patient's finding should be given by a practitioner.

Fig. 6.8 shows the basic structure of this part of the ontology.

6.3.3 Medical domain terminology

The last part of the ontology represents the specific vocabulary used in clinical guidelines. It systematises all specific knowledge required in any guideline execution engine, divided in *Diseases*, *Procedures* and *Personal data*. It is necessary to define a set of relations between each concept and its identifier (Code Unique Identifier or CUI), its semantic type, which entity of the system is responsible of its accomplishment, and the produced result (*i.e.* if it as number, a Boolean, an enumerate or a complex object). Relations are bidirectional because it is interesting to know that the finding *Active_cancer* `isResponsibilityOf` a *Family_Doctor*, and the family doctor's responsibilities. Each agent can access the concepts related to its own domain and be aware of the consequences of the application of an action.

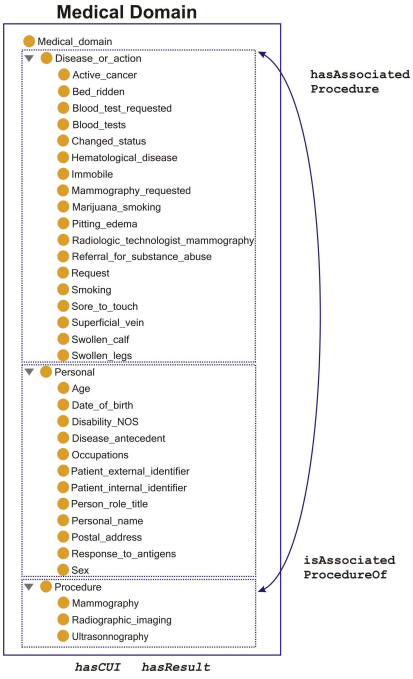


Figure 6.9: Medical ontology: medical terminology

Before to add a new CG to the system, a medical expert must update the ontology with the concepts, relations, actions, and effects, included in the CG.

If there is no information about a concept, the agent requests the doctor for making a decision. If a concept has more than one semantic type (during the execution of two different guidelines), the agent cannot follow an option because both directions are correct. In this

case, it is recommended to create another term, by searching the UMLS repository, that fits better with the ontology; the percentage of terms with more than one semantic meaning in UMLS is very low because the creators of this repository tried to avoid these kind of problems and facilitate its use by decision-support systems.

6.3.4 Combination of medical and organizational knowledge

As explained above, three main groups of concepts are defined in the medical ontology: *agent-based health care concepts*, *semantic types* of entities and events, and *medical concepts*. All the defined concepts are interrelated by taxonomic and non taxonomic relations. The former are based on `is-a` relations and are established by generalisation-specialisation of concepts. Some are picked from UMLS and others are picked from healthcare organisations. The second kind of relations is more difficult to establish, due to its semantic dependency. In fact, they are usually omitted in standard repositories [Ding et al. 2004].

Table 6.2: Object and data type properties defined in the *medical ontology*

Object Properties	*Description*
`belongsTo`	Any instance of a class that belongs to another
`hasAssociatedProcedure`	A medical concept has an associated procedure. It is used by doctors to simplify a search (from UMLS)
`hasResponsible`	Establishes the responsibility of a medical concept that has to be performed by a healthcare party
`hasSemanticType`	Functional property to specify the semantic type of a concept
`isAssociatedProcedureOf`	Inverse of `hasAssociatedProcedure`
`isComposedBy`	If an instance $a \in A$ `belongsTo` $b \in B$ then, $b \in B$ `isComposedBy` $a \in A$. It is not just the the inverse because the first relation is $1 - N$ and the second is $M - N$
`isResponsibleOf`	Inverse of `hasResponsible`
Data type Properties	*Description*
`hasCUI`	Value of the CUI (*Code Unique Identifier*) (from UMLS)
`hasDescription`	Concept definition provided from UMLS (when it is available)
`hasResult`	Type of output of an element (action or data concept)
`hasResultBoolean`	Sub class of `hasResult` that sets a Boolean as output
`hasResultInteger`	Sub class of `hasResult` that sets an Integer as output
`hasResultString`	Sub class of `hasResult` that sets a String as output
`hasResultEnumerate`	Sub class of `hasResult` that sets an enumerate as output
`hasResultComplex`	Sub class of `hasResult` that sets a complex element formed by one or more simple results (concepts) as output
`hasTUI`	In UMLS, semantic types are labelled with a *Type Unique Identifier*

By analysing the information required in the execution of a clinical guideline, a set of relations were defined (see Fig. 6.10). They are shown in Table 6.2, along with their type and description. When a new CG is added, all new concepts should be added and all required relationships between concepts should be established.

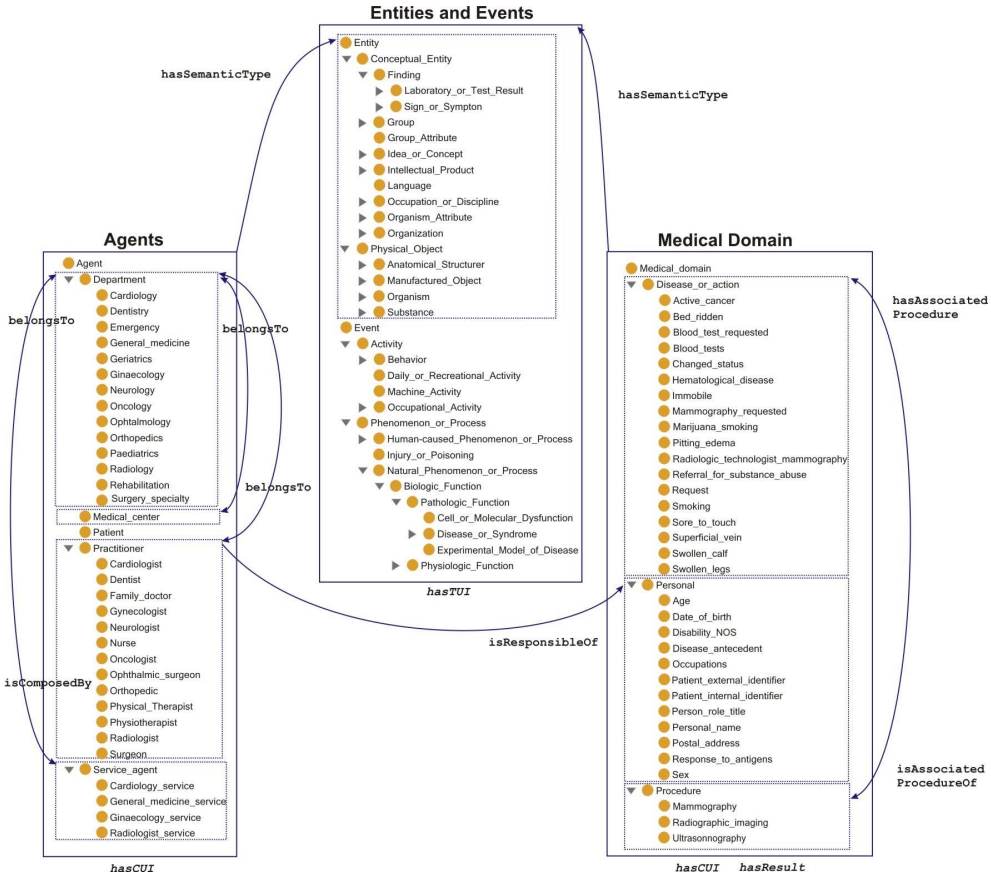


Figure 6.10: Medical ontology with relations between all parts

6.4 Ontology-driven execution of clinical guidelines

The procedural knowledge embedded within clinical guidelines should be complemented with a representation of declarative knowledge that allows to know exactly the semantic meaning of all elements, and hence, to implement a guideline-based execution engine to follow step-by-step all the stages. The previous section has shown the basic pieces to consider in any guideline: decisions, actions, and inputs of data. The CG consists in a combination of these elements with labelled transitions between them: temporal constraints.

The execution of a CG involves different kinds of actors, with different skills and constraints. In the case of HECASE2 the execution of a CG is a semi-supervised procedure handled by the doctor. In the following, all the procedures followed by agents, the decisions made by the system, and the information requested to the users (patients and practitioners) will explained.

6.4.1 Retrieving the appropriate clinical guideline

The first task to accomplish during the enactment is getting the requested CG to the practitioner when required. HECASE2 has a particular agent called guideline agent (GA) that stores a collection of CGs of a medical centre. These CGs are annotated with information about the department where they belong, the author, the version, the date of inclusion into the repository, and the names of the owner medical centre as well as of the clinical guideline itself. These attributes are used to filter the requested CGs and, at the same time, maintain a versioning of guidelines.

When the doctor begins a medical visit with a patient, the DRA pro-actively performs a set of tasks, such as getting the patient's medical record, getting the information of the medical visit, and getting the list of available CGs. The patient's data is collected through the *medical record agent* (MRA), and the list of available CGs from the GA. If there is no previous information about the patient, the doctor receives an empty medical record. Otherwise, he receives the personal details of the patient (*e.g.*, name, address, date of birth, phone, allergies), the collection of results of past medical visits, and the information of current ongoing CGs. If the patient is currently following a CG, the name of this guideline is highlighted in the list of guidelines, but the doctor can decide to change it. After selecting the CG to follow, the DRA requests the content of this CG to the GA. The DRA receives the CG, and establishes the current status of the patient.

6.4.2 Execution of a clinical guideline

As shown in previous sections, the combination of a knowledge base and an agent-based system that exploits that knowledge can be interesting to achieve flexibility and re-usability. In order to illustrate how these elements have been integrated, in this section we explain the procedure followed in a CG adopted from the National Guideline Clearinghouse and coded in PRO*forma* [Sutton & Fox 2003] which is intended to diagnose and treat *Gastrointestinal Cancer* (GC). The CG was created using Tallis and it is depicted in Fig. 6.11. An screenshot of its inclusion in the HECASE2 platform and its use through the DRA is shown in Fig. 6.12.

First of all, the doctor selects the GC guideline from the repository (through his Doctor Agent (DRA) and the Guideline Agent (GA) of the department as explained above) (see

Fig. 4.16 for the detailed explanation of the agent-based architecture). The DRA receives the CG which is graphically presented to the doctor.

The DRA indicates to its guideline execution module (see Section 6.2.3) that this guideline must be loaded to be executed, and the DRA is ready to receive events from this module. First of all, the DRA needs to determine the exact entry point within the CG to begin the execution (usually, there is only one entry point), and gets the required data to begin the enactment. When the DRA receives these data, it collects all past values from the patient's health record, and puts these values automatically. The interface described previously (Section 6.2.3) is the *gateway* between the DRA and the execution module to follow the events raised during this execution. Particularly, two important methods need a especial mention: `decisionReached` and `actionReached`. The first case, the module informs to its listener that a decision has been reached; the agent can check the information related to this decision, such as the logical condition reached. In this case, the DRA informs to the medical expert through his interface agent and it does not need any other step. The second case is more complex than the first. In this case, the method informs that an action should be performed. In this case, the DRA must search the entity able to execute this action. This information is collected through the medical ontology.

The DRA knows always the current state of execution, and there is an special method to update this pointer (`jumpReached`). With this information, the DRA can access to this node and check all the information. In the case of an state in the SDA*, or enquiry in PRO*forma*, or context in SAGE, the DRA needs to find a value (or values) for values, for instance, a finding, the age of the patient, etc. These data is collected in the same way than the actions explained previously. Each item to be collected is found into the medical ontology, and the entity able to perform or to provide this item is contacted.

Coming back with the execution of the CG, the first step is to evaluate the importance of the disease. As a result of the evaluation, it collects information about the patient and an invasive test (*Biopsy*). The DRA analyses the CG and observes that the first enquiry is composed by six parameters. For each one, the DRA asks the Ontology Agent (OA) to know more details. The OA replies with the information found in the *Medical Ontology*. In this case, the parameter *Age* is included in the category of *Personal* that can be found in the medical record. The DRA requests the MRA that value. Other required parameters like *Pain site*, *Weight loss*, *Pain time* and *Smoking*, are *Findings* that the doctor can evaluate and set if the record is not found in the patient's history. The ontology also contains information about each element, like the resulting format output and the allowed values by using data type properties (see Table 6.2). For instance, in the case of *Pain time*, due to its nature, the allowed data values are *short*, *moderate*, and *long*. Finally, the last value to consider is the result of the *Biopsy*.

A *Biopsy* is an invasive *Diagnostic_Procedure* that `isResponsibleOf` a *Surgery_specialty*. In that case, if the biopsy has not been performed previously, the DRA is able to look for a surgeon (*Practitioner* that `belongsTo` a *Surgery_specialty* department) and book a meeting for the patient (at first, it looks for available surgeons, and then, it begins a *contract net* negotiation with them). If agreed, the enactment is stopped until that result is received. In a future medical visit, the doctor will have the result of the biopsy and he will be able to perform a first diagnosis: if the biopsy is negative, the patient will follow a plan for gastroin-

testinal treatment of a *Peptic_ulcer.* Otherwise, the patient suffers from cancer and should be treated for that disease.

In the case of cancer, there is a final step to be performed before referring the patient to a surgeon: to check if the patient is elderly or not. In the former case, the patient cannot be hospitalised and should be treated with symptomatic treatments. In the latter case, there are two possible plans to follow, a chemical treatment such as *Chemotherapy*, or a *Surgery*. The decision is taken by the surgeon in a further medical visit.

As shown through the example, the CG provides the general care flow to follow (declarative and explicit knowledge) and the ontology provides semantic information about all concepts or actions to be performed. Detailed information allows to represent relations between all entities and to collect all required data by agents in order to know which decision to take. When the doctor agent considers the term *Biopsy* it does not know if that concept is a procedure or a finding or any other kind of element, and it does not know the existing relations of that concept either. The ontology allows to correlate all elements present in the CG and know exactly all the details. This information is not included in CGs because it depends on the specific scenario where the CG should run or the specific organisation.

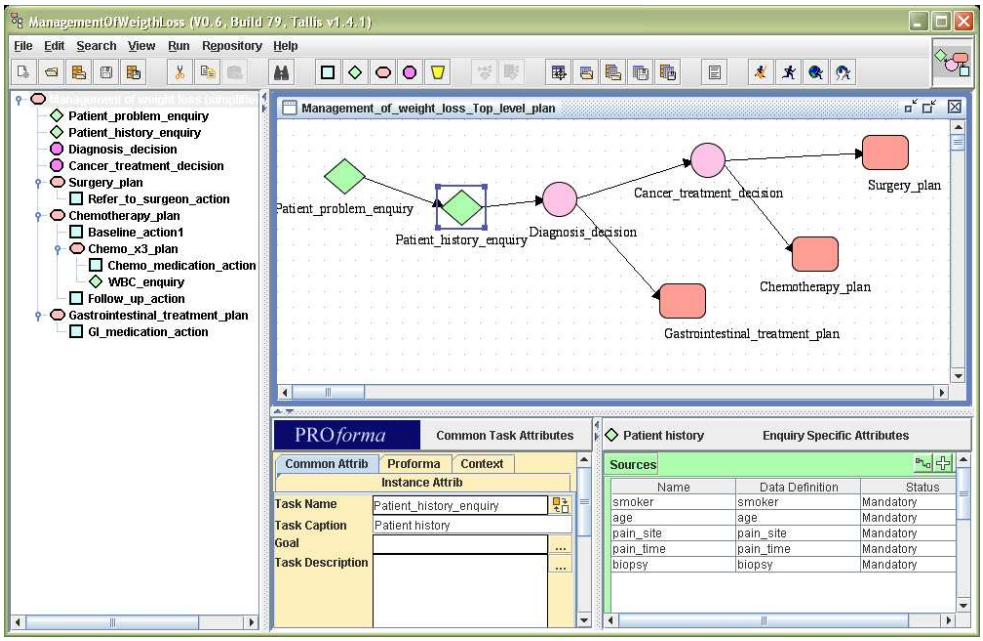


Figure 6.11: Case study guideline edited using the PRO*forma* composer tool (Tallis)

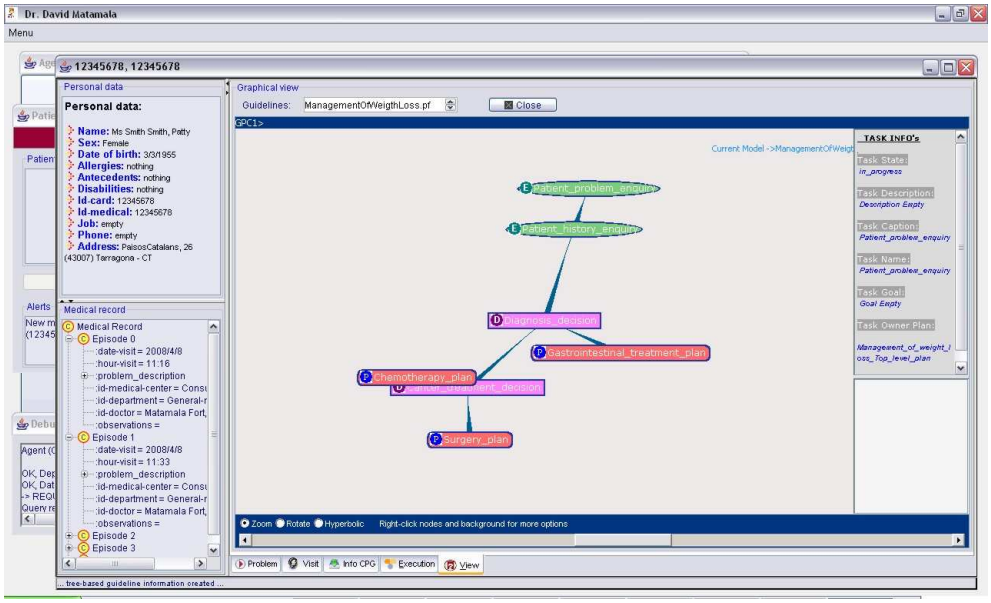


Figure 6.12: Case study guideline viewed through the DRA interface

6.5 Conclusions

The inclusion of a several ontologies in the multi-agent system HECASE2 has been discussed. As shown in the Section 6.1, the use of ontologies in the medical domain is increasing and offers some advantages such as making domain assumptions explicit, separating domain knowledge from operational knowledge, and sharing a consistent understanding of what information means.

The first ontology concerns the management and representation of clinical guidelines (*clinical guideline ontology*). In this case, a generic module to execute clinical guidelines using this representation has been designed. In fact, the ontology is a high level representation used to transmit the basic information about guidelines and its execution, but the module includes different execution engines to execute the guideline in the original form. In this way, a translation of all guidelines to the new representation is not required. Nowadays, the implementation and validation of this part is an ongoing task.

On the other hand, the *medico-organizational ontology* brings the following advantages to the guideline-based execution system: *a*) to identify the required actors that are able to accomplish an action and to know the source/details of an item, *b*) to adapt the execution framework to the particular casuistry of any healthcare organisation without modifying the MAS implementation or the guideline, and *c*) to provide an application independent context. Thus, by changing the ontology and its relations, the execution procedure also changes.

Note that the only issue that should be addressed is the manual definition of the appropriate task ontology. This question usually requires the intervention of a domain expert, but UMLS provides a large corpus of concepts and relations that can be easily reused, and the ontology creation process could be automatised by collecting automatically the required information when a new CG was added to the repository.

Chapter 7

Applications of agent-based execution of guidelines

HECASE2 has not been deployed in a real healthcare institution. However, the ideas underlying this system are being actually used in two research projects, which aim at building clinical informatics systems. These systems, which are based on agents that follow clinical guidelines, will be validated in real environments.

In particular, the two projects related to this work are K4Care (EU-funded) and Hygia (Spanish-funded). Basically, the first project adopts the idea of having a collection of agents representing actors of the system (care givers and patients) that allow collecting and giving the appropriate data to the appropriate point of care. The coordination of tasks between those actors during the enactment of a CG is a basic pillar of the project. The second project adopts part of the HECASE2 agent architecture for a particular doctor that wants to apply a specific clinical guideline over a patient; in this case, the system also aims to study the adherence of practitioners to a clinical guideline. The system will monitor all the decisions made by the doctor in order to identify problems and propose changes. Both projects share common elements like the representation of the medical knowledge through an ontology, the addition of facilities to access an electronic health record, and the use of a formal language to represent clinical guidelines.

Both projects are described in the following sections. At the end of the chapter, the main ideas of HECASE2 adopted by these works are summarised.

7.1 Agent-based Execution of Home Care Individual Intervention Plans

The basic ideas about agent-based provision of health care services developed in the HECASE2 system are being directly used in the EU-funded project K4Care - Knowledge Based Home Care eServices for an Ageing Europe - (IST-2004-026968)[1] [Campana et al. 2008].

The *K4Care* project is studying the feasibility of using Information and Communication Technologies to improve the management of Home Care (HC) of patients that require assistance at home. It is coordinated by the University Rovira i Virgili, and includes the partners listed in Table 7.1.

Type	*Partner*	*Location*
Academic	Universitat Rovira i Virgili (URV)	Tarragona, Spain
Academic	Computer and Automation Research Institute of the Hungarian Academy of Sciences (SZTAKI)	Budapest, Hungary
Academic	Czech Technical University (CTU)	Prague, Czech Republic
Health care	Azienda Sanitaria Locale RM B	Rome, Italy
Health care	Amministrazione Comunale di Pollenza	Pollenza, Italy
Health care	Universitá degli Studi di Perugia, Department of Geriatrics (UNIPG)	Perugia, Italy
Health care	Fundazione Santa Lucia	Rome, Italy
Health care	General University Hospital in Prague	Prague, Czech Republic
Health care	'Ana Aslan' International Academy of Ageing (ANA)	Bucharest, Romania
Health care	The Research Institute for the Care of Elderly (RICE)	Bath, United Kingdom
R+D	Telecom Italia	Milan, Italy
Management	European Research and Project Office GmbH (Eurice)	Saarbruecken, Germany

Table 7.1: K4Care partners

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 [Campana et al. 2008]. The healthcare of the HCP is particularly complex because of the growing number of patients in such circumstances, and also because of the great amount of resources required to guarantee a quality long-term assistance. The project has developed a platform to manage the information needed to guarantee an ICT Home Care service, which includes: an integration with ICT whilst ensuring private and customized data access; the use of ontologies to define the profile of accessing subjects; a mechanism to combine and refine the ontologies to personalise the system; the incorporation of know-how from geriatric clinical guidelines (named *Formal Intervention Plans* [NGC 2007] - FIPs); the generation of FIPs from the personalised healthcare treatments; the extraction

[1] For more information, please visit http://www.k4care.net [last visit: 01/07/2008].

of evidence from real patients and its integration with published evidence derived from randomised clinical trials; and finally, the configuration of a knowledge-based decision support tool that can supply e-Services to all subjects involved in the home care model. The last item, the decision support tool, has been designed using a Web-accessible multi-agent platform.

7.1.1 K4Care Model

The *K4Care Model* defines the basic elements supported by the system and their relationships [Campana et al. 2006, Hajnal et al. 2007, Isern, Millan, Moreno, Pedone & Varga 2008*a*]. In the model, services are distributed by local health units and integrated with the social services of municipalities, and eventually with other organizations of care or social support. The model is aimed at providing the patient with the necessary sanitary and social support to be treated at home. To accomplish this duty, the *K4Care Model* gives priority to the support of the HCP, his relatives and Family Doctors (FD) as well. Because of its aim, the model is represented by a modular structure that can be adapted to different local opportunities and needs. The success of this model is directly related to the levels of *efficacy*, *effectiveness* and *best practice* of the healthcare services the model is able to support.

Basically, the *K4Care Model* is based on a nuclear structure (HCNS) which comprises the minimum number of common elements needed to provide a basic HC service. The HCNS can be extended with an optional number of *accessory services* (HCAS) which will respond to specialized cares, specific needs, opportunities, means, etc. The distinction between the HCNS and the complementary HCASs should be interpreted as a way of introducing flexibility and adaptability in the *K4Care Model*. In more detail, each one of the HC structures (*i.e.,* HCNS and HCASs) has the same components: a) *Actors* are all the sort of human figures included in the HC structure; b) *Professional Actions* and *Liabilities* define the tasks that each actor performs to provide a service within the HC structure; c) *Services* provided by the HC structure for the care of the HCP; d) *Procedures* are the chains of events that lead an actor in performing actions to provide services; and e) *Information* contained in documents required and produced by the actors to provide services in the HC structure.

As new HCASs are incorporated to the *K4Care Model*, new actors, actions, services, procedures and documents enter to be part of the extended model. In this way, the *K4Care Model* is compatible both with the current situation in the European countries where the international, national, and regional laws define different HC systems, and also with the forthcoming expected situation in which a European model for HC will be decided.

7.1.2 K4Care Architecture

The architecture, shown in Fig. 7.1, is divided in three main modules: the *Knowledge Layer*, the *Data Abstraction Layer*, and the *K4Care agent-based platform* [Campana et al. 2008].

The *Knowledge Layer* includes all the data sources required by the platform. It contains an Electronic Health Record (EHR) subsystem that stores patient records with personal information, medical visits and ongoing treatments. In addition, this layer contains all declarative and procedural knowledge used in the system (this topic is discussed later, see Section 7.1.2.2).

The *Data Abstraction Layer* (DAL) provides Java-based methods that allow the K4Care platform entities to retrieve the data and knowledge they need to perform their tasks. This

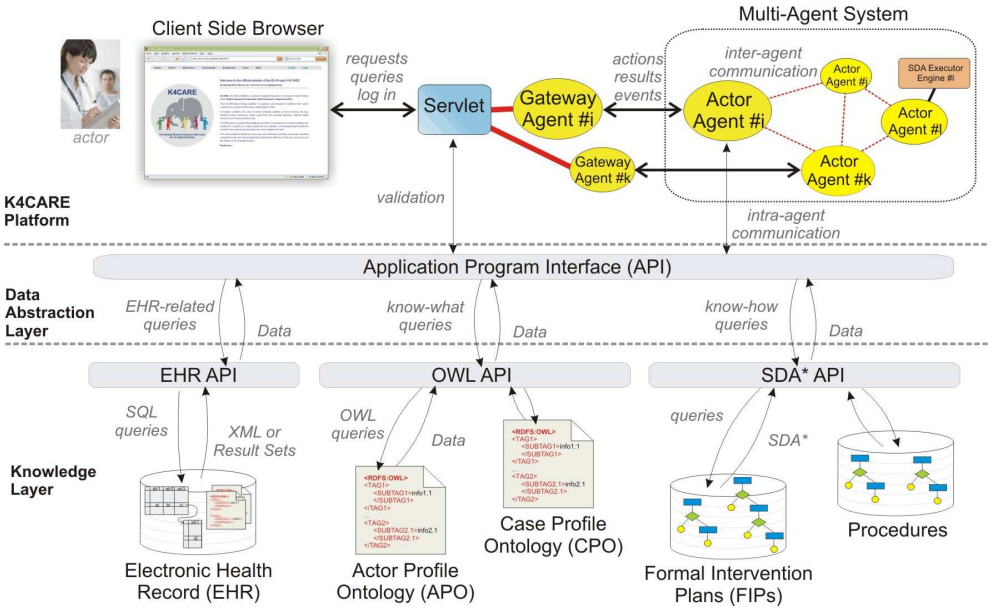


Figure 7.1: K4Care Platform Architecture [Campana et al. 2008]

layer offers a wide set of high-level queries that provide transparency between the data (knowledge) and its use (platform) [Batet et al. 2008].

The *K4Care platform* is a web-based application with a client side (a Web browser) and a server side (a servlet). Each actor interacts with the system through a Web browser and is represented in the system by a permanent agent (*Actor Agents* in Fig. 7.1) that knows all details about his roles, permissions, pending results, pending actions, and that manages all queries and requests coming from the user or other agents. In order to exchange information between the agents and the actors there is an intermediate bridge constituted by a servlet and a *Gateway Agent* (GA). The servlet is connected with the browser user session. It creates a GA each time that an actor logs in the system, whose mission is to keep a one-to-one connection with the corresponding permanent agent.

7.1.2.1 Actors

In HC there are several people interacting: patients, relatives, physicians, social assistants, nurses, rehabilitation professionals, informal care givers, citizens, social organisms, etc. These individuals are the members of three different *groups of HC actors*: a) the *patient*; b) the *stable members* of HCNS (the family doctor, the physician in charge of HC, the head nurse, the nurse, and the social worker); and c) the *additional care givers* (see Fig. 7.2).

The family doctor, the physician in charge of HC, the head nurse, and the social worker join in a temporary structure called *Evaluation Unit*, which aims to assess the patient's problems and needs, to decide a particular treatment and to monitor its progress. The patient is located in the centre of the HCNS of the *K4Care Model* (see Fig. 7.2), and the rest of the groups are organised around it as a symbol of a patient-oriented HC Model.

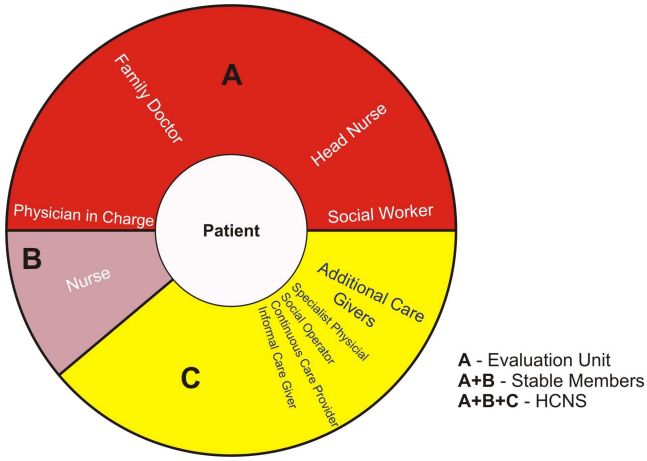


Figure 7.2: Actors in the Home Care Nuclear Structure (HCNS)

7.1.2.2 Representation of medical knowledge

There are two kinds of knowledge to be represented in the system: *declarative* and *procedural*. The former contains the information on the basic elements of the *K4Care Model* and the organisational relationships between the system actors. The later is concerned with the representation of the sequences of actions involved in the provision of a service or the treatment of a patient. Apart from all this knowledge, all data concerning patients are stored in the EHR, which is consulted by actors as needed in the different stages of the patients' treatment.

Declarative Knowledge *Ontologies*, as a set of concepts, properties and relations, constitute a feasible paradigm to represent the declarative knowledge used in the system [Fensel 2001, Pisanelli 2004, Pisanelli et al. 2004]. There are two basic ontologies in *K4Care*, which have been defined *ad hoc* for this project. The first ontology, named *Actor Profile Ontology* (APO), details the basic elements of the *K4Care HC model* (actors, actions, services, procedures, documents) and the relationships between them (*e.g.*, which actions may be performed by each kind of actor, or which document is associated to each action). The second one, named *Case Profile Ontology* (CPO), stores all the medical terms related to HC (diseases, syndromes, signs, symptoms, assessment tests, clinical interventions, laboratory analysis, social issues) and the relationships between them (*e.g.*, the diseases included in a certain syndrome, or the symptoms of a disease). Agents are able to reason using the knowledge contained in this ontology, which can be considered as a bridge between the concepts that agents are able to recognize (conditions, diseases) and how actors have to act on those situations (associated interventions). Taxonomic and non taxonomic relations between concepts have been defined in order to allow structuring the information in an appropriate way to answer high level queries about that data. Both ontologies are represented in OWL [McGuinness & Harmelen 2004].

Procedural Knowledge On the other side, *procedural* knowledge that codifies complex medical tasks is required to define the set of available actions performed by all actors in the platform [Batet et al. 2008]. Medical experts have defined a set of procedures related to chronic

diseases that are stored and managed by actors. That knowledge has been coded using a flowchart-based representation called SDA* [Kamisalic et al. 2007, Riaño 2007].

The most basic components of SDA* structures are the domain variables, which can be of three types (see Fig. 7.3):

- *State variables*, that represent terms that are useful to determine the condition of the patient at a certain stage (*e.g.*, a patient with chronic kidney disease, who has high blood pressure).
- *Decision variables*, which are required by medical experts to choose among alternative medical, surgical, clinical or management actions within a treatment (*e.g.*, a treatment may follow different paths depending on the blood pressure of the patient). A decision contains a set of logical conditions that should be evaluated before proceeding. Some of the values required in these conditions can be retrieved from the patient's record (*e.g.*, the last value of the patient's blood pressure), but there could be findings to be entered by the practitioner during the medical visit (*e.g.*, consider a secondary cause of hypertension).
- *Action variables*, which represent the medical, surgical, clinical or management actions that may appear within a treatment (*e.g.*, define or change a drug therapy).

Fig. 7.3 shows an example of SDA* structure for the treatment of hypertension. States are represented as circles, decision as rhombus and actions as rectangles. The patient may initially be in any of four possible states (*e.g.*, a diabetic patient with a high level of blood pressure -BP). After an initial assessment, and if there is not any secondary cause of the hypertension, the patient is recommended some life style modifications and possibly treated with a drug therapy. The blood pressure is controlled periodically. If the initial drug therapy fails, then the physician should change the treatment until the blood pressure is stabilised in a safe range.

The SDA* formalism is used in *K4Care* to represent three kinds of elements:

- *Procedures*: descriptions of the steps to be taken within the *K4Care platform* to provide one of the HC services.
- *Formal Intervention Plans* (FIP): general descriptions defined by healthcare organisations such as the National Guideline Clearinghouse ([NGC 2007]) used to represent health care procedures to assist patients suffering from one or several ailments or diseases.
- *Individual Intervention Plans* (IIP): descriptions of the specific treatment that has to be provided to a particular patient.

7.1.2.3 Individual intervention plans

In the *K4Care platform*, FIPs may be related to a syndrome (*e.g.*, cognitive impairment), a symptom (*e.g.*, abdominal pain) or a disease (*e.g.*, dementia). A patient may suffer from several of these conditions; thus, the recommendations made by several FIPs must be taken into account to construct an *Individual Intervention Plan* (IIP). The IIP indicates the personalised care actions to be applied on a specific patient. Both FIPs and IIPs are represented using the SDA* formalism (see Fig. 7.3).

The creation and management of IIPs follow a complex procedure which is controlled by the *Evaluation Unit* (EU), which includes four actors: the physician in charge of the Home Care unit (PC), a family doctor (FD), a social worker (SW) and the head nurse (HN).

After the patient is admitted in the Home Care service, he is assigned a personalised EU. The first step in the patient's care is the performance of a comprehensive assessment (CA), which includes a multi-dimensional evaluation (made by all members of the EU, filling a

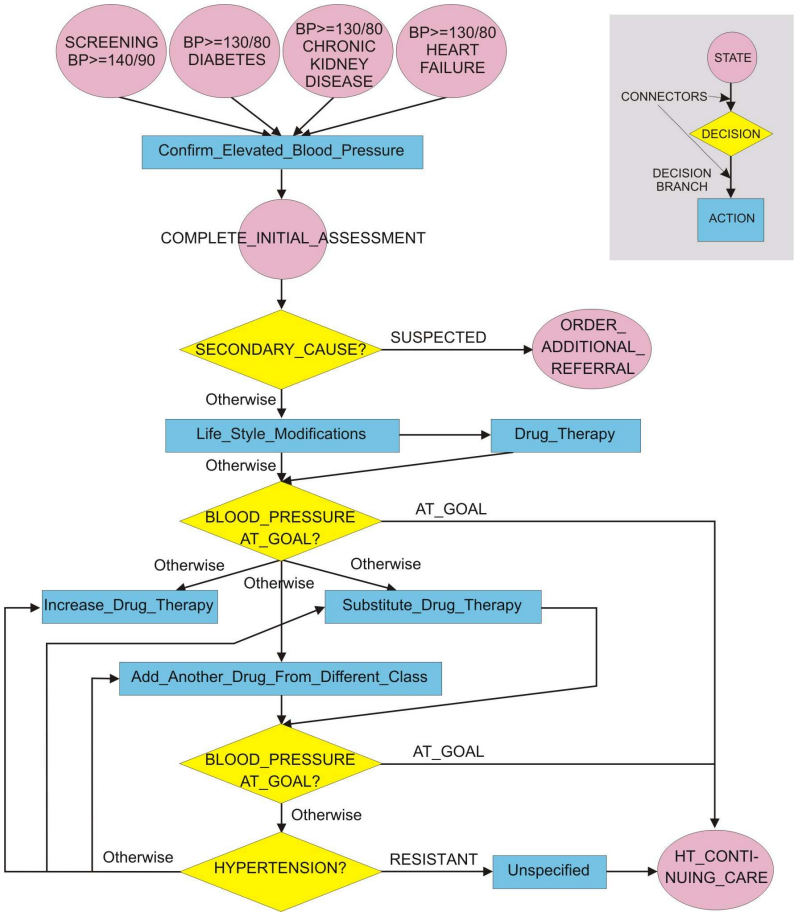


Figure 7.3: SDA* flowchart for the treatment of hypertension [Isern, Millan, Moreno, Pedone & Varga 2008*c*]

set of internationally standardised scales), a clinical assessment and a physical examination (which may be performed either by the PC or the FD) and a social needs and social network assessment (performed by the SW). Once all the results of the CA are available, the EU members analyze them and determine the syndromes, symptoms and diseases of the patient. The platform retrieves automatically the FIPs related to these conditions and the EU members then use a SDA* graphical editor, which is embedded in the web interface, to combine and personalise the relevant sections of these FIPs in order to build the specific IIP for that particular 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. The IIP usually contains follow-up actions in which the state of the patient is checked and, if necessary, the IIP may be cancelled or changed. Once the IIP is ready, it is saved in the Electronic Health Record of the patient.

7.1.3 Agent-based execution of IIPs

The agents of the multi-agent system of the *K4Care Platform* embed all the system logic by representing the actors involved in the delivery of services. 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 management of pending actions, are performed by agents without the intervention of human users [Isern, Millan, Moreno, Pedone & Varga 2008*b*, Isern, Millan, Moreno, Pedone & Varga 2008*c*]. There are other actions, such as the confirmation of the formation of an EU or the evaluation of some results received from laboratories, which require the user validation.

Basically the *K4Care* upper layer (Fig. 7.1) is composed by the following elements:

- *Actor agents* (AAs) represent practitioners and patients, and use the Data Abstraction Layer methods in order to access the data they need. The AAs are permanent in the system and monitor all pending and done tasks.
- A web interface, through which human users can access the system.
- One servlet and several *Gateway Agents* (GA) that allow exchanging information between the MAS and the web-based application. GAs are created dynamically when an actor logs into the system.
- *SDA* Executor Agent* (SDA-E) that allows to enact a care plan for a patient and recommends the next step to follow according to his current state. The SDA-E is also created dynamically by an AA in order to enact a SDA* structure corresponding to an IIP or to a management procedure. SDA-E agents have been designed to be able to manage concurrently several SDA-based structures.

7.1.3.1 Preliminary aspects

The enactment of an IIP is a complex task that requires the interaction of agents and humans (see Fig. 7.4). Technically, it is one of the services provided by the K4Care platform. This service may be requested by the PC. The person who is responsible of the management of the execution of an IIP is usually the HN. Note that each Home Care unit (modelled by a K4Care platform) only has one PC and one HN, but it may have many FDs, SWs and other kinds of actors (nurses, specialist physicians, informal care givers, etc.).

The PC logs into the K4Care platform through the web interface and requests the execution of the IIP of a particular Home Care Patient (HCP), David Jones. This information is received by the servlet, which forwards it to the GA associated to the PC (the GA is dynamically created when the user logs into the platform, and remains alive as long as the working session is active). The GA transmits the request to his corresponding permanent actor agent (step 1). As in any other service, the actor who is able to perform it (in this case, the HN) is stored in the APO, and the AA gets this information (steps 2-3). The AA of the PC contacts with the AA of the HN, who stores the request to begin the execution of an IIP (step 4).

When the HN logs into the system, a GA is automatically created. This GA asks to the AA of the HN which are the pending actions (actions that have been requested by other users while the HN was not logged in the system, or that were pending from previous sessions), and sends this information to the web interface through the servlet (step 5).

At some moment the HN will select that pending action (the execution of the IIP of Mr. Jones, step 6). The AA of the HN will then dynamically create a SDA-E agent, which will receive the identifier of the HCP (step 7).

The SDA-E retrieves the corresponding SDA representation of the IIP from the EHR of the patient by calling the appropriate method of the DAL (steps 7a-7b). At this point the SDA-E is ready to start sending to the AA of the HN the information contained in each of the elements of the IIP (i.e. states, decisions and actions).

The AA of the HN will manage the execution of the IIP. It will obtain each of the elements of the SDA* structure, one by one, through requests to the SDA-E agent. Thus, the AA of the HN will receive, after each request, either a state, a decision to be taken or an action to be performed.

7.1.3.2 Agent-based enactment of an IIP

States are descriptions of the expected clinical situation of a patient at a certain point of his care. In the present version of the system, the AA of the HN does not process states in any way. The idea is that the HN should check whether the expected state (retrieved from the IIP) matches with the present clinical condition of the patient. If that is the case, the treatment is giving the expected outcomes and the HN may proceed with the execution of the IIP; otherwise, the EU should evaluate the difference between the intended and actual states and decide whether to continue with the execution of the IIP, to modify it or even to cancel it and make a brand new one.

When the SDA-E finds a decision, the logical expression contained in it should be evaluated, in order to decide which of the decision branches should be followed. The SDA-E collects all available data present in the EHR of the patient about the variables included in the expressions. If some data are not available, the AA of the HN sends the decision to the graphical interface of the HN, so that he can decide if the condition holds or not. Finally, the last element that the SDA-E agent may send to the AA of the HN is an action (step 8). Concretely, the SDA-E sends a task, containing an action identifier and a type of actor as subject (*e.g.*, action S6.WRITE_SOCIAL_REPORT, subject SW). The execution of a care action (in the real world) is confirmed by filling (in the platform) a document[2], which is stored in the EHR of the patient.

[2]The designed model includes 84 specific care actions that may be performed by 10 types of actors [Campana et al. 2008]. There are 43 documents defined in the K4Care model (some documents are general enough to be usable to reflect the performance of different actions).

The task description contains the subject which is the type of agent which is expected to execute the service or the action. When the AA of the HN receives the task, its first mission is to assign the action to a specific person (which will be represented by a specific agent in the system). The agent will be assigned dynamically when the task is to be executed by selecting the most suitable agent through a negotiation process (*e.g.*, applying the well-known Contract Net protocol). The advantage of this solution is that we exploit the benefits of the agent design approach; however, Home Care centres usually like to have a tight control of resources, and would like to know in advance who does what and when.

Once the AA of the HN knows the person assigned to the action, it sends a message to the AA of that person requesting the performance of the action (indicating the action identifier and the patient to whom it must be applied, step 9a). When that person logs into the system, he will see this request in the list of pending actions (step 9b). When he selects this action, a message will be sent by his GA to his AA (step 10). Then, the AA will retrieve, using the appropriate method of the DAL, the document to be filled to reflect the performance of the action (steps 11 and 12). The AA sends the document to the web interface (through its associated GA and the servlet), and the user fills it (steps 13 and 14). Then the AA stores the filled document in the EHR of the patient (step 15) and sends a message to the AA of the HN to indicate that the action has been performed (step 16). After that, the AA of the HN would request to the SDA-E the next element (state, decision or action) of the IIP.

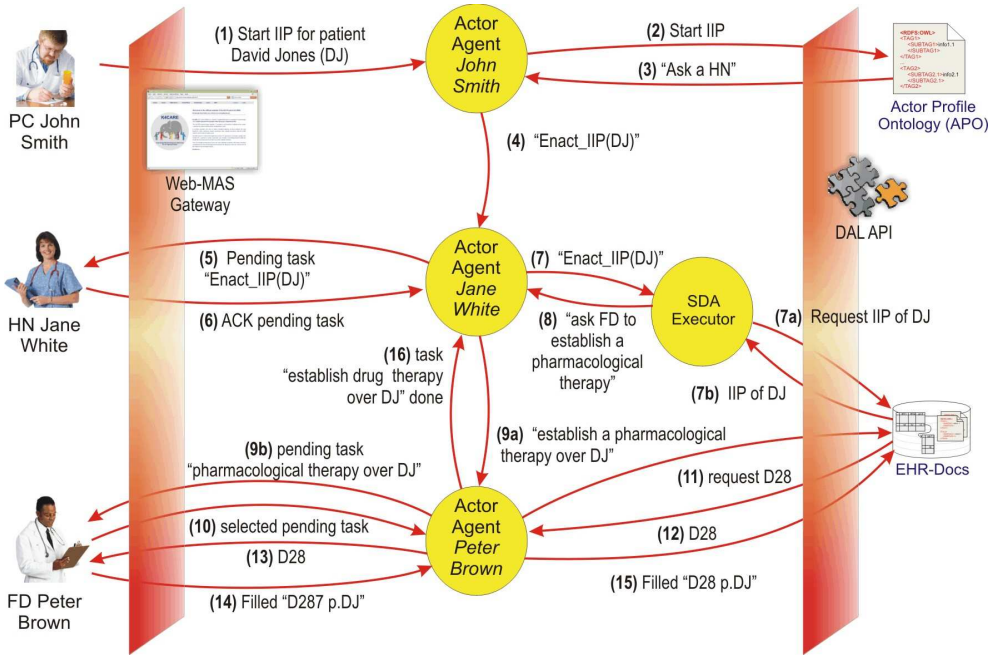


Figure 7.4: Agent-based execution of an Individual Intervention Plan

7.2 Hygia - Agent-based Execution of Care Pathways

The *Hygia* project proposes the use of intelligent systems in the process of acquiring, formalizing, adapting, using and assessing knowledge models that are equivalent to clinical guidelines [Alonso et al. 2008]. Those knowledge models may be employed by a distributed and open computer system to ease the process of decision making, offering the possibility of practising medicine in the context of the new information society.

This project[3] is coordinated by the University Rovira i Virgili, and includes the partners listed in Table 7.2.

Type	*Partner*	*Location*
Academic	Universitat Rovira i Virgili (URV)	Tarragona, Spain
Academic	Universidad de Santiago de Compostela (USC)	Santiago de Compostela, Spain
Academic	Universitat Jaume I (UJI)	Castelló, Spain
Health care	Fundació Privada Clínic per a la Recerca Biomèdica (HCB)	Barcelona, Spain

Table 7.2: Hygia partners

The *Hygia* project deals with several CGs that detail the stages to be followed to cure an ailment in a particular range of patients, and in a concrete health care context. Moreover, they include indications of the expected evolution of the patients, fact that permits the evaluation at any moment of the patient, the group of patients, or the physician levels of adherence. In addition, the used guidelines are focused in the co-ordination aspects, which are derived from the patient management by several professionals through different assistance levels.

To achieve this general goal the project proposes the following, more specific, sub objectives:

O_1 To design, develop and implement a set of tools aiming at making as much automatic as possible the knowledge acquisition process from textual CGs documents to electronic CGs that could be interpreted by computers.

O_2 To propose a methodological framework for making CGs from electronic protocols and, eventually, from other additional resources, such as the data stored in hospital information systems (see O_3).

O_3 To construct and use new inductive algorithms to generate health care knowledge from data about the medical activities stored in the hospital information systems and with the use of ontologies that supply the semantic profile about the domain where the guideline is located.

O_4 To employ CGs for supporting doctors in decision making, and hospital managers in quality assessment by means of a multi-agent system (MAS) that can interpret that knowledge within the institutional context where the medical activity is developed.

[3]For more information, please visit http://banzai-deim.urv.net/∼riano/TIN2006-15453/ [last visit 10/06/2008].

O_5 To identify some indicators about the level of adherence of the health care professionals to the CGs, and also ways for monitoring them.

O_6 To evaluate the adherence between the actions of the health care professionals and the suggestions of the multi-pathology CGs. It is intended to test the system in the context of a program of chronic patients.

7.2.1 Project workflow

The main objective is to develop a set of computer science methodologies and tools to help in the use and follow-up of clinical guidelines. The workflow includes three main subprojects that complete the most relevant steps in the knowledge life-cycle (in this case, medical knowledge): acquisition, representation, use and evaluation. Fig. 7.5 shows the basic modules of the workflow and their intersections.

The workflow begins with the implementation of some tools for the acquisition of medical knowledge, using natural language processing, information extraction, ontological techniques and knowledge organisation (the USC subproject). These preliminary guidelines created from texts should be tailored accordingly in order to obtain useful guidelines. This stage is accomplished by two subprojects managed by URV and UJI. URV is developing techniques allowing automatic learning of patient treatment models, from the experience stored in the hospital databases [Riaño et al. 2008, Real & Riaño 2008]. These particular (patient-based) models are combined by the UJI generic models (evidence-based) in order to create new CGs including information about treatments and careflows. The URV is also responsible of designing a multi-agent system that eases the use of the obtained CGs. The HCB project runs in parallel with the activities of the other three subprojects, providing databases and the medical expertise, proposing work lines and interpreting and analysing the obtained results. It will also test the software developed in this project.

7.2.2 Agent-based execution of clinical guidelines

In order to complement the approach of the knowledge-based systems described previously it is frequent to use computational models based on communication, like MAS, in front of other more archaic models than do not contemplate the distributed character of a real health care system (recall the benefits of these kinds of systems described in Section 1.3).

The distributed system designed in this subproject has two basic functionalities: (*1*) the automated pursuit of welfare patterns, and (*2*) the screening of the adhesion of the decisions of the doctors to the indications of the patterns.

As shown in Fig. 7.6, the architecture designed in this project is a subset of the architecture proposed in the HECASE2 system. There are four kind of agents: the doctor agent, the guideline agent, the ontology agent, and the medical record agent. The functionalities included in these agents are being changed following the project requirements.

The *guideline agent* (GA) stores the set of designed clinical guidelines. There will be available three basic guidelines to treat chronic obstructive pulmonary disease (COPD), heart failure (HF), and diabetes (DIA). Interesting combinations of these diseases, such as COPD+HF, and COPD+DIA, are also managed .

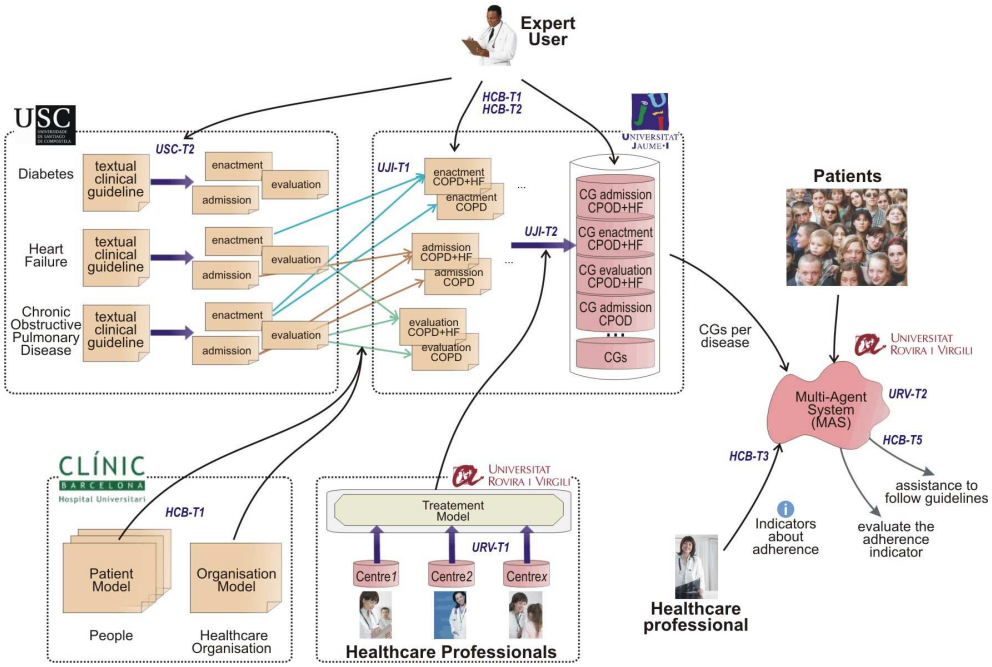


Figure 7.5: Scheme of the *Hygia* project with areas of work by subproject

The *ontology agent* (OA) will provide the same function as described in the HECASE2 system. A medical ontology will allow to identify all the elements handled in the CGs.

The *medical record agent* (MRA) stores all patient's data. The database contains values about the treatment's attributes followed by all patients (*e.g.*, discrete values such as blood pressure, and continuous ones such as an electrocardiogram) and all the results obtained in clinical tests. These data should be collected when appropriate, depending on the guideline being followed and the current state of the patient inside the treatment.

The *doctor agent* (DRA) is the main agent in the system. The DRA interacts with the human expert in order to know the action that he wants to perform and the patient on which it should be performed. With these two items, the DRA contacts with the OA, the GA and the MRA in order to collect all past results and identify the current status of the patient in the whole treatment. The doctor supervises the retrieved patient's-related values, and he can fill empty attributes and make decisions. All the steps followed by the doctor are stored by the agent in order to study how he follows a guideline. The information collected during the enactment (*e.g.*, decisions made, tasks recommended, tasks followed, tasks not followed, etc.) will be used to evaluate the adherence indicators.

The *adherence agent* (AA) is connected to the DRA in order to collect all tasks accomplished by the expert user. These data are transmitted to the AA, and they are used to evaluate an indicator of adherence of the expert with the CG.

At the moment of writing this manuscript, the definition of these indicators is not yet complete. A preliminary analysis contemplates the following possibilities: a) an academic model that includes an alarm-based system when the patient's values are not within the ranges

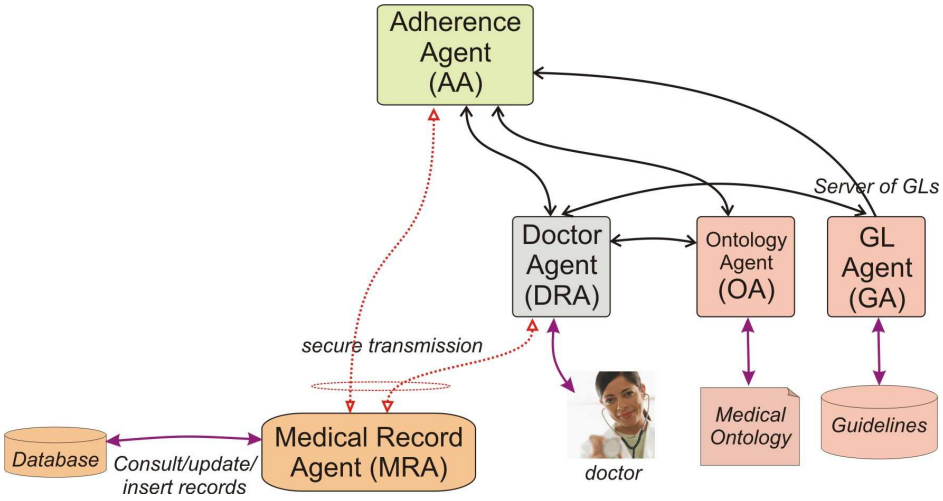


Figure 7.6: Hygia agent-based architecture

defined in the CG; b) an off-line analysis of data related to particular groups of patients; and c) an on-line model that observes the current patient's data values, the decisions made by the practitioner, and analyses the deviation. The on-line and academic models can be embedded into the DRA, but the off-line model should be implemented by the AA.

7.3 Conclusions

This chapter has shown two ongoing research projects where the ideas explained in this dissertation are being directly applied: *K4Care* and *Hygia*.

K4Care presents a complex knowledge-driven system that allows healthcare actors to execute individual intervention plans over patients. The main ideas adopted by *K4Care* inherited from HECASE2 are the following:

- To model the actors of the system as agents with its own roles. HECASE2 distinguishes among two different types of agents, patients and practitioners, whereas the *K4Care* model defines a wider range of actors including social workers, physician in charge, head nurse, etc.
- To make an agent-based coordinated execution of clinical guidelines. In the case of HECASE2, the CGs are used by doctors to know the required services (*e.g.*, clinical tests to be performed on the patients that have a particular disease). *K4Care* adopts patient-oriented CGs coordinated by the head nurse and involves other actors during the enactment of a CG.
- To include an electronic medical record (EMR) as an external service. In the case of HECASE2 there is a specialised agent that is able to access the EMR, and it allows to obtain the required patient's data, as well as insert new records. On the other hand, *K4Care* defines the EMR as a facility accessed by actors through an intermediate interface, providing a transparency similar to the one in HECASE2.
- To use a formal language to represent CGs. HECASE2 adopted initially PRO*forma*, but a facility to use more languages (*e.g.*, PRO*forma*, *SAGE*, and SDA*) was later added. In the case of *K4Care*, the SDA* language has been adopted.
- To represent the declarative knowledge through ontologies. HECASE2 uses two different ontologies, one to represent CGs and another to represent organizational and medical information about the execution of those CGs. *K4Care* has two ontologies (the actor profile ontology and the case profile ontology) that embed also the organizational and medical knowledge used to tailor and execute CGs.

The other related project, *Hygia*, is quite similar to HECASE2. The main ideas adopted by *Hygia* inherited from HECASE2 are the following:

- To define an agent-based platform to execute CGs. The *Hygia* execution module is a subset of the HECASE2 architecture adapted to the *Hygia* requirements, by adding some functionalities such as the adherence indicators evaluation during the enactment of a guideline.
- To include the EMR as an external service. *Hygia* and HECASE2 have a specialised agent that is able to access an EMR, and it allows to look up the required patient's data, as well as insert new records.
- To use a particular formal language to represent CGs. As said before, HECASE2 adopted PRO*forma* but allowing the use of other languages. In the case of *Hygia*, the PRO*forma* language has been adopted.

- To represent the declarative knowledge through ontologies. HECASE2 and *Hygia* share the same functionality by using an application ontology to represent the organizational workflow information used during the execution of CGs.

Table 7.3 summarises all topics explained above. Moreover, these two projects will be used to validate the ideas of HECASE2 in a real environment. The appropriate execution of CGs between different actors or the collection of the patient's data will be thoroughly tested. In addition, the medical partners of these projects are providing several CGs (*e.g.*, COPD, hypertension, post-stroke, heart failure), real patient's data for testing them, and finally, they will verify and use the final implemented prototypes (Hospital Clínic in the case of *Hygia*, and Amministrazione Comunale di Pollenza in the case of *K4Care*).

Topic	HECASE2	*K4Care*	*Hygia*
Actors	Practitioners, resources, patients	Set of practitioners (*e.g.*, nurse, social worker), patients	Practitioner
Electronic medical record	Accessed through a wrapper agent	Data abstraction layer with a collection of APIs	Accessed through a wrapper agent
Enactment of CGs	Coordinated	Coordinated	Collecting values from EMR step-by-step
Language to represent CG	PRO*forma*, but not closed to others	SDA*	PRO*forma*
Knowledge representation	Ontologies	Ontologies	Ontologies

Table 7.3: Related topics of HECASE2, *K4Care* and *Hygia*

Chapter 8

Conclusions and future work

Up to this moment, we have described in detail all the issues related to the design and implementation of the HECASE2 platform, the methodology, the offered services, the provision of patient-oriented services, the representation of the required knowledge, and the application into two research projects. In this final chapter, we provide a final summary of the work and present the conclusions summarising the main ideas of this dissertation, emphasising the novelty of the work done. Then, we summarise the current ongoing work and the working plan for the next three months (estimated period of the revision of this document), and in the last section, we suggest several lines of future research on different open issues presented in previous chapters and give some ideas on how they can be tackled.

8.1 Summary

The main aim of the present work has been to develop a distributed platform that allows to execute clinical guidelines in an efficient way. The most important and novel point is the complete integration of different actors, delivering patient-oriented services, allowing a coordination of their daily activities, and representing all the used knowledge in an effective way.

Many guideline-based execution engines, which provide functionalities similar to those of HECASE2 have been developed in the past, but it is not until now that researchers are starting to focus their efforts to exploit the benefits of the inclusion of CGs in the daily practice. This inclusion requires to build open and easy-to-deploy systems to allow a customisation to diverse medical centres that usually have proprietary and local clinical management systems. Although this platform has been designed with a particular purpose and provides a concrete set of services, the novel point-of-view is to represent the active actors of health care organizations (*e.g.*, practitioners, nurses, patients) as autonomous entities with different goals to achieve and acting with different roles and permissions. This permits to implement flexible systems and, at the same time, tackle some barriers that limit or restrict the adherence to CGs.

Representing all participants separately allows adding special features for particular actors, or groups of them, in a scalable way. The proposed work has suffered different updates among in the last years, adding new services and new relationships between partners. These updates, such as the management of clinical guidelines and the delivery of patient-oriented

services, have been added through the definition of new agents and conversations between them, keeping previous services unaltered.

Regarding the design of the MAS, an accurate analysis, design and implementation has been done, following the INGENIAS agent-oriented software engineering methodology. This point is not usually done by developers of MAS and allowed us to structure, document and improve the quality of the final product.

An especial attention has been paid to the inclusion of patient's preferences. The transformation of traditional health care management models into patient-centric ones is one of the current lines of funding of the European Union in eHealth. This work shows the viability and adequacy of our distributed model to provide this kind of services. This environment adds new challenges to the user's profile representation and maintenance, derived from the heterogeneity of the criteria, their number, and the way to compare and present the information to the user. Many rating, ranking and learning methods are available in the literature. Regarding the first two points, a combination of existing decision making techniques allows us to present the information to the user filtered and sorted according to his interests. To tackle the user's profile adaptation, a novel unsupervised algorithm has been designed and implemented.

In addition, the use of ontologies to represent all the knowledge has proved to be an appropriate approach. This work proposes the combination of application ontologies (*e.g.,* parts of the medico-organizational ontology) designed to cover particular issues, with some domain ontologies (*e.g.,* clinical guideline ontology) designed for general purposes, used to automate the enactment of clinical guidelines.

Finally, the evaluation of several ideas of this dissertation is currently underway in two research projects. They provide encouraging results on the suitability of our approach for executing clinical guidelines in two well distinguished domains.

Taking all of these characteristics into consideration, we believe that our proposal represents a new and interesting addition over the current state-of-the-art of the agent technology applied in medical informatics.

8.2 Future work

In this section, we describe several future lines of research and present some preliminary ideas on how they can be tackled.

- We are currently working in the development of the *clinical guideline ontology* (Section 6.2). Regarding this issue, several topics such as the validation of the ontology using various guideline representation languages, and debugging the guideline execution module by adding more specific methods to the API if they are required, are currently being studied.

- Monitoring the use of a guideline execution engine may be useful to evaluate the adherence indicators during the enactment of CGs over patients. These indicators will indicate if the CGs are being followed exactly as they are defined or some steps are repeatedly avoided by practitioners. All this information can be used to refine and improve the use of CGs in daily practice. This is also one of the main tasks to accomplish in the *Hygia* project and the ideas and implementation can be reused in both applications.

- The retrieval of CGs that have to be applied over a patient may be improved if additional recommendation processes are considered. As Shahar et al. (2003) propose, the use of information stored into the EMR of the patient and the annotation of stored CGs can allow to collect the most appropriate CGs according to the current circumstances of the patient, using case-based reasoning or other learning methods.

- There is the possibility to connect different guideline agents belonging to different medical centres creating a network of guideline agents. This group of agents may be used to versions new CGs created to cover an area. When a panel of medical experts creates a new CG, it is been intended to cover a network of medical centres. The particular adoption of a CG according to the specific circumstances of the medical centre (*e.g.*, available practitioners, resources), can be tailored by each agent and propose changes. This is a difficult task that can be performed using case-based reasoning but it requires to implement complex similarity functions in order to compare all cases.

- The execution of a CG is a complex task that involves several actors during a treatment. An improvement can be the delegation of execution during a treatment. This fact is done when a patient is referred from a department to another to perform a clinical procedure. In this case, the patient can begin another CG to perform the referred procedure and at the end, establish some result or diagnostic, required during the main treatment. This improvement will require to update the medical record to allow the execution of different CGs at the same time (with its monitorisation) and to update the current management of CGs in the doctor agent.

- The inclusion of HECASE2 into a current clinical management system requires the adaptation to the current circumstances of the client. Particularly, a translation of terms used in the CG into the particularities of the EMR are required (*e.g.*, the age of a patient can be located into a column name “age” or calculated previously taking into account the patient’s date of birth, or more complex retrieval actions).

Appendix A

Agent-Oriented Software Engineering Methodologies

One of the most important barriers to large-scale take-up of agent technology is the lack of mature software development methodologies for agent-based systems. This drawback hampers the analysis, design and implementation of this kind of distributed applications, and prevents the reuse of pieces of software from one project to another.

The use of a methodology, which consists of a set of procedures, models, and techniques, facilitates a *systematic* software development process, improving the quality of the software product. These methodologies should provide methods to describe from particular agent's elements to general organizational behaviours, as well as communication issues.

An analysis and classification of existing methodologies allows multi-agent systems developers to select the most suitable approach for a specific application. This appendix presents a state-of-the-art and classifies all the studied methodologies in two main groups. *Organizational* ones are intended to describe MAS as open systems at a high level of abstraction. *Agent-based* ones are designed for closed systems and have been more developed, being more complete. A thorough, comparison shows the advantages and shortcomings of each category and approach.

A.1 Classification of agent-oriented software engineering methodologies

All papers considered in this review were searched on SciFinder, ScienceDirect and Citeseer databases. In addition, proceedings of the most relevant conferences in the domain such as the Autonomous Agents and Multi-agent Systems or the International Workshop on Agent Oriented Software Engineering were also examined. Only relevant articles published between 2000 and 2007 were considered. The keywords that were used include agent-oriented methodology, aose, organizational methodologies, agent methodology, agents implementation, and life-cycle agent implementation. The references cited in the articles were also examined.

All collected papers were analysed and filtered. Twelve projects coming from academic research were selected: AGR, AUML, EXTENDED GAIA, GAIA, INGENIAS, MAS-COMMONKADS, MASE, MOISE/MOISE+, OPERA, PROMETHEUS, PASSI and TROPOS.

In the next sections these tools are summarised and compared but, first of all, a classification of these methodologies to develop MAS is required. Nowadays, there does not exist a widely accepted classification of those methodologies but most of the authors agree with two main families named *agent-oriented* and *organizational*. The whole classification is shown in Fig. A.1.

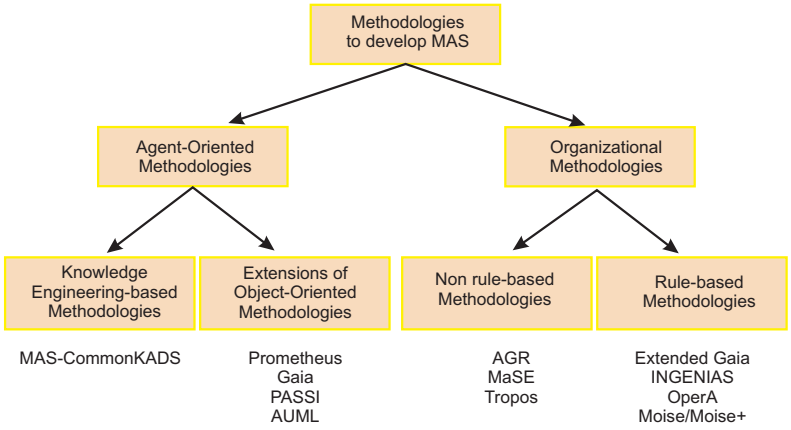


Figure A.1: Classification of methodologies to develop MAS

The first classifications identified different methodologies intended to design agent-oriented systems [Wooldridge & Ciancarini 2000], which inherited the ideas of previous methodologies based on object-oriented systems. Then, new ideas were added to that domain providing more flexible and open approaches, adopting the ideas of knowledge engineering and cognitive acquisition of knowledge [Alonso et al. 2004]. Recently, several authors have identified another branch that exploits the idea of agents as collections of entities that play a social role [Ferber et al. 2004, Horling & Lesser 2005]. These organizational methodologies have been divided depending on whether they follow predetermined rules of behaviour or not [Argente et al. 2006].

In the following, a deep study of all these types of methodologies is performed paying

special attention to their main features.

A.2 Agent-Oriented Methodologies

As it was said previously, agent-oriented methodologies can be seen as a collection of rules that allow to describe the process elements and the work product and documentation. In addition, these methodologies should be able to represent the autonomous and proactive behaviour of agents.

These methodologies focus on agent-based system development by distinguishing two main steps: analysis and design. Some works extend the design stage to the development using an agent-oriented programming language.

As shown in Fig. A.1, these kinds of methodologies have been divided into two more specific families: knowledge engineering-oriented and object-oriented depending on the design paradigm used in each approach.

A.2.1 Knowledge Engineering-Based Methodologies

Knowledge engineering methodologies can provide a good basis for MAS modelling since they deal with the development of knowledge-based systems. Since agents have cognitive characteristics, these methodologies can provide the techniques for modelling the agents' knowledge. The definition of the knowledge of an agent can be considered as a *knowledge acquisition* process, and only this process is addressed in these methodologies.

Adapting knowledge engineering methodologies for the design of agent-based systems has certain advantages, as such methodologies provide techniques for modelling the agents' knowledge and knowledge acquisition processes. In addition, any existing tools, ontology libraries and problem-solving methods can be reused. However, such methodologies fail to address the distributed or social aspects of agents, or their reflective and goal-oriented attitudes, since a knowledge-based system is conceived as a centralised one [Iglesias et al. 2000].

A.2.1.1 MAS-CommonKADS

MAS-COMMONKADS is an agent-oriented software engineering methodology that guides the process of analysis and design of MAS, which was developed at Technical University of Madrid, Spain [Iglesias & Garijo 1999, Iglesias & Garijo 2005].

MAS-COMMONKADS extends the COMMONKADS methodology [Schreiber et al. 2000] for knowledge-based systems by employing techniques from object-oriented methodologies as well as protocol engineering. MAS-COMMONKADS adapts the models proposed by COMMONKADS in order to allow the description of agents (by adding an specific agent model) and their interactions (by adding a coordination model).

MAS-COMMONKADS follows the RUP life-cycle phases. The first one, called *Conceptualisation Phase*, deals with extracting the basic system requirements from the user. Then, in the *Analysis Phase*, a number of models are developed (see Fig. A.2) [Tran et al. 2005]:

- *Agent Model*: specifies the agent characteristics using use cases, problem statements, Responsibility Driving Design (RDD) and Class Responsibility Collaboration (CRC) techniques [Wirfs-Brock et al. 1990].

- *Task Model*: identifies and decomposes the tasks that the agents can carry out. In practice, tasks can be organised in terms of *parallel decomposition*, *sequential decomposition*, *optional task* and *iterative task*.
- *Coordination Model*: describes the agents interactions from use case scenarios. It describes the conversations between agents as well as the services that each agent offers to other agents.
- *Expertise Model*: specifies different types of knowledge that agents require to achieve their goals (*e.g.* domain knowledge, task knowledge, inference knowledge and problem-solving methods). That model expresses how an agent makes a decision and uses the notation inherited from COMMONKADS.
- *Organization Model*: describes the MAS organization in terms of agent aggregation and inheritance.
- *Communication Model*: identifies the human-software agent interactions for developing appropriate interfaces.

The third phase, called *Design Phase*, is based on the previously developed models and defines a last model:

- *Design Model*: specifies infrastructure facilities, agent architecture, software and hardware required for MAS implementation.

The fourth phase, *Development and testing*, codes all previous models into a programming language. The authors propose JADE as agent-oriented programming framework [Bellifemine et al. 2007].

Finally, the *Operation* stage allows developers to run and maintain the system.

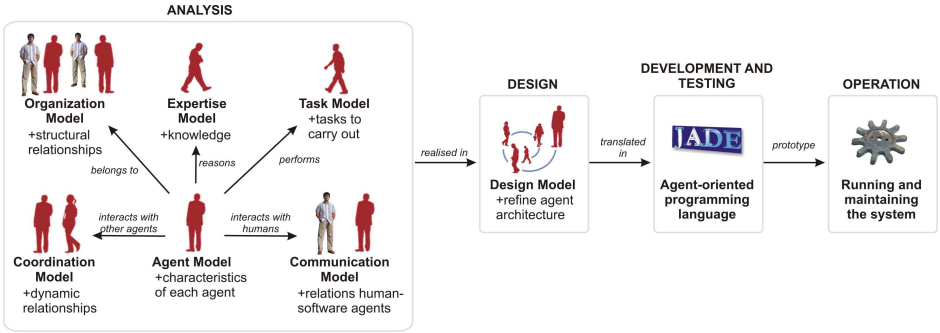


Figure A.2: Life-cycle and models of MAS-COMMONKADS (adapted from [Iglesias & Garijo 2005])

A.2.2 Object-Oriented Methodologies

Due to the similarity between the object and agent paradigms, some researchers have proposed the extension of object-oriented methodologies to the design of agent-based systems.

Good examples of such methodologies are PROMETHEUS or GAIA. Agents are handled as *complex* objects with remote calls mechanisms, but also with proactivity and autonomy. Some authors have argued that organizational patterns are difficult to design with these approaches [Alonso et al. 2004, Odell 2002].

A.2.2.1 Prometheus

PROMETHEUS is an start-to-end methodology (analysis, design and implementation) developed at Royal Melbourne Institute of Technology (RMIT University), Australia [Padgham & Winikoff 2002, Padgham & Winikoff 2004, Padgham & Winikoff 2005].

Basically, it consists of three phases (see Fig. A.3):

a) The *system specification* identifies the basic *functionalities*, using *percepts* (inputs), *actions* (outputs) and *case scenarios* of the target MAS.

The models defined in that phase could be a few paragraphs of descriptions and are intended to describe roughly the system. This is an iterative and non-linear process that is not necessary to be achieved in any specific sequence, but it is desirable to begin from the definition of the goals, then the definition of the case scenarios, followed by the design of the functionalities, and finally, a description of the interface of the system with the environment. A functionality encompasses a number of related goals, percepts that are relevant to it, actions that it performs, and data that it uses.

b) The *architectural design* is focused on identifying agents, events, interactions and shared data objects.

This phase identifies the agent types by grouping agent functionalities through a *coupling diagram* and an *acquaintance agent* diagram. The resulting agents are described using *agent descriptors*. From the case scenarios, the interactions between agents are described in the *interaction diagrams* and the *interaction protocols*.

c) The *detailed design* describes the internal details of each agent. The goal of this stage is to describe internal events, plans and detailed data structures, in order to begin the translation into a programming language.
The *agent overview* and the *capability descriptors* diagrams show the capabilities features. From the *interaction diagrams*, the *process specifications* are made through a set of protocols.
At the bottom of the life-cycle there are the most detailed diagrams. From the *agent overview* and the *capability descriptors*, the plans within a capability are captured in a *capability overview diagram*. Finally, *data*, *event* and *plan descriptors* for each capability are described.

Each of those phases includes models that are intended to explain the *dynamics* of the system, graphical models to define the structure of the system and its (basic) components, and textual models to provide a detailed description of all individual entities.

In addition, the authors of this methodology implemented a design tool called Prometheus Design Tool (PDT)[1] [Padgham & Winikoff 2004, Thangarajah et al. 2005]. That tool allows

[1]Prometheus Design Tool (PDT) is freely available at http://www.cs.rmit.edu.au/agents/pdt/ [Last visit: 12/03/2008]

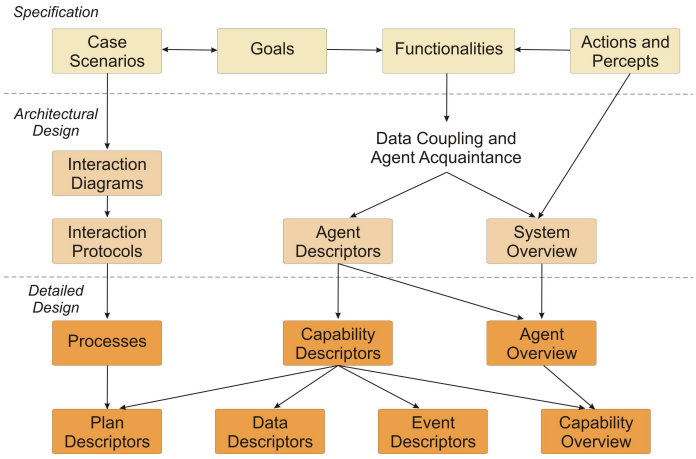


Figure A.3: Development life-cycle proposed in PROMETHEUS [Padgham & Winikoff 2005]

developers to reuse and share different models and, at the end, obtain a core of prototypical agents coded using the JACK[2] agent-oriented language.

A.2.2.2 GAIA

GAIA was one of the first methodologies created to assist agent developers in the analysis and design phases [Wooldridge & Ciancarini 2000, Zambonelli et al. 2001, Zambonelli et al. 2000]. It was an effort from the University of Liverpool (UK), University of Southampton (UK), and Università di Modena e Reggio Emilia (Italy).

The authors suggested a general framework to analyse distributed problems based on the idea of interacting roles between actors (Fig. A.4). Such complex systems are implemented using a top-down agent identification process. First, they identify agents from roles or actors (high-level analysis procedure) and their components (knowledge, behaviours, etc.). After that, they identify interactions with other agents (low-level design).

GAIA supports two phases of the development process (Fig. A.4):

- *Analysis phase*: identifies the system's organisation (roles) and its associated protocols. Two models are created:
  - *Roles Model*: identifies what the actors should perform. Each role is defined by four attributes: responsibilities (functionality of the roles expressed through liveness (execution trajectory) and safety (invariants) rules), permissions to interact with resources, activities (do not require interaction with other agents) and protocols (actions that require interaction with other agents).
  - *Interactions Model*: explains how the actors should interact in order to achieve their goals.

[2] JACK is an environment for building, running and integrating commercial-grade multi-agent systems using a component-based approach. JACK is a trademark of Agent Oriented Software Group (http://www.agent-software.com) [Last visit: 12/03/2008].

- *Design phase*: aggregates roles into several agent types, documents all instances of those agent types, identifies the services and, finally, identifies all acquaintance relationships.
  - *Agent Model*: is defined using a simple agent type tree in which leaf nodes correspond to roles (as defined in the *roles model*), and other nodes correspond to agent types.
  - *Services Model*: is the most important document, that allows to identify the services (functionality) associated with each agent role, and specify their main properties. Every activity (procedure that involves only one agent, in contrast with protocols that involve more than one agent) will correspond to a service, but not every service will correspond to an activity. The model identifies inputs, outputs, pre-conditions and post-conditions of each service.
  - *Acquaintance Model*: defines the communication links between agent types. It does not define which messages are sent or when messages are sent. It is used to identify any potential communication bottleneck.

As a result of applying that methodology, one has a set of models (documents) in both phases (analysis and design) that allows a programmer to begin the implementation.

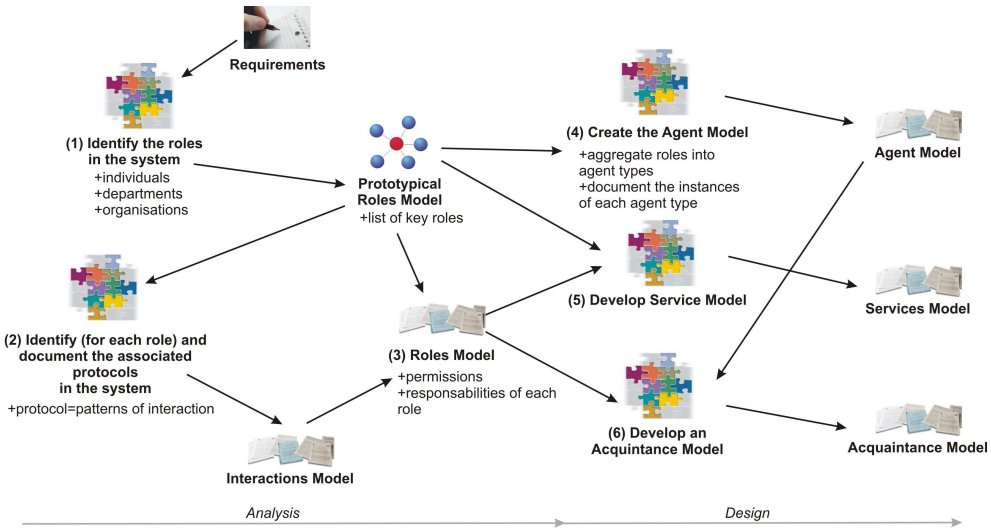


Figure A.4: Models defined in GAIA (adapted from Bernon et al. (2005))

GAIA2JADE [Moraïtis et al. 2002, Moraïtis & Spanoudakis 2004] constitutes a set of recommendations or patterns to generate code for agents implemented in JADE[Bellifemine et al. 2007]. That set of rules intends to help agent programmers to ease the code generation. From the *agent model* the programmers have all agent types to be implemented. The *services model* shows all functions implemented in the system. The *acquaintance model* allows to identify the interactions between agents. Then, the most difficult documents to interpret are the *interactions* and the *roles model*. The authors propose to implement each liveness rule, protocol and activity with some JADE behaviours (cyclic or one-shot).

A.2.2.3 Process for Agent Societies Specification and Implementation (PASSI)

Process for Agent Societies Specification and Implementation (PASSI) is a step-by-step requirement-to-code iterative methodology, which was developed at Istituto di Calcolo e Reti ad Alte Prestazioni of Consiglio Nazionale delle Ricerche (ICAR-CNR), Palermo (Italy) [Chella et al. 2006, Cossentino et al. 2005, Cossentino 2005].

PASSI was designed to develop multi-agent societies integrating design models and concepts from both object-oriented software engineering and MAS, using an agent-oriented extension of the UML notation. One of the main characteristics of PASSI is that it proposes to use several standards such as FIPA Architecture [FIPA 2002*c*] to model the MAS, supported by UML Modelling Language [OMG 2007], and represented using XML [W3C 2007].

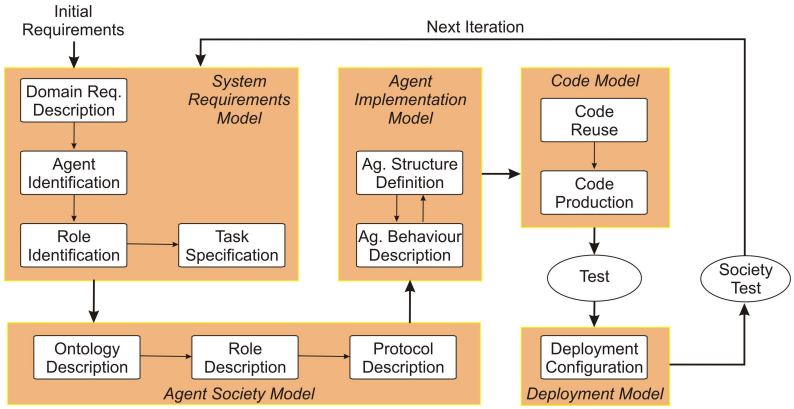


Figure A.5: Phases of the life-cycle proposed in PASSI (from [Cossentino 2005])

PASSI proposes an iterative sequence of five phases in its development life-cycle (see Fig. A.5):

a) *System Requirements Model*: a model of the system requirements in terms of agency and purpose. It is composed of four steps:

- *Domain Requirements Description*: a functional description of the system using conventional use case diagrams.
- *Agent Identification*: the phase of attribution of responsibilities to agents, represented as stereotyped UML packages.
- *Role Identification*: a series of sequence diagrams exploring the responsibilities of each agent through role-specific scenarios.
- *Task Specification*: specification of the capabilities of each agent with activity diagrams.

b) *Agent Society Model*: a model of the social interactions and dependencies among the agents involved in the solution. Developing this model involves three steps:

- *Ontology Description*: use of class diagrams and constraints (specified using the Object Constraint Language notation, [IBM 1997]) to describe the knowledge

ascribed to individual agents and their communications. It is depicted in terms of concepts (taxonomy of basic terms), actions (affect the concepts' status) and predicates (assertions about concepts); it follows the FIPA specifications.

- *Role Description*: class diagrams are used to show the roles played by agents, the tasks involved, communication capabilities, and inter-agent dependencies.
- *Protocol Description*: use of sequence diagrams to specify the grammar of each pragmatic communication protocol in terms of speech-act performatives.

c) *Agent Implementation Model*: a classical model of the solution architecture in terms of classes and methods. The most important difference with the common object-oriented approach is that there are two different levels of abstraction, the social (multi-agent) level and the single-agent level. This stage is composed of the following steps:

- *Agent Structure Definition*: conventional class diagrams describe the structure of solution agent classes.
- *Agent Behaviour Description*: activity diagrams or state charts describe the behaviour of individual agents.

d) *Code Model*: a model of the solution at the code level requiring the following steps to produce it:

- *Generation of code* from the model using:
  - PASSI TOOLKIT (PTK): this plug-in[3] allows to export the multi-agent system model to AGENTFACTORY[4] or generate the code for just the skeletons of the designed agents, behaviours, and other classes included in the project.
  - AGENTFACTORY: it can create complex multi-agent systems by using patterns from a large repository and can also provide the design documentation of the composed agents.
- *Manual completion of the source code* can be required in this stage.

e) *Deployment Model*: a model of the distribution of the parts of the system across hardware processing units and their migration between processing units. It includes a final step:

- *Deployment Configuration*: deployment diagrams describe the allocation of agents to the available processing units and any constraints on migration and mobility.

A.2.2.4 Agent Unified Modelling Language (AUML)

Agent Unified Modelling Language (AUML) is a formalism used as standard by FIPA to represent agent communication interactions and protocols, which was created by James Odell Associates (US) and Siemens (Germany) [Bauer et al. 2001, Huget & Odell 2004, Huget et al. 2004, Odell et al. 2000].

[3] PASSI Toolkit is freely available at http://sourceforge.net/projects/ptk. [Last visit: 21/11/2007]

[4] AGENTFACTORY is a framework that supports a structured approach to the development and deployment of agent-oriented applications. It was developed by researchers from the School of Computer Science and Informatics at University College Dublin. More information can be found at http://www.agentfactory.com. [Last visit 21/11/2007]

AUML is based on the object-oriented modelling representation in UML (a very recent release updates the diagrams to UML 2.0) [OMG 2007]. The main goal of AUML is to offer to developers a *notation* that can be used to analyse, design, and implement MAS. There are several diagrams offered by UML (uses cases, packages, objects, collaboration, components, deployment); AUML extends the functionality to represent MAS through two new diagrams: *sequence diagrams* and *agent class diagrams*:

a) *Sequence diagrams* in MAS are diagrams which express the exchange of messages through protocols. Sequence diagrams have two dimensions: the vertical dimension represents time, and the horizontal dimension represents different instances or roles (see Fig. A.6) [Bauer et al. 2001, Odell et al. 2001]. Messages in sequence diagrams are ordered according to a time axis. It is the standard *de facto* used by FIPA to represent all FIPA Interaction specifications [FIPA 2002*c*].

b) *Agent class diagrams*. A class diagram in UML shows a set of classes, interfaces, collaborations and their relationships, and it is the most common diagram found when modelling object-oriented systems. AUML uses the same syntax but with different purposes because agents are not objects and the representation should allow to express knowledge, plans or protocols used in the agent-based system [Bauer 2001, Huget 2004].
The agent class diagram includes the name of the agent, a description of its internal attributes, a detailed description of all tasks implemented/allowed in each agent, a list of the offered methods, descriptions of the capabilities, services and supported protocols, information about the group where an agent is located, and an agent finite state automata to describe the internal behaviour.

The main goal of AUML is to represent agent systems in terms of interaction. That interaction is explained through protocols and behaviours using UML-based diagrams. An evolution of AUML based on UML 2.0 offers more possibilities of expressiveness with new diagrams. One of the main advantages is that AUML is based on a well-known modelling language and it is easy to learn but, unfortunately, it was only applied to describe FIPA interaction protocols and there are no tools to support the full design of a MAS [FIPA 2002*c*].

A.3 Organizational Methodologies

A multi-agent system has two properties which seem contradictory: a global purpose and an autonomous behaviour of its individual components. While the autonomy of the agents is essential for the MAS, it may cause the loss of the global coherence. The organization of a MAS is used to manage these intra- and inter- agent properties.

Sichman et al. (2005) introduced the basic principles of the organizations in agents:

- *Agents interactions may eventually create dynamic organizations*. Whenever the same interaction patterns are repeated several times, involving the same agents, these interactions may be captured by pre-established structures, thus avoiding the inherent complexity of bottom-up emergent organization formation.
- *Agents organizations limit agents interactions, aiming to optimize the achievement of global goals*. As a consequence, collective behaviour will be more efficient, since the organization formation is carried on a priori.

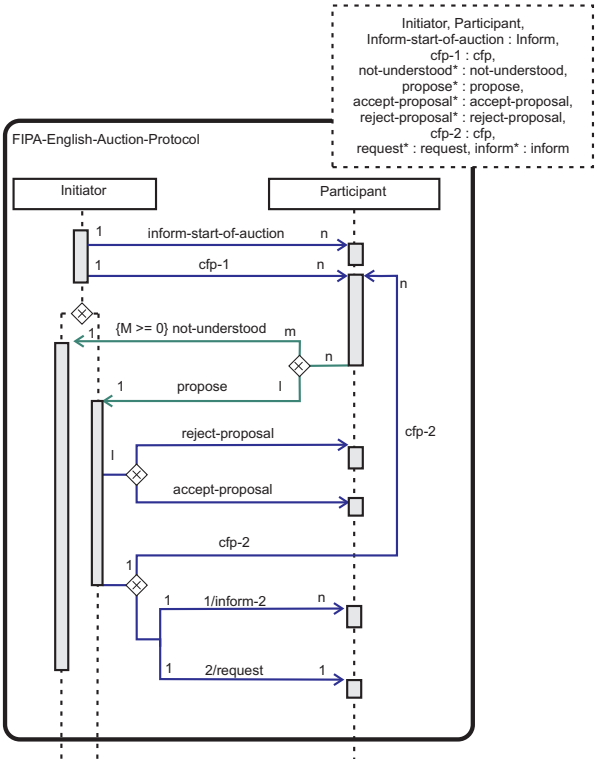


Figure A.6: Example of Agent Interaction Protocol template in AUML: English Auction protocol [Bauer et al. 2001, FIPA 2002*d*]

Consequently, these dimensions of MAS create a virtuous circle: interactions build dynamic organizations, and pre-defined static organizations limit agents' interactions in order to achieve more efficiently the MAS global goals.

A.3.1 Classification

There are two types of organizational methodologies, depending on the concept of organization:

i) methodologies which only consider an organizational structure and propose common goals for the set of agents, and

ii) methodologies which, moreover, explicitly present *rules of behaviour* among agents, called *norms*. A language for norms of role r is defined as:

$$\varphi ::= O_r\varphi | P_r\varphi | F_r\varphi$$

where $O_r\varphi$, $P_r\varphi$ and $F_r\varphi$ indicate the obligation, permission and prohibition for a role r to see to it that φ holds, respectively [Dignum 2004].

In other words, *norms* define the *rights* and *obligations* of the agents related to the roles that they play and their environment or area of activity. To some extent, organizational rules can be considered as the liveness and safety properties of the organization. Therefore, it is essential to consider *norms* to describe the rules of behaviour of the agents within the organization [Dastani et al. 2002], although some methodologies do not allow these features.

A.3.2 Rule-Based Methodologies

These methodologies define a general framework with *norms* that govern the agents' actions and their conversations. Examples of such methodologies are EXTENDED GAIA, INGENIAS, OPERA, and MOISE, which will be analysed in the following sections.

A.3.2.1 Extended GAIA

The GAIA methodology explained in Section A.2.2.2 was revisited three years later by the same authors in order to represent a MAS as an organized *society* of individuals [Cernuzzi et al. 2004, Zambonelli et al. 2003, Zambonelli et al. 2005].

Agents play social *roles* (or responsibilities) and interact with others according to *protocols* determined by the roles of the involved agents. With that approach, the overall system behaviour is understood in terms of both micro and macro levels. The former explains how agents act according to their roles, and the latter explains the pattern of behaviour of those agents. These constraints are labelled *organization rules* and *organization structures* respectively.

The analysis phase documents all the functional characteristics that the system should offer, together with the characteristics of the operational environment in which the MAS is situated. The leitmotiv of this phase is the determination of organizations as groups of individuals that exhibit a common behaviour, or that interact loosely with other portions of the system, or that require competencies that are not needed in other parts of the system.

A new model, named *environmental model*, documents explicitly the entities and resources that a MAS can exploit by defining a set of allowed actions (see Fig. A.7).

Two *preliminary* documents are also outputs of this phase: *role* and *interaction* models. The *preliminary role model* is intended to identify some characteristics of the system that are likely to remain the same independently of the required organizational structure, and the *preliminary interaction model* is intended to distinguish the dependencies and relationships between roles, named *protocols* (actions that require to exchange messages with other agents).

The design stage takes the models and preliminary models from the previous phase in order to define a reliable architecture of agents and complete the unfinished models.

First of all, the agent designer should identify the structure of the organization to be implemented. GAIA is independent from the final selected implementation and for this reason it gives some general rules to define that organization. To tackle this issue it could be useful to adopt one predefined organizational paradigm (*e.g.*, federations, hierarchies, holarchies) according to the system requirements [Horling & Lesser 2005]. As in the first version of GAIA, the *Roles model* identifies what the actors should perform, and the *Interactions model* explains how the actors should interact in order to achieve their goals, *i.e.*, it is an operational description of the MAS organization.

When these models are defined, only agent and services models remain to be defined. The *Agent model* defines for each agent which roles should implement and, if it is possible, the number of instances of each kind of agent that the MAS should support. Finally, the *Services model* documents the services (derived from the list of protocols, responsibilities, activities, and roles) that may be performed by an agent. The agent designer identifies inputs, outputs, pre-conditions, and post-conditions but does not prescribe implementation details.

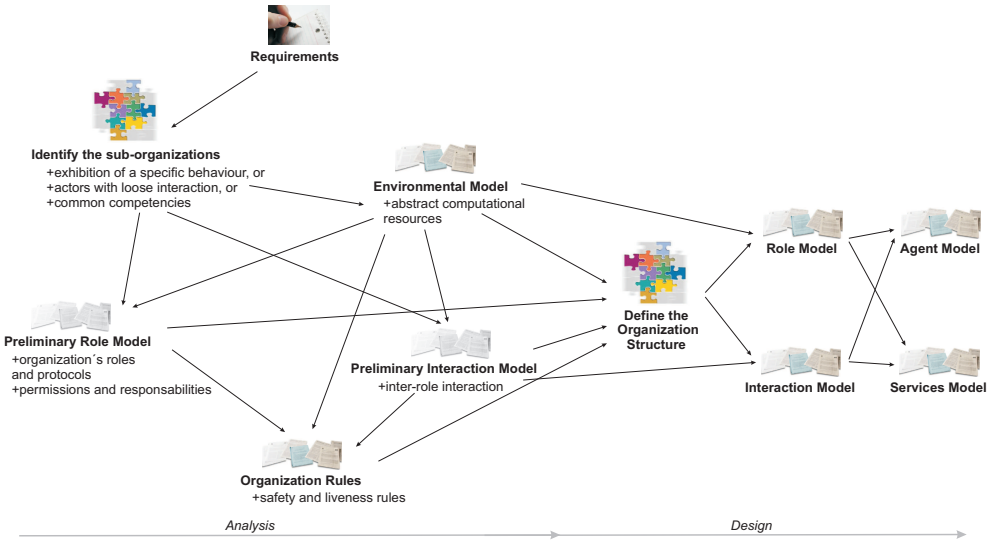


Figure A.7: Step-by-step EXTENDED GAIA life-cycle (adapted from [Zambonelli et al. 2005])

A.3.2.2 INGENIAS

INGENIAS provides a notation to guide the development process of a MAS from analysis to implementation. INGENIAS was developed at Universidad Complutense of Madrid (Spain) [Gómez-Sanz 2003, Pavón et al. 2005, Pavón et al. 2006].

INGENIAS is an improvement of MESSAGE/UML (Methodology for Engineering Systems of Software Agents [Caire et al. 2001]). The relationships among models and the identification of activities have been refined to generate MAS specifications, and it incorporates new support tools and development examples.

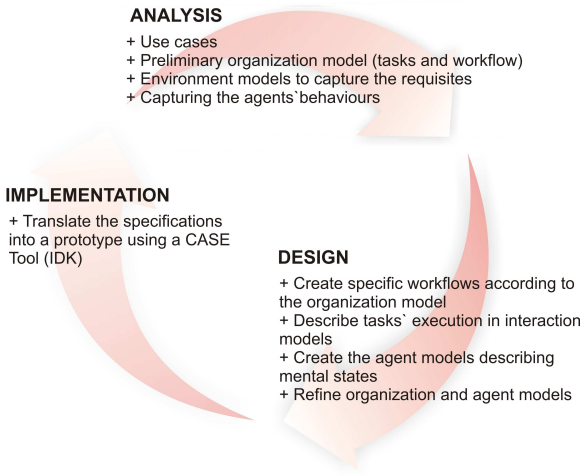


Figure A.8: Life-cycle defined in INGENIAS (adapted from Pavón et al. (2005))

INGENIAS proposes different models in the phases of analysis and design (see Fig. A.8).

As regards social norms, they are not explicitly modelled, although they are implicit in the organizational viewpoint. Organizational dynamics are not considered *i.e.*, how agents can join or leave the system, how they can form groups dynamically, what their life-cycle is, etc. [Argente et al. 2006].

The authors have developed an agent-oriented software tool called Ingenias Development Kit (IDK)[5]. It allows to edit *consistent* models (according to INGENIAS specification) and generate code with documentation for different platforms such as JADE, Robocode, Servlets or Grasia! Agents.

A deeply study of this methodology is done in Section 4 and complemented with the Appendix B, due to it has been used to design the work presented in this dissertation (HECASE2).

A.3.2.3 Organizations per Agents (OperA)

Organizations per Agents (OPERA) is the result of a PhD Thesis made at Utrecht University (The Netherlands) [Dignum 2004, Dignum & Dignum 2005, Dignum et al. 2002, Dignum

[5]Ingenias Development Kit (IDK) is freely available at http://ingenias.sourceforge.net with some examples. Its last version is 2.6 [last visit 11/11/2007]

et al. 2004].

OPERA is a framework that describes a MAS as an organizational structure regulated by social contracts (that describe the roles in the society) and interaction contracts (that describe the interactions between agents).

The three components of an OPERA model are (see Fig. A.9):

a) *Organizational model*. It describes the behaviour of the organization. The organization goals are distributed among several roles. It includes four structures:

- *Social structure*. It specifies the objectives of the society, its roles and what kind of model governs coordination.
- *Interaction structure*. It describes scene scripts (a set of coordinated tasks included in roles) and a partial ordering between them, which are interactions between roles.
- *Normative structure*. It specifies the society norms and regulations, expressed in terms of roles and interaction norms.
- *Communicative structure*. It specifies the ontologies that describe the domain concepts and communication acts.

b) *Social model*. This model details how the enactment of roles by agents is made. Social contracts establish an agreement between the agent and the organization model and define the way in which the agent will fulfil its roles. A social contract defines a role-enactment agent (REA).

c) *Interaction model*. According to the interaction scripts defined in the interaction structure, different scenes are created dynamically by REAs. REAs negotiate specific interaction agreements with each other, which are arranged in interaction contracts.

Depending on the coordination model, the design of the agent society will be different. Therefore, it is recommended to identify as a preliminary step in the development of the agent society which it will be. OPERA authors consider that their methodology supports open society systems because it fulfils the following requirements:

- *Internal autonomy requirement*. The interaction and structure of the society must be represented independently from the internal design of participating entities.
- *Collaboration autonomy requirement*. The activity and interaction in the society are specified without completely arranging in advance the interaction structures. As a consequence, OPERA provides flexibility and personalization of the organizational design.

Its critical point is that there is not any implementation of the OPERA model which demonstrates its practical possibilities and proves its conceptual choices.

A.3.2.4 Model of Organization for Multi-Agent Systems (Moise)

Model of Organization for multI-agent SystEms (MOISE) constitutes a family of methodologies created among École Nationale Supérieure des Mines (France) and Universidade de São Paulo (Brazil) [Hannoun et al. 2000, Hübner et al. 2006, Hübner et al. 2002*b*, Hübner et al. 2005, Hübner et al. 2007].

MOISE is a methodology structured along three levels [Hannoun et al. 2000]:

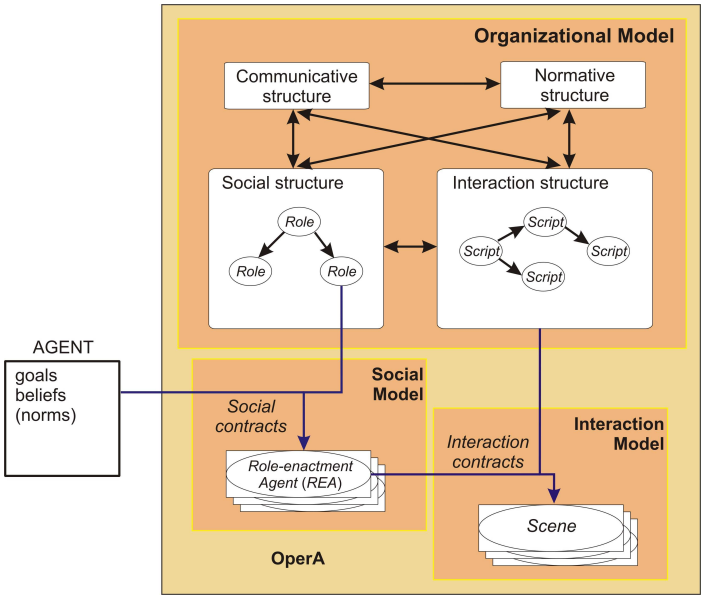


Figure A.9: OPERA Architecture [Dignum 2004]

a) *Individual level*. It defines the tasks that each agent is responsible of. They are the *roles*, which constrain the action possibilities of each agent according to their missions (parts of a plan decomposed in this agent). An agent, playing a given role in the organization, must obey the permitted behaviours specified by the missions building the role.

b) *Collective level*. It defines the aggregation of roles in large structures (*groups*), which constrain the agents they can cooperate with. It is defined by a set of roles, a set of missions and a set of links. There are two kinds of groups:

- *Internal*, that link only related roles of the group (sources and targets).
- *External*, in any other case. This aggregation depends on the policy that is adopted by the designer of the application.

c) *Social level*. It defines the interconnections between roles that constrain the agents' behaviour. There exist constraints related to other agents (*e.g.*, authority, communication channels), and related to a common task (*e.g.*, commitments).

MOISE covers the organizational aspects from two points of view [Hübner et al. 2002*b*]: i) *functioning* that describes global plans, policies to allocate tasks to agents, the coordination to execute a plan, and the quality of a plan (*i.e.*, time consumption, resources usage) and, ii) *structure* that explains the roles, the relations among them (*e.g.*, communication, authority), roles obligations and permissions, group of roles, etc. Fig. A.10 shows how an organization could explain or constrain the agents' behaviour in case we consider an organization as having both structural and functional dimensions.

If only the functional dimension is specified, the organization has nothing to"tell" to the agents when no plan can be performed. Otherwise, if only the organizational structure

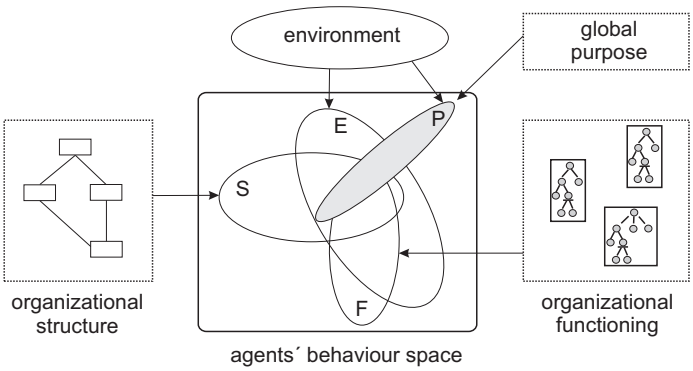


Figure A.10: The organizational effects on a MAS (**P**: all behaviours which draw the MAS's global purposes; **E**: all possible behaviours in the current environment; **S**:organizational structure (roles, groups and links); **F**:functional dimension) [Hübner et al. 2002*a*]

is specified, the agents have to reason for a global plan every time they want to act together. Thus, in the context of some application domains, it is feasible to think that if the organization model specifies both dimensions while maintaining a suitable independence among them, then the MAS that follows such a model can be more effective in leading the group behaviour to its purpose. Another advantage of having both specifications is that the agents can reason about the others and their organization regarding these two dimensions in order to better interact with them (in the case, for example, of social reasoning).

Two diagrams characterise the MOISE model: a) the *organizational structure* (OS), which describes a web of roles (nodes), links (arcs) and groups independently of the agents being in the system, and b) the *organizational entity* (OE), which shows a set of agents functioning under an organizational structure; it is an instantiation of an OS.

The main shortcoming of MOISE, which motivated its extension, is the lack of the concept of an explicit global plan in the model and the strong dependence among the structure and the functioning [Hübner et al. 2002*a*]. The main objective of MOISE+ was to create an organization-centred model including these three aspects [Hübner et al. 2006, Hübner et al. 2007]:

- *Structural level*. It defines the agents' relations through the notions of roles and links. In this proposal, the original MOISE model is enriched with concepts such as role inheritance, recursive groups, role compatibility, and role cardinality.

- *Functional level*. It describes how a MAS achieves its global goals, *i.e.*, how these goals are decomposed (by plans) and distributed to the agents (by missions). The original MOISE's plans are local to the agents. The MOISE+ contributions here are the inclusion of the concept of global plan, called Social Scheme, and the definition of preferences between missions.

- *Deontic level*. It describes the roles' permissions and obligations for missions (*e.g.*, norms, laws).

Other variations of MOISE+ have emerged in the last years:

- *Software* MOISE+ ($S-$MOISE+) [Hübner et al. 2005]. It tries to fill the gap between the organisational constraints and the agents autonomy. This software[6] ensures that all agents will follow the organisation norms without requiring that they are developed in a specific language or architecture.

- *Jason* MOISE+ ($J-$MOISE+) [Bordini et al. 2007]. It helps to program agents with AGENTSPEAK using the open-source interpreter JASON[7].

A.3.3 Non-Rule-Based Organizational Methodologies

In the following sections we analyse some organizational methodologies that are not based on rules. Examples of such methodologies are AGR, MASE and TROPOS.

A.3.3.1 Agent-Group-Role (AGR)

Agent-Group-Role (AGR) is the evolution of the AALAADIN model [Ferber & Gutknecht 1998], which was developed at Université Montpellier II (France) [Báez-Barranco et al. 2007, Ferber et al. 2004, Ferber et al. 2005, Gutknecht et al. 2001].

The AGR model is based on three main concepts:

- *Agent*. An agent is an active, communicating entity playing roles within groups. An agent may hold multiple roles, and may be member of several groups. No constraints are placed upon the architecture of an agent or about its mental capabilities.

- *Group*. A group is a set of agents sharing some common characteristic. A group is used as a context for a pattern of activities, and creates partitions in the organizations. Two agents may communicate if and only if they belong to the same group, but an agent may belong to several groups. This feature allows the definition of organizational structures.

- *Role*. The role is the abstract representation of a functional position of an agent in a group. An agent must play at least a role in a group, although it may play several ones. Roles are local to groups, and a role must be requested by an agent. A role may be played by several agents.

AGR proposes three stages in its methodology:

a) Identify the main *groups* of the application (a set of similar agents or a set of agents that perform the same function).

b) Build the overall *organizational structure*. Two kinds of diagrams are used:

  - A *CheeseBoard Diagram* gives a first sketch of the organizational patterns (see Fig. A.11). A *group* is represented as an oval, *agents* are represented as skittles that stand on the board (and sometimes go through the board, when they belong to several groups) and a *role* is represented as an hexagon. Lines link hexagons with agents.

[6] S-MOISE+is freely available at http://moise.sourceforge.net [last visit: 12/03/2008].
[7] J-MOISE+ is freely available at http://jason.sourceforge.net [last visit: 12/03/2008].

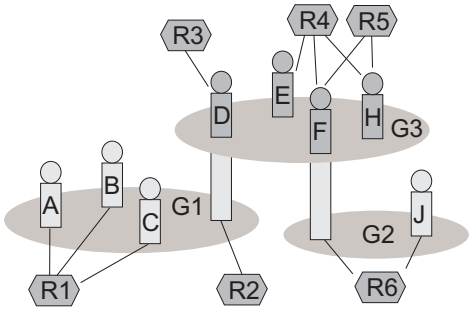


Figure A.11: AGRCheeseBoard notation

- The *Organizational Structure Diagram* represents the static aspects of agents, as well as hierarchies and divisions.

c) Build the *organizational dynamics* of group creation and adhesion with *Organizational Sequence Diagrams* which get into the definition of roles in a functional way. These diagrams are a variant of sequence diagrams oriented to organizations.

A.3.3.2 Multi-agent systems Software Engineering (MaSE)

Multi-agent systems Software Engineering (MASE) is an start-to-end methodology, which was developed at the Air Force Institute of Technology (USA) [DeLoach 2001, DeLoach 2004, DeLoach et al. 2001, Wood & DeLoach 2000].

The primary focus of MASE is to guide a designer through the software life-cycle from a documented specification to an implemented agent system. MASE is independent of a particular multi-agent system architecture, agent architecture, programming language, or message-passing system.

MASE is an iterative methodology across all phases with the intent that successive models add detail to the previous ones (see Fig. A.12). Based on RUP [Jacobson et al. 1999], its development process phases are:

a) *Analysis*: produces a set of roles whose tasks have to cover the initial system requirements. It includes these three phases:

- *Capturing goals*: identifies the set of system goals and structures them in a *Goal Hierarchy Diagram* by importance.
- *Applying Use Cases*: captures use cases from the initial system requirements and restructures them as a *Sequence Diagram*. At least one sequence diagram is created from each use case. If there are several possible scenarios, multiple diagrams are created.
- *Refining Roles*: transforms the structured goals of the *Goal Hierarchy Diagram* into roles, which define agent's classes and capture system goals during the design phase. The general case transformation is one-to-one; each goal maps to a role. Role definitions are captured in a traditional *Role Model*.

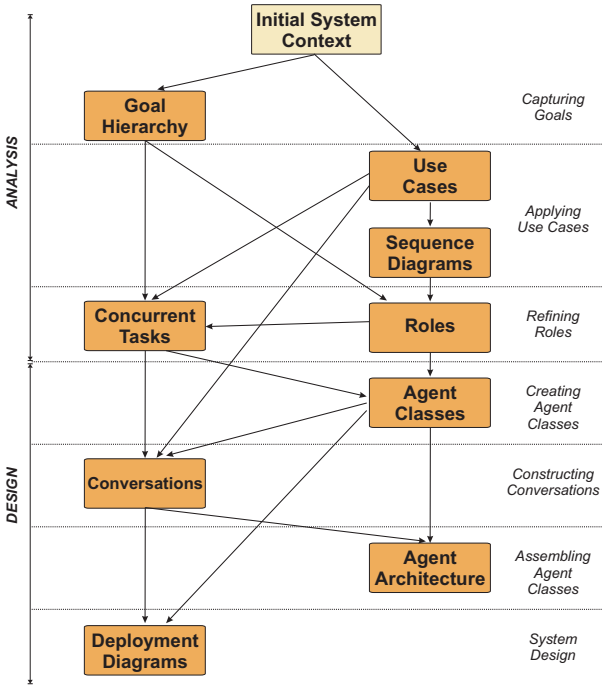


Figure A.12: Life-cycle defined in MASE [Wood & DeLoach 2000]

b) *Design*: defines the overall system organization by transforming the roles and tasks into agent types and conversations, following these steps:

- *Creating Agent Classes*: identifies agent classes from component roles. The product of this phase is an *Agent Class Diagram* which depicts agent classes and the conversations between them. Mapping goals to roles is generally one-to-one between roles and agent classes, but it is not compulsory.
- *Constructing Conversations*: defines a coordination protocol between two agents. A conversation consists of two *Communication Class Diagrams*, one each for the initiator and the responder, and it must support and be consistent with all sequence diagrams derived earlier.
- *Assembling Agent Classes*: creates the set of agent classes.
- *System Design*: takes the agent classes and instantiates them as actual agents. It uses a *Deployment Diagram* to show the numbers, types and locations of agents within a system.

A final element to consider is the automatic code generation. The authors developed a CASE tool called AGENTTOOL[8] that supports the entire life-cycle and also verifies the correctness of the agent protocols. Furthermore, it eases MASE deployment and learning.

[8] AGENTTOOL is freely available at http://macr.cis.ksu.edu/projects/agentTool/agentool.htm [Last visit 12/03/2008].

A.3.3.3 TROPOS

TROPOS is an incremental methodology that allows to design and implement multi-agent systems [Bresciani et al. 2004, Giorgini et al. 2004, Giorgini et al. 2005]. It was a collaborative project among research groups at Università degli Studi de Trento (Italy), Centro per la Ricerca Scientifica e Tecnologica (ITC-IRST) (Italy), and University of Toronto (Canada).

TROPOS is based on two key ideas. First, the notion of agent and all related mentalistic notions (for instance goals and plans) are used in all phases of software development, from early analysis down to the actual implementation. Second, TROPOS covers also the very early phase of requirements analysis, thus allowing for a deeper understanding of the environment where the software must operate, and of the kind of interactions that should occur between software and human agents (founded on *belief*, *desire*, and *intention* (BDI) agent architectures [Georgeff et al. 1999]).

It adopts a goal-based approach from the *i** model [Bresciani et al. 2004], which offers actors, goals and actor dependencies as primitive concepts, and for its modelling activities (actor, goal, plans and capabilities), UML [OMG 2007] and AUML (see Section A.2.2.4) diagrams are used.

The four main development phases of the TROPOS methodology are (see Fig. A.13):

a) *Requirements analysis*. It is split in two main phases which share the same conceptual and methodological approach:

- *Early Requirements*: the requirements engineer identifies the domain stakeholders and models them as social actors, who depend on one another for goals to be achieved, plans to be performed, and resources to be furnished.
- *Late Requirements*: the conceptual model is extended including a new actor, which represents the system, and a number of dependencies with other actors of the environment. These dependencies define all the functional and non-functional requirements of the system-to-be.

b) The *architectural design* defines the system's global architecture in terms of sub-systems (actors), interconnected through data and control flows (dependencies). It provides also a mapping of the system actors to a set of software agents, each characterized by specific capabilities. In Kolp et al. (2006), a classification of *Organizational Styles* is proposed in TROPOS according to two disciplines: i) *Organization Theory* that describes the structure and design of an organization such as structure-in-5, pyramid style, chain of values, matrix, bidding style; and ii) *Strategic Alliances* which describe collaborations of independent organizational stakeholders who have agreed to pursue a set of agreed-upon business goals. Some styles are: joint venture, arm's length, or hierarchical contracting.

These authors also recommend to incorporate into the system's architecture some patterns at a micro level like broker, mediator, wrapper or embassy.

c) The *detailed design* aims at specifying agent capabilities and interactions.

d) The *implementation* stage establishes a mapping between the implementation platform constructs and the detailed design notions. It provides a detailed implementation of organizational models into JACK (see Section A.2.2.1).

Dignum et al. (2004) noted the main shortcomings of TROPOS: *a*) it is *not formal* (although there is some ongoing work on providing a formal semantics for TROPOS), and *b*) it is *too organizational-centred* in the sense that it does not consider that agents can have their own goals and plans, and not just those coming from the organization. Furthermore, TROPOS has no concept representing the normative aspects of an organization.

In Mallya & Singh (2005), the *Theory of Commitment Protocols* was applied to TROPOS to improve the detection of interactions and dependencies in early phases.

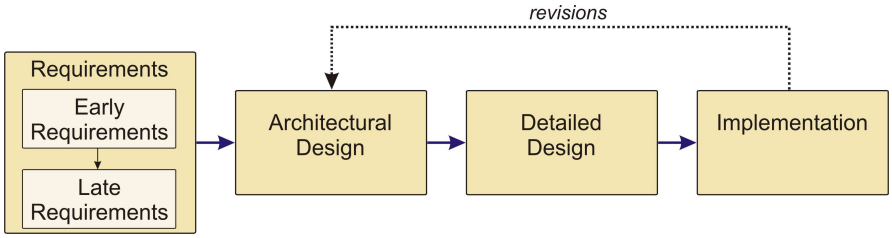


Figure A.13: Life-cycle defined in TROPOS

A.4 Comparative evaluation

From the exhaustive analysis of several approaches made in previous sections, some general features which are essential in practice can be extracted, such as organizational aspects or internal agent's behaviour. Some basic features of any software engineering methodology are compared in this section, from which basic concepts of multi-agent systems are covered to the provided documentation. The evaluation framework is intended to give a general overview of all tools. Before presenting the framework, an analysis of previous surveys will be sketched.

A.4.1 Previous works in the field

Several researchers have tried to analyse and compare different AOSE approaches [Alonso et al. 2004, Cuesta et al. 2004, Iglesias et al. 2000, Sturm & Shehory 2004*b*, Tran et al. 2003, Wooldridge & Ciancarini 2000]. Basically, each proposal defines a framework with several properties that are studied and evaluated for all cases. There is not any proposal that covers both organizational and traditional approaches and we propose in this appendix another framework and an exhaustive comparison of heterogeneous approaches.

The first comparative studies were based in object-oriented languages [Frank 1998, Wood & DeLoach 2000]. Unfortunately, these comparatives lack an analysis of specific agent-oriented parameters.

[Cernuzzi & Rossi 2002] proposed a qualitative analysis followed by a quantitative rating. They constructed an *Attributes Tree*, which organizes the considered criteria in weighted branches. After rating the leafs, the value of the root can be calculated and compared to other methodologies. The authors identified three kinds of criteria. *Internal* attributes characterize the internal structures of the agents. *Interaction* attributes describe how the interactions inside the system can be modelled. Finally, the *Process Requirements* judge the design and development process proposed by the methodologies. [O'Malley & DeLoach 2001] propose

a quantitative evaluation based on two main sets of criteria: *management* (evaluates usage and applicability features) and *technical* (*e.g.*, scalability, level of integration in legacy systems, robustness).

Recently, [Lin et al. 2007] proposed an evaluation framework divided into four main categories. The first group of criteria analyses the general features, concepts and their general properties in a MAS. Then, the internal agent's behaviour is analysed in terms of the main general concepts, their relations and their complexity. After that, a software engineering-based set of parameters allows to compare the level of coverage of all tools, such as life-cycle completeness and system domain definition. The last group deals with the practical deployment of the methodology in any application. This framework is an update of a previous work done by [Sturm & Shehory 2004*a*] and [Shehory & Sturm 2001].

Two similar works made by [Sabas et al. 2002] and [Tran et al. 2003] defined two other evaluation frameworks, named Multidimensional framework of Criteria for the Comparison of MAS methodologies (MUCCMAS) and Feature Analysis, respectively. These frameworks were inspired in the OO area but were customised to the MAS domain covering different metrics to analyse general concepts of MAS systems; however, there is a lack of organizational issues and open systems coverage. Another contribution in this field was made by [Cuesta et al. 2004], which propose an exhaustive framework based on five main groups covering several software engineering areas as well as agent-oriented topics. One of the most important advantages of this framework is that the items are evaluated with some of allowed values in an objective fashion. In addition, the evaluation was made by collecting several opinions received from researchers through a web form.

One of the most used frameworks was designed by [Dam & Winikoff 2004]. It defines four evaluation criteria: concepts or properties used by the methodology, modeling, process and pragmatics. Using this framework, the authors compared three methodologies: TROPOS, MASE and PROMETHEUS. Furthermore, a revision of this framework done by [Sudeikat et al. 2004] added new terms to the main four groups of criteria. Unfortunately, this framework does not provide criteria about openness and topologies supported by the approaches.

In general, each framework evaluates a collection of methodologies with similar features and referring to similar meta-models.

A.4.2 Description of the evaluation criteria

The proposed framework's evaluation criteria are classified into five main groups:

a) *Concepts and properties*, which refer to the agent-oriented concepts and properties which any methodology should implement [Wooldridge & Ciancarini 2000].

- *Autonomy*, that expresses the ability of any agent to solve a problem in an autonomous way.
- *Communication*, that describes the communication model used, such as message-based or blackboard-based.
- *Cooperation issues*, that explains *how* a common goal is managed by agents.
- *Adaptability*, that describes whether it is possible to express the internal agent's behaviour by reacting/changing to environment events or incoming requests from other agents.

  - *Proactivity*, that describes whether it is possible to express that agents can initiate actions following a goal.

b) *Modelling language criteria*, which deal with the expressiveness and formalisation levels of the defined notation.

  - *Formalisation/preciseness of models* indicates if the models are clearly defined.
  - *Expressiveness* that allows to represent the data and the flow of data inside a system.
  - *Abstraction* to create different levels of detail of models.

c) *Model-related criteria* which evaluate the capabilities of the methodologies' models.

  - *Coverage of the life-cycle*. The set of phases of the life-cycle covered by the methodology.
  - *Complexity* measures the effort level needed to learn and use it.
  - *Temporal continuity* that expresses the changes of the agents across time.
  - *Human computer-interaction*. Multi-agent systems may require to exchange information with (human) users. That interaction should be designed appropriately and represented (typically) as use cases.

d) *Organizational criteria* which evaluate social relationships among the agent communities.

  - *Open systems* expresses if the methodology allows to represent the incorporation/removal of new agents/resources dynamically.
  - *Topology*. Agent communities relationships can be expressed with different paradigms such as hierarchies, holarchies, federations, coalitions [Horling & Lesser 2005]. Methodologies can be constrained to some of those or be independent.
  - *Social norms* specify high-level communication patterns among agents or groups of agents.

e) *Supporting feature criteria* try to give some considerations about support tools.

  - *Software and methodological support* describes if any CASE tool exists (*e.g.*, libraries of agents, agent components, architectures or technical support).
  - *Availability of examples* is a useful help while learning or implementing any methodology.
  - *Usage in projects* is an important factor that represents the maturity of a methodology.

A.4.3 Methodologies comparison

Tables A.1 and A.2 show the results obtained after the evaluation of the previously described methodologies in the defined framework. The evaluation was made by three researchers. The evaluations were aggregated using a linguistic ordered weighted aggregator (LOWA) with five linguistic terms (*very high*, *high*, *medium*, *low* and *very low*) [Isern et al. 2006*b*]. The aggregation has an internal membership function that allows to give priority to low or high values. In that case, the *most* policy was selected, which gives priority to high values.

Table A.1: Summary of studied methodologies with basic details

	MAS-CommonKADS	Prometheus	Gaia	Passi	AUML	AGR	MaSE	Tropos	Extended Gaia	Ingenias	OperA	Moise-/Moise+
Concepts and properties												
Autonomy	$+$	$++$	$+$	$++$	$-$	$+$	$+$	$+$	$+$	$++$	$+$	$++$
Communication	$+$	$++$	$++$	$++$	$++$	$+$	$+$	$+$	$++$	$++$	$+$	$++$
Cooperation	$+$	$+$	$+$	$+$	$-$	$+$	$+$	$\approx$	$+$	$++$	$-$	$+$
Adaptability	$+$	$n.a.$	$-$	$-$	$-$	$+$	$-$	$\approx$	$-$	$+$	$-$	$\approx$
Proactivity	$+$	$++$	$++$	$-$	$--$	$+$	$--$	$-$	$++$	$++$	$-$	$\approx$
Modelling language												
Formalisation	$-$	$++$	$++$	$\approx$	$+$	$-$	$-$	$+$	$++$	$++$	$\approx$	$+$
Expressiveness	$-$	$\approx$	$+$	$\approx$	$+$	$-$	$-$	$+$	$\approx$	$+$	$\approx$	$+$
Abstraction	$-$	$+$	$+$	$+$	$+$	$-$	$-$	$-$	$+$	$+$	$+$	$+$

Notation: $++$: Very high or totally agree; $+$: High or agree; $\approx$: Medium or don't specified explicitly by the authors; $-$: Low or disagree; $--$: Very low or totally disagree; $n.a.$: Not available

Table A.2: Summary of studied methodologies with basic details (*continuation*)

	MAS-CommonKADS	Prometheus	Gaia	Passi	AUML	AGR	MaSE	Tropos	Extended Gaia	Ingenias	OperA	Moise-/Moise+
Model-related												
Coverage	$A/D/I$	$A/D/I$	A/D	$A/D/I$	A/D	A/D	$A/D/I$	$A/D/I$	A/D	$A/D/I$	A/D	A/D
Complexity	$\approx$	$++$	$-$	$n.a.$	$-$	$-$	$-$	$--$	$n.a.$	$++$	$\approx$	$+$
Temporal	$n.a.$	$\approx$	$-$	$-$	$++$	$--$	$-$	$-$	$\approx$	$+$	$-$	$+$
Human-computer	$++$	$-$	$-$	$--$	$--$	$--$	$--$	$--$	$--$	$-$	$--$	$--$
Organizational criteria												
Open systems	$+$	$-$	$--$	$+$	$-$	$-$	$+$	$+$	$+$	$++$	$-$	$+$
Topology	$+$	$-$	$+$	$-$	$--$	$-$	$-$	$+$	$-$	$++$	$+$	$+$
Social norms	$\approx$	$-$	$-$	$-$	$\approx$	$+$	$+$	$+$	$+$	$+$	$+$	$+$
Supporting features												
Software	$+$	$++$	$--$	$++$	$--$	$--$	$++$	$++$	$--$	$++$	$--$	$-$
Examples	$+$	$+$	$+$	$+$	$\approx$	$-$	$+$	$++$	$+$	$+$	$--$	$-$
Projects	$+$	$++$	$+$	$+$	$-$	$\approx$	$+$	$++$	$+$	$++$	$-$	$-$

Notation: $++$: Very high or totally agree; $+$: High or agree; $\approx$: Medium or don't specified explicitly by the authors; $-$: Low or disagree; $--$: Very low or totally disagree; $n.a.$: Not available; $A/D/I$: Analysis/Design/Implementation

Concepts and properties With regard to the first group of criteria, the overall level of support for autonomy of all the methodologies is good. Communication issues are well covered by all approaches. Concerning cooperation concepts, INGENIAS supports very well these issues whereas AUML and OPERA provide weaker support. Adaptability, which is a difficult feature to evaluate, is the worst covered property, and only MAS-COMMONKADS, AGR and INGENIAS pass this evaluation. Finally, proactivity is very well supported by PROMETHEUS, GAIA, EXTENDED GAIA, and INGENIAS.

Modelling language There exists a disparity of notations in the internal agents' behaviours. With regard to the formalisation, some of the tools define clearly the sequence of stages with well documented models, such as PROMETHEUS, GAIA, EXTENDED GAIA, and INGENIAS, whereas MAS-COMMONKADS, AGR and MASE do not provide enough information. The expressiveness of data structures are not very well supported, in general. Most of the approaches hide details of how data are included or mapped in the system and how agents exploit them. Finally, the evaluation of the level of abstraction, which measures the degree of abstraction in the whole methodology, in non ruled-based approaches and MAS-COMMONKADS can not be well rated because papers hide these details.

Model-related criteria From the software-engineering point of view, all methodologies cover the analysis and design phases of the RUP life-cycle. Moreover, some of these methodologies cover the implementation using an agent-oriented programming language, such as MAS-COMMONKADS, PROMETHEUS, PASSI, MASE, TROPOS and INGENIAS. The evaluation of the difficulty to learn and use these methodologies is an arduous task. After the evaluation, all methodologies were tested and, in some cases, CASE tools were downloaded. For instance, MOISE and INGENIAS are very difficult to learn, whereas GAIA, AUML, or TROPOS are easier. Temporal issues are not covered widely by the analysed methodologies. Only AUML, INGENIAS and MOISE add time constraints and changes (*e.g.*, deadlines, roles changing across time). Finally, the human-agent interaction is only covered (explicitly) in MAS-COMMONKADS, which defines an specific model to design the interfaces and the conversations between them.

Organizational criteria Designing a MAS as an open system where agents can join and leave dynamically can be a good feature. INGENIAS covers perfectly this issue, and MOISE, EXTENDED GAIA, TROPOS, MASE, PASSI and MAS-COMMONKADS allow to specify a certain degree of dynamics in their models. The definition of an explicit topology in a MAS is not a feature widely covered. INGENIAS, OPERA, MOISE, TROPOS, GAIA and MAS-COMMONKADS allow to design the required agent society model. Finally, the inclusion of social norms, which provides flexibility to the conversations, is managed by the organizational-based methodologies.

Supporting features The methodologies that include the implementation stage give some support to the developers (*e.g.*, INGENIAS, TROPOS, MASE, PASSI, PROMETHEUS) whereas the others only offer information in published papers (usually without all details of implementation). Generally, methodologies give some examples to show the main features as well as to explain the design phases. AGR, MOISE and OPERA presented the difficulty of looking for implemented examples. A good measure of the maturity of a methodology is its use

in projects. Some of the approaches have been tested in projects (MAS-COMMONKADS, PROMETHEUS, GAIA, PASSI, MASE, TROPOS, EXTENDED GAIA and INGENIAS).

A.5 Conclusions

Throughout this chapter we have described and classified the most relevant approaches related to agent-oriented software engineering. In our opinion, this is a broad research area with a lot different methodologies, meta-models and ways to design a MAS, which should be analysed deeply in order to both understand and compare all existing approaches. We now outline some conclusions on four main topics: *disparity of models*, *CASE tools*, *evaluation frameworks*, and *shortcomings*.

Disparity of models. There exists a large variety of notations and criteria in the internal roles of actors in all methodologies. In that sense, Bernon et al. (2005), after analysing some methodologies, intended to propose a unified meta-model of concepts and internal relations attending to the features extracted in the analysed tools. Unfortunately, this proposal has not been adopted in a full fledged methodology, but it is a good option to be considered. Previously, other efforts tackled by the FIPA Methodology Task Group[9] and the AgentLink III AOSE Technical Forum Group[10] also analysed the problem and tried to define a *standard* meta-model, but their results have had a limited impact in the research community. All the methodologies cover the analysis and design phases, creating a set of models using some kind of (more or less structured) notation. The subset of those approaches that includes implementation are language-dependent, because the models are designed taking into account the final agent-oriented language. For instance, INGENIAS is designed to work with JADE, and PROMETHEUS is designed to work together with JACK, but not in any other way. The adoption of a common meta-model for all partners (methodology developers) should allow to export the designed models in different programming languages (with, of course, differences in the implementation according to the features offered in each case), but with a degree of compatibility between them, which now does not exist. For this reason, it is an immature research area with too many methodologies that difficult its adoption and its usage.

CASE tools. An agent designer uses a methodology to structure and design all parts and resources of a project. The use of any CASE tool to assist during the development is very interesting to ease and share the results, as well as to learn the used approach. In our view, this is one of the most important features to take into account in order to select one methodology. For instance, even though GAIA and EXTENDED GAIA are two of the most cited methodologies, they are not useful in a daily basis because, at the end of the design phase, the agent designer has a lot of documents that are very difficult to validate and translate to an agent-oriented programming language. In contrast, methodologies such as INGENIAS, MASE and PROMETHEUS provide graphical aids that allow an agent developer to work easily with them in a real application.

Evaluation frameworks. The main objective of the proposed evaluation framework is to give an overall view of the main features provided in all cases. This comparative study should facilitate an agent designer to select the most appropriate approach according to his requirements. The results of the evaluation show that the adoption of a methodology from start

[9] More information at http://www.fipa.org/activities/methodology.html [Last visit: 12/03/2008].
[10] More information at http://www.agentlink.org/ [Last visit: 12/03/2008].

(requirements analysis) to the end (implementation) is very difficult and the available tools should be improved. Fortunately, this is an ongoing work and, for instance, INGENIAS developers plan to finish a new release next year.

Shortcomings. The design of a MAS is a complex task due to the fact that many conversations, protocols, roles, and organizational issues should be considered. The latter are not usually being covered by most of the evaluated methodologies. Specifically, human-computer interactions are only explicitly modelled in MAS-COMMONKADS.

Unfortunately, other agent-related issues such as mobility and security are not covered in the analysed methodologies yet. The former allows the design of mobile agents requiring autonomous entities, which additionally move between computers, using different resources (with permissions) and processing methods. The latter helps to add safety issues in the data management of a MAS.

Appendix B

Ingenias notation symbol

The INGENIAS Development Kit (IDK) allows an agent developer to implement a MAS following the stages (analysis, development and implementation) defined in the INGENIAS Development Process[1] [Gómez-Sanz & Pavón 2006, Pavón & Gómez-Sanz 2003].

This methodology relies on the widespread Unified Process and the used release allows to generate code automatically and other useful processes, such as specification verification.

Using diagrams similar to those of UML, a developer specifies different aspects of the system. The main models to design are: agent models, environment models, organization models, interaction models, and task and goal models (see also Chapter 4). The particular notation defined in INGENIAS can be difficult to read and learn. This appendix summarises the main terms and symbols used in the IDK tool.

[1] For a detailed review of INGENIAS, readers can check http://ingenias.sourceforge.net/ [last visit 14/03/2008]

B.1 Graphic symbols

The following list summarises the main graphic symbols and its associated meaning:

Agent. Represents an agent of the multi-agent system.

Application. An application is a wrapper of a computational system entity. It is an interface with a concrete behaviour.

Belief. A believe is a set of asserts that are not certainties, just expectations.

Collaboration diagram in Grasia!. This description allows to talk about the technology used to transfer information from one agent to another, refer to the mental conditions that must meet the initiator and the collaborators at each step, what tasks will be executed and when, and what is the execution order of the different communication acts.

Collaboration diagram in UML. Describes how the interaction among agents takes place. Each interaction declaration includes the involved actors, goals pursued by the interaction, and a description of the protocol that follows the interaction. Grasia! and UML specifications can include the whole specification of an interaction.

Concrete agent. Represents an instance of an agent or a set of agents in runtime. Its main goal is to express a running instance of an agent without an special concern on the type of agent.

Environment application. It represents an application that already exists in the environment that surrounds the MAS. An Application is a wrapper of an element that it is neither agent nor a resource. You can configure methods in the application and relate this application with agents. Applications define agents perception. They correspond to other systems that are already implemented and with which we agents have to interact.

General Event. It is an event produced by an application.

Fact. Describes an information that the agent accepts as reliable. This general entity contains this information in the description field.

Goal. According to the BDI model, a goal is a desired state that an agent wants to reach. In planning, a goal is represented by a world state. Here a goal is an entity by itself, however it can be related with a representation of the world state using satisfaction relationships with tasks. These relationships contains references to descriptions of mental states of agents, so they refer to the image of the world that agents have.

Group. A group contains other groups, roles, agents, applications, or resources. It represents the structure of an organization. Groups, and organizations as well, are useful when the developer foresees a high number of agents that may be working together.

Ingenias Use Case. A use case is configured with information about preconditions and postconditions, as well as information of the different interactions that may appear.

Interaction. Represents an interaction between two or more agents or roles. There can be only one initiator and at least one collaborator. An interaction also details the goal that it pursues.

Interaction unit. It consists on invoking methods on objects allocated on other machines as if they were in the same local one. It assumes that there exists an interface for the remote object.

Organization. An organization is a set of agents, roles and resources that get together to achieve one or several goals. Inside an organization there are not other organizations, just groups. An organization is similar to an enterprise. Internally it is composed by departments that may be restructured without affecting its external image.

Resource. Describes a resource according to TAEMS notation [Decker 1996]. Unlike TAEMS, there is no distinction between consumable and non-consumable resources.

Role. A role is a self-contained group of functionalities. When an agent plays a role we want to express that it has to execute tasks associated to a role.

Task. Tasks are the encapsulation of actions or non-distributable algorithms. Tasks can use Applications and resources. Tasks generate changes in the mental state of the agents that execute them. Changes consist of: (a) modifying, creating or destroying mental entities; or (b) changes in the perception of the world by acting over applications (applications act over the world producing events, that are perceived by the agent). Though tasks can be also assigned to roles, at the end, they will belong to an agent.

Workflow. A workflow is an abstraction of a process that has been automatised using activities. It identifies its responsible agents.

B.2 Relationships codification

All the relationships present in the diagrams follow a codification to ease its comprehension.

Workflow	→	**WF**+*identifier*
Agent	→	**A**+*identifier*
Interaction	→	**I**+*identifier*
Interaction unit	→	**IU**+*identifier*
Goal task	→	**GT**+*identifier*
AGent Operation	→	**AGO**+*identifier*
Organization	→	**O**+*identifier*
Environment	→	**E**+*identifier*

Glossary

Agent-Oriented Software Engineering (AOSE) Software engineering methodology designed to analyse and design distributed multi-agent systems.

Aggregation operator An aggregation operator composes a single object of a given set from n-tuples of objects belonging to the same set. An aggregation operator should satisfy these properties: identity when unary, boundary conditions and a non-decreasing behaviour.

Clinical guideline (CG) Clinical guidelines are a set of directions or principles to assist the health care practitioner with patient care decisions about appropriate diagnostic, therapeutic, or other clinical procedures for specific clinical circumstance.

Clinical Management System (CMS) See *Electronic Health Record*.

Clinical practice guideline (See *clinical guideline*)

Electronic Health Record (EHR) There is no universally accepted definition of an EHR. As more functionality is added the definition will need to be broadened. Importantly, EHRs are also known as clinical management system, electronic medical records (EMRs), computerised medical records (CMRs), electronic clinical information systems, and computerised patient records (CPR). Throughout this dissertation we will use EHR as a larger system that includes the EMR and PHR and interfaces with multiple other electronic systems locally, regionally or nationally (See also *EMR and PHR*) [Hoyt et al. 2007].

Electronic Medical Record (EMR) EMR is the electronic patient record located in an office or hospital (see also *Electronic Health Record*).

FIPA The Foundation of Intelligent Physical Agents (FIPA) is now the eleventh Standards Committee of the IEEE Computer Society, which promotes agent-based technology and the interoperability of its standards with other technologies. FIPA Specifications are grouped in different categories such as agent communication, agent transport, agent management, abstract architecture, and applications and can be found at `www.fipa.org`.

JADE Java Agent DEvelopment Framework (JADE) is an open source software framework fully implemented in Java language that simplifies the implementation of multi-agent

systems. It complies with the FIPA-IEEE specifications and through a set of graphical tools that supports the debugging and deployment phases. The latest version of JADE is JADE 3.5 released on 25th June 2007.

Multi criteria decision analysis (MCDA) MCDA is a set of systematic procedures for analysing complex decision problems (see MCDM). MCDA techniques can be used to identify a single most preferred option, to rank options, to list a limited number of options for subsequent detailed evaluation, or to distinguish acceptable from unacceptable possibilities.

Multi criteria decision making (MCDM) MCDM is a discipline aimed at supporting decision makers who are faced with making numerous and conflicting evaluations.

Patient Health Record (PHR) PHR is a collection of health information by and for the patient that can be part of the EMR (see also *Electronic Health Record*).

RUP IBM Rational Unified Process (RUP) is a comprehensive process framework that provides industry-tested practices for software and systems delivery and implementation and effective project management. The life-cycle starts with a planning of the requirements, the analysis and design detail the performance of the system (architecture and identify all functions and procedures to execute). Then, implementation translates the design documents into a system. Finally, all the functions and procedures are testing to guarantee its correctness.

References

Abidi, S. R., Abidi, S., Hussain, S. & Shepherd, M. (2007), Ontology-based Modeling of Clinical Practice Guidelines: A Clinical Decision Support System for Breast Cancer Follow-up Interventions at Primary Care Settings, *in* K. A. Kuhn, J. R. Warren & T.-Y. Leong, eds, 'Proceedings of the 12th World Congress on Health (Medical) Informatics, MEDINFO 2007', Vol. 129 of *Studies in Health Technology and Informatics*, IOS Press, Brisbane, Australia, pp. 845–850.

Advani, A., Shahar, Y. & Musen, M. A. (2003), 'Medical Quality Assessment by Scoring Adherence to Guideline Intentions', *Journal of the American Medical Informatics Association* **9**, 92–97.

Agree, C. (2003), 'Development and validation of an international appraisal instrument for assessing the quality of clinical practice guidelines: the AGREE project', *Quality and Safety in Health Care* **12**(1), 18–23.

Ahn, B. S. (2006), 'The Uncertain OWA Aggregation with Weighting Functions Having a Constant Level of Orness', *International Journal of Intelligent Systems* **21**, 469–483.

Alonso, A., Marcos, M., Alonso, J., Gelagert, G., Martínez-Salvador, B., Riaño, D. & Taboada, M. (2008), A Knowledge-Acquisition Framework to Facilitate the Development and Reengineering of Care Plans in Electronic Format, *in* 'In Proc. of Tromsø Telemedicine and eHealth Conference 2008', p. to appear.

Alonso, F., Frutos, S., Martínez, L. & Montes, C. (2004), Towards a Natural Agent Paradigm Development Methodology, *in* G. Lindemann, J. Denzinger, I. J. Timm & R. Unland, eds, 'Multiagent System Technologies. Proc. of Second German Conference, MATES 2004', Vol. 3187 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 155–168.

Alves, V., Neves, J., Nelas, L. & Marreiros, F. (2006), Web-based Medical Teaching using a Multi-Agent System, *in* A. Macintosh, R. Ellis & T. Allen, eds, 'In Proc. the Twenty-fifth SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence, SGAI 2005', Springer Verlag, Cambridge, UK, pp. 181–194.

Amigoni, F., Dini, M., Gatti, N. & Somalvico, M. (2003), 'Anthropic agency: a multiagent system for physiological processes', *Artificial Intelligence in Medicine* **27**, 305–334.

Amigoni, F. & Schiaffonati, V. (2007), Multiagent-Based Simulation in Biology: A Critical Analysis, *in* L. Magnani & P. Li, eds, 'Model-Based Reasoning in Science, Technology,

and Medicine', Vol. 64 of *Studies in Computational Intelligence,*, Springer, pp. 179–191.

Anderson, J., Chaturvedi, A. & Cibulskis, M. (2007), 'Simulation tools for developing policies for complex systems: Modeling the health and safety of refugee communities', *Health Care Management Science* **10**(4), 331–339.

Annicchiarico, R., Cortés, U. & Urdiales, C., eds (2008), *Agent Technology and e-Health*, Whitestein Series in Software Agent Technologies and Autonomic Computing, Birkhäuser Verlag, Basel, Switzerland.

Anselma, L., Terenziani, P., Montani, S. & Bottrighi, A. (2006), 'Towards a comprehensive treatment of repetitions, periodicity and temporal constraints in clinical guidelines', *Artificial Intelligence in Medicine* **38**, 171–195.

Anselma, L., Terenziani, P., Montani, S. & Bottrighi, A. (2007), Automatic Checking of the Correctness of Clinical Guidelines in GLARE, *in* K. A. Kuhn, J. R. Warren & T.-Y. Leong, eds, 'Proceedings of the 12th World Congress on Health (Medical) Informatics, MEDINFO 2007', Vol. 129 of *Studies in Health Technology and Informatics*, IOS Press, Brisbane, Australia, pp. 807 – 811.

Argente, E., Julian, V. & Botti, V. (2006), 'Multi-Agent System Development Based on Organizations', *Electronic Notes in Theoretical Computer Science* **150**, 55–71.

Bajo, J., Botti, V., Corchado, J., Ilarramendi, A., Ilarri, S., Julián, V., Carmona, M., Marsá, I., Mena, E., Moreno, A., Pavón, J. & Valls, A. (2007), *Issues in Multi-Agent Systems. The AgentCities.ES Experience*, Software Agent Technologies, Birkhäuser Verlag Basel / Heidelberg, chapter Agent Applications in Tourism, pp. 179–206.

Bajo, J., Corchado, J., Fernández, S., Fuentetaja, R., González, M., Isern, B., López, B. & Valls, A. (2007), *Issues in Multi-Agent Systems. The AgentCities.ES Experience.*, Whitestein Series in Software Agent Technologies, Birkhäuser Verlag Basel / Heidelberg, chapter Cognitive Abilities in Agents, pp. 59–85.

Barahona, P., Azevedo, F., Veloso, M., Estevao, N. & Gallego, R. (2001), 'Computerising a guideline for the management of diabetes', *International Journal of Medical Informatics* **64**, 275–284.

Barrué, C., Cortés, U., Martínez, A., Escoda, J., Annichiarico, R. & Caltagirone, C. (2006), e-Tools: An agent coordination layer to support the mobility of persons with disabilities, *in* M. Bramer, ed., 'In Proc. of 19th World Congress of International Federation for Information Processing, Artificial Intelligence in Theory and Practice, IFIP AI 2006', Springer, Santiago, Chile, pp. 425–434.

Bates, D. W., Kuperman, G. J., Wang, S., Gandhi, T., Kittler, A., Volk, L., Spurr, C., Khorasani, R., Tanasijevic, M. & Middleton, B. (2003), 'Ten commandments for effective clinical decision support: Making the practice of evidence-based medicine a reality', *Journal of the American Medical Informatics Association* **10**, 523–530.

Batet, M., Gibert, K. & Valls, A. (2008), The Data Abstraction Layer as Knowledge Provider for a Medical Multi-Agent System, *in* D. Riaño, ed., 'Knowledge Management for Health Care Procedures. From Knowledge to Global Care AIME 2007 Workshop K4Care 2007', Vol. 4924 of *Lecture Notes in Artificial Intelligence*, pp. 87–100.

Bauer, B. (2001), UML Class Diagrams Revisited in the Context of Agent-Based Systems, *in* 'Revised Papers and Invited Contributions from the Second International Workshop on Agent-Oriented Software Engineering II', Vol. 2222 of *Lecture Notes in Computer Science*, Springer-Verlag / Heidelberg, pp. 101 – 118.

Bauer, B., Müller, J. P. & Odell, J. (2001), Agent UML: A Formalism for Specifying Multiagent Software Systems, *in* 'Agent-Oriented Software Engineering: First International Workshop, AOSE 2000. Revised papers', Vol. 1957 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 109–120.

Becker, M., Heine, C., Herrier, R. & Krempels, K.-H. (2003), OntHoS - an Ontology for Hospital Scenarios, *in* A. Moreno & J. Nealon, eds, 'Applications of Software Agent Technology in the Health Care Domain', Whitestein Series in Software Agent Technologies, Birkhäuser Verlag, Basel, Switzerland, pp. 87–103.

Becker, M. & Krempels, K. H. (2003), Agent Based Scheduling of Operation Theaters, *in* I. Rudomín, J. Vázquez-Salceda & J. L. Díaz de León Santiago, eds, 'e-Health: Application of Computing Science in Medicine and Health Care', Vol. 5, Instituto Politécnico Nacional, Cuernavaca, Mexico, pp. 220–227.

Beda, A., N., G. & Amigoni, F. (2004), Hearth-rate pacing simulation and control via multiagent systems, *in* J. Nealon, A. Moreno, J. Fox & U. Cortés, eds, 'In Proc. of 2nd Workshop on Agents Applied in Health Care in the 16th European conference on Artificial Intelligence, ECAI 2004', Valencia, Spain, pp. 22–30.

Beer, M. & Hill, R. (2006), Using Multi-agent Systems to Manage Community Care, *in* B. Gabrys, R. J. Howlett & L. C. Jain, eds, 'Knowledge-Based Intelligent Information and Engineering Systems, KES 2006', Vol. 4252 of *Lecture Notes in Artificial Intelligence*, Springer Verlag, Bournemouth, UK, pp. 1240–1247.

Beliakov, G., Pradera, A. & Calvo, T. (2007), *Aggregation Functions: A Guide for Practitioners*, Vol. 221 of *Studies in Fuzziness and Soft Computing*, Springer-Verlag Berlin Heidelberg.

Bellifemine, F., Caire, G. & Greenwood, D. (2007), *Developing Multi-Agent Systems with JADE*, Wiley Series in Agent Technology, John Wiley and Sons.

Belton, V. & Stewart, T. J. (2001), *Multiple Criteria Decision Analysis: An Integrated Approach*, Kluwer Academic Publishers.

Berg, D., Ram, P. & Glasgow, J. (2004), SAGEDesktop: An Environment for Testing Clinical Practice Guidelines, *in* '26th Annual Conference of the IEEE Engineering in Medicine and Biology Society, IEBMS 2004', Vol. 2, IEEE Press, San Francisco, USA, pp. 3217–20.

Bernon, C., Cossentino, M. & Pavón, J. (2005), 'Agent-oriented software engineering', *The Knowledge Engineering Review* **20**, 99–116.

Báez-Barranco, J.-A., Stratulat, T. & Ferber, J. (2007), A Unified Model for Physical and Social Environments, *in* D. Weyns & F. Van Parunak, H.and Michel, eds, 'Third International Workshop on Environments for Multi-Agent Systems III, E4MAS 2006', Vol. 4389 of *Lecture Notes in Artificial Intelligence*, Springer Berlin / Heidelberg, pp. 41–50.

Blaser, R., Schnabel, M., Biber, C., Baumlein, M., Heger, O., Beyer, M., Opitz, E., Lenz, R. & Kuhn, K. (2007), 'Improving pathway compliance and clinician performance by using information technology', *International Journal of Medical Informatics* **76**, 151–156.

Bond, C. S. (2007), Nurses and Computers. An International Perspective on Nurses' Requirements, *in* K. A. Kuhn, J. R. Warren & T.-Y. Leong, eds, 'Proceedings of the 12th World Congress on Health (Medical) Informatics, MEDINFO 2007', Vol. 129 of *Studies in Health Technology and Informatics*, IOS Press, Brisbane, Australia, pp. 228–232.

Bordini, R. H., Hübner, J. F. & Wooldridge, M. (2007), *Programming Multi-Agent Systems in AgentSpeak using Jason*, John Wiley and Sons.

Bortolussi, L., Dovier, A. & Fogolari, F. (2005), Multi-agent simulation of protein folding, *in* 'In Proc. of First Workshop on Multi-Agent Systems for Medicine, Computational Medicine, and Bioinformatics, MAS*BIOMED 2005', Utrech, The Netherlands, pp. 91–106.

Boxwala, A. A., Peleg, M., Tu, S. W., Ogunyemi, O., Zeng, Q., Wang, D., Patel, V. L., Greenes, R. A. & Shortliffe, E. H. (2004), 'GLIF3: A Representation Format for Sharable Computer-Interpretable Clinical Practice Guidelines', *Journal of Biomedical Informatics* **37**, 147–161.

Boxwala, A. A., Tu, S., Peleg, M., Zeng, Q., Ogunyemi, O., Greenes, R. A., Shortliffe, E. H. & Patel, V. L. (2001), 'Toward a Representation Format for Sharable Clinical Guidelines', *Journal of Biomedical Informatics* **34**, 157–169.

Braun, L., Wiesman, F., van der Herik, J. & Hasman, A. (2005), Agent Support in Medical Information Retrieval, *in* A. Moreno, ed., 'In Proc. of IJCAI05 Workshop on Agents Applied in Health Care', Edimburgh, Scotland, pp. 16–25.

Bresciani, P., Giorgini, P., Giunchiglia, F., Mylopoulos, J. & Perini, A. (2004), 'TROPOS: An Agent-Oriented Software Development Methodology', *Autonomous Agents and Multi-Agent Systems* **8**(3), 203–236.

Cabana, M. D., Rand, C. S., Powe, N. R., Wu, A. W., Wilson, M. H., Abboud, P. & Rubin, H. (1999), 'Why don't physicians follow clinical practice guidelines? A framework for improvement', *Journal of the American Medical Informatics Association* **282**, 1458–1466.

Caire, G., Coulier, W., Garijo, F. J., Gómez-Sanz, J., Pavón, J., Leal, F., Chainho, P., Kearney, P. E., Stark, J., Evans, R. & Massonet, P. (2001), Agent Oriented Analysis Using

Message/UML , *in* 'Revised Papers and Invited Contributions from the Second International Workshop on Agent-Oriented Software Engineering, AOSE 2001', Vol. 2222 of *Lecture Notes in Computer Science*, Springer-Verlag UK, pp. 119–135.

Calisti, M., Funk, P., Biellman, S. & Bugnon, T. (2003), A Multi-Agent System for Organ Transplant Management, *in* A. Moreno & J. Nealon, eds, 'Applications of Software Agent Technology in the Health Care Domain', Whitestein Series in Software Agent Technologies, Birkhäuser Verlag, Basel, Switzerland, pp. 199–212.

Camarinha-Matos, L. M. & Afsarmanesh, H. (2004), TeleCARE: Collaborative virtual elderly support communities, *in* L. M. Camarinha-Matos, ed., '1st Workshop on TeleCare and Collaborative Virtual Communities in Elderly Care, ICEIS 2004', INSTICC Press, Porto, Portugal.

Campana, F., Annicchiarico, R. & Riaño, D. (2006), Knowledge-based homecare eservices for an ageing europe: D01 - the k4care model, Technical Report Deliverable 1, K4Care Consortium. Available online at http://www.k4care.net [Last visit: 08/06/2008].

Campana, F., Moreno, A., Riaño, D. & Varga, L. Z. (2008), K4Care: Knowledge-Based Homecare e-Services for an Ageing Europe, *in* R. Annicchiarico, U. Cortés & C. Urdiales, eds, 'Agent Technology and e-Health', Whitestein Series in Software Agent Technologies and Autonomic Computing, Birkhäuser Verlag, Basel, Switzerland, pp. 95–116.

Cernuzzi, L., Juan, T., Sterling, L. & Zambonelli, F. (2004), *Methodologies and Software Engineering for Agent Systems*, Springer US, chapter The Gaia Methodology: Basic Concepts and Extensions, pp. 69–88.

Cernuzzi, L. & Rossi, G. (2002), On the Evaluation of Agent Oriented Modeling Methods, *in* 'Proceedings of the Workshop on Agent-Oriented Methodologies, OOPSLA 2002', pp. 21–30.

Cervantes, L., Lee, Y.-S., Yang, H., Ko, S.-h. & Lee, J. (2007), Agent-Based Intelligent Decision Support for the Home Healthcare Environment, *in* M. S. Szczuka, D. Howard, D. Slezak, H.-k. Kim, T.-h. Kim, I.-s. Ko, G. Lee & P. M. Sloot, eds, 'First International Conference Advances in Hybrid Information Technology, ICHIT 2006', Vol. 4413, Springer Berlin / Heidelberg, Jeju Island, Korea, pp. 414–424.

Chella, A., Cossentino, M., Sabatucci, L. & Seidita, V. (2006), 'Agile PASSI: An Agile Process for Designing Agents', *Computer Systems: Science and Engineering. Special issue on "Software Engineering for Multi-Agent Systems"* **21**(2).

Chesani, F., Matteis, P. D., Mello, P., Montal, M. & Storari, S. (2006), A Framework for Defining and Verifying Clinical Guidelines: A Case Study on Cancer Screening, *in* F. Esposito, Z. W. Ras, D. Malerba & G. Semeraro, eds, 'Proceedings of 16th International Symposium, ISMIS 2006', Vol. 4203 of *Lecture Notes on Artificial Intelligence*, Springer Berlin / Heidelberg, Bari, Italy, pp. 338–343.

Choi, J., Currie, L. M., Wang, D. & Bakken, S. (2007), 'Encoding a clinical practice guideline using guideline interchange format: A case study of a depression screening and management guideline', *International Journal of Medical Informatics* **76S**, S302–S307.

Ciampolini, A., Mello, P. & Storari, S. (2004), A multi-agent system for medical coordination services synergy and coordination, *in* J. Nealon, A. Moreno, J. Fox & U. Cortés, eds, 'In Proc. of 2nd Workshop on Agents Applied in Health Care in the 16th European conference on Artificial Intelligence, ECAI 2004', Valencia, Spain, pp. 38–46.

Ciccarese, P., Caffi, E., Boiocchi, L., Quaglini, S. & Stefanelli, M. (2004), A guideline management system, *in* M. Fieschi, E. Coiera & Y.-C. Li, eds, 'Proceedings of 11th World Congress of the International Medical Informatics Association, MEDINFO 2004', Vol. 107 of *Studies in Health Technology and Informatics*, IOS Press, San Francisco, USA, pp. 28–32.

Ciccarese, P., Caffi, E., Quaglini, S. & Stefanelli, M. (2005), 'Architectures and Tools for innovative Health Information Systems: the Guide Project', *International Journal of Medical Informatics* **74**, 553 – 562.

Clercq, P. A. d., Blom, J. A., Korsten, H. & Hasman, A. (2004), 'Approaches for creating computer-interpretable guidelines that facilitate decision support', *Artificial Intelligence in Medicine* **31**, 1–27.

Coiera, E. (2003*a*), *Guide to Health Informatics*, Hodder Arnold.

Coiera, E. (2003*b*), Healthcare terminologies and classification systems, *in* 'Guide to Health Informatics', Hodder Arnold, pp. 201–216.

Corchado, J. M., Bajo, J. & Abraham, A. (2008), 'GerAmi: Improving Healthcare Delivery in Geriatric Residences', *IEEE Intelligent Systems* **23**, 19–25.

Corcho, O., Fernández-López, M. & Gómez-Pérez, A. (2003), 'Methodologies, tools and languages for building ontologies: Where is their meeting point?', *Data Knowledge Engineering* **46**(1), 41–64.

Cortés, U., Annicchiarico, R., Urdiales, C., Barrué, C., Martínez, A., Villar, A. & Caltagirone, C. (2008), Supported Human Autonomy for Recovery and Enhancement of Cognitive and Motor Abilities Using Agent Technologies, *in* R. Annicchiarico, U. Cortés & C. Urdiales, eds, 'Agent Technology and e-Health', Whitestein Series in Software Agent Technologies and Autonomic Computing, Birkhäuser Verlag, Basel, Switzerland, pp. 117–140.

Cortés, U., Urdiales, C., Annichiarico, R., Barrué, C., Martínez, A. & Caltagirone, C. (2007), *Advanced Computational Intelligence Paradigms in Healthcare - 1*, Vol. 48 of *Studies in Computational Intelligence*, Springer-Verlag, pp. 165–187.

Cossentino, M. (2005), *Agent-Oriented Methodologies*, Idea Group, chapter From Requirements to Code with the PASSI Methodology, pp. 79–106.

Cossentino, M., Gaglio, S., Sabatucci, L. & Seidita, V. (2005), The PASSI and Agile PASSI MAS Meta-models Compared with a Unifying Proposal, *in* '4th International Central and Eastern European Conference on Multi-Agent Systems, CEEMAS 2005', Vol. 3690 of *Lecture Notes in Artificial Intelligence*, Springer Berlin / Heidelberg, pp. 183–192.

Cuesta, P., Gómez, A., González, J. C. & Rodríguez, F. J. (2004), Evaluating agent oriented software methodologies to propose MESMA, *in* V. Botti & E. Corchado, eds, 'Proceedings of 3rd International Workshop on Practical Applications of Agents and Multiagent Systems, IWPAAMS 2004', Universidad de Burgos, pp. 103–114.

Cuesta, P., Gómez, A., González, J. & Rodríguez, F. J. (2005), 'Developing a multiagent system for mobile devices', *Revista Iberoamericana de Inteligencia Artificial* **9**(25), 49–57.

Curé, O. (2003), Designing Patient-Oriented Systems with Semantic Web Technologies, *in* '16th IEEE Symposium on Computer-Based Medical Systems, CBMS 2003', IEEE Press, New York, NY, USA, pp. 195–200.

Dam, K. H. & Winikoff, M. (2004), Comparing Agent-Oriented Methodologies, *in* P. Giorgini, B. Henderson-Sellers & M. Winikoff, eds, 'Agent Oriented Information Systems. 5th International Bi-Conference Workshop, AOIS 2003. Revised Selected Papers.', Vol. 3030 of *Lecture Notes in Artificial Intelligence*, Springer Berlin / Heidelberg, pp. 78–93.

Dastani, M., Dignum, V. & Dignum, F. (2002), Organizations and Normative Agents, *in* H. Shafazand & A. M. Tjoa, eds, 'Information and Communication Technology. First EurAsian Conference, EurAsia-ICT 2002', Vol. 2510 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 982–989.

Davis, D., Goldman, J. & Palda, V. A. (2007), *Handbook on Clinical Practice Guidelines*, Canadian Medical Association, Ottawa, CA. Available on-line http://mdm.ca/cpgsnew/cpgs/index.asp [last visit 10/10/2007].

Davis, J. & Blanco, R. (2005), 'Analysis and Architecture of Clinical Workflow Systems using Agent-Oriented Lifecycle Models', *Intelligent Paradigms for Healthcare Enterprises* **184**, 67–119.

Decker, K. S. (1996), *Foundations of Distributed Artificial Intelligence*, John Wiley & Sons, chapter TAEMS: A Framework for Environment Centered Analysis and Design of Coordination Mechanisms, pp. 429–448.

Delgado, M., Herrera, F., Herrera-Viedma, E. & Martínez, L. (1998), 'Combining numerical and linguistic information in group decision making', *Inf Sci* **107**, 177–194.

DeLoach, S. A. (2001), Analysis and Design using MaSE and agentTool, *in* '12th Midwest Artificial Intelligence and Cognitive Science Conference, MAICS 01', Oxford, Ohio.

DeLoach, S. A. (2004), *Methodologies and Software Engineering for Agent Systems*, Multiagent Systems, Artificial Societies, and Simulated Organizations, Springer US, chapter The MaSE Methodology, pp. 107–125.

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

Dignum, V. (2004), A Model for Organizational Interaction: Based on Agents, Founded in Logic, PhD thesis, Universiteit Utrecht.

Dignum, V. & Dignum, F. (2005), Task and Social Coordination in Agent Organizations, *in* 'Proceedings of the fourth International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS 2005', ACM Press, Utrech, The Netherlands, pp. 1183–84.

Dignum, V., Meyer, J.-J. & Weigand, H. (2002), An Organizational-Oriented Model for Agent Societies, *in* 'First International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS 2002', Vol. Part 2, ACM Press, Bologna, Italy, pp. 694–5.

Dignum, V., Vázquez-Salceda, J. & Dignum, F. (2004), OMNI: Introducing Social Structure, Norms and Ontologies into Agent Organizations , *in* R. H. Bordini, M. Dastani, J. Dix & A. E. F. Seghrouchni, eds, 'Second International Workshop Programming Multi-Agent Systems, ProMAS 2004', Vol. 3346 of *Lecture Notes in Artificial Intelligence*, Springer Berlin / Heidelberg, pp. 181–198.

Ding, L., Finin, T., Joshi, A., Pang, R., Cost, R., Peng, Y., Reddivari, P., Doshi, V. & Sachs, J. (2004), Swoogle: A Search and Metadata Engine for the Semantic Web, *in* 'Proceedings of the 13th ACM Conference on Information and Knowledge Management, CIKM 2004', ACM Press, Washington DC., USA, pp. 652–659.

Dixon, B. E., Zafar, A. & McGowan, J. J. (2007), Development of a Taxonomy for Health Information Technology, *in* K. A. Kuhn, J. R. Warren & T.-Y. Leong, eds, 'Proceedings of the 12th World Congress on Health (Medical) Informatics, MEDINFO 2007', Vol. 129 of *Studies in Health Technology and Informatics*, IOS Press, Brisbane, Australia, pp. 616–620.

Domínguez, D., Grasso, F., Miller, T. & Seraffin, R. (2006), PIPS: An Integrated Environment for Health Care Delivery and Healthy Lifestyle Support, *in* '4th Workshop on Agents Applied in Health Care in conjunction with the 17th European Conference on Artificial Intelligence, ECAI 2006', Riva del Garda, Italy, pp. 81–90.

Dube, K. (2004), A Generic Approach to Supporting the Management of Computerised Clinical Guidelines and Protocols, PhD thesis, Institute of Technology, Dublin, Ireland.

Dube, K., Mansour, E. & Wu, B. (2005), Supporting Collaboration and Information Sharing in Computer-Based Clinical Guideline Management, *in* '18th IEEE Symposium on Computer-Based Medical Systems, CBMS 2005', IEEE Press, Dublin, Ireland, pp. 232–237.

Dube, K. & Wu, B. (2006), An Active Database Approach to Computerised Clinical Guideline Management, *in* J. Cordeiro & J. Filipe, eds, 'Computer Supported Activity Coordination, Proceedings of the 3rd International Workshop on Computer Supported Activity Coordination, CSAC 2006', INSTICC Press, Paphos, Cyprus, pp. 107–115.

EC (2007), White Paper. Together for Health: A Strategic Approach for the EU 2008-2013, Technical report, European Commission. Available at http://ec.europa.eu/health-eu/doc/whitepaper_-en.pdf [last visit 21/04/2008].

Elkin, P. L., Peleg, M., Lacson, R., Bernstam, E., Tu, S. W., Boxwala, A., Greenes, R. A. & Shortliffe, E. H. (2001), 'Toward Standardization of Electronic Guideline Representation', *MD Computing* **17**, 39–44.

Fensel, D. (2001), *Ontologies: Silver Bullet for Knowledge Management and Electronic Commerce*, 1st edn, Springer Verlag.

Ferber, J. & Gutknecht, O. (1998), AALAADIN: A Meta-Model for the Analysis and Design of Organizations in Multi-Agent Systems, *in* Y. Demazeau, ed., 'International Conference on Multi-Agent Systems, ICMAS 1998', IEEE Press, pp. 41–50.

Ferber, J., Gutknecht, O. & Michel, F. (2004), From Agents to Organizations: an Organizational View of Multi-AgentSystems, *in* P. Giorgini, J. P. Müller & J. Odell, eds, '4th International Workshop on Agent-Oriented Software Engineering IV, AOSE 2003', Vol. 2935 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 214–230.

Ferber, J., Michel, F. & Báez-Barranco, J.-A. (2005), AGRE: Integrating Environments with Organizations, *in* D. Weyns, H. V. D. Parunak & F. Michel, eds, 'First International Workshop on Environments for Multi-Agent Systems, E4MAS 2004', Vol. 3374 of *Lecture Notes in Artificial Intelligence*, Springer Berlin / Heidelberg, pp. 48–56.

Field, M. J. & Lohr, K. N., eds (1990), *Clinical Practice Guidelines: Directions for a New Program*, National Academy Press, Washington, DC.

Figueira, J., Greco, S. & Ehrgott, M., eds (2005), *Multiple Criteria Decision Analysis: State of the Art Surveys*, Vol. 78 of *International Series in Operations Research and Management Science*, Springer New York.

FIPA (2002*a*), FIPA Abstract Architecture Specification, Technical Report SC00001L, Foundation for Intelligent Physical Agents. Available at http://www.fipa.org/specs/fipa00031/ [last visit 26/04/2008].

FIPA (2002*b*), FIPA Brokering Interaction Protocol, Technical Report SC00033H, Foundation for Intelligent Physical Agents. Available at http://www.fipa.org/specs/fipa00033/ [last visit 16/05/2008].

FIPA (2002*c*), FIPA Communicative Act Library Specification, Technical Report SC00037J, Foundation for Intelligent Physical Agents. Available at http://www.fipa.org/specs/fipa00037/ [last visit 26/04/2008].

FIPA (2002*d*), FIPA English Auction Interaction Protocol Specification, Technical Report SC00031F, Foundation for Intelligent Physical Agents. Available at http://www.fipa.org/specs/fipa00031/ [last visit 26/04/2008].

FIPA (2002*e*), FIPA Query Interaction Protocol Specification, Technical Report SC00027H, Foundation for Intelligent Physical Agents. Available at http://www.fipa.org/specs/fipa00027/ [last visit 16/05/2008].

Flatley, P. (2007), *Patient-Focused Technology and the Health Care Delivery System*, Health Informatics, Springer New York, pp. 1–6.

Foster, D., McGregor, C. & El-Masri, S. (2005), A survey of agent-based intelligent decision support systems to support clinical management and research, *in* 'In Proc. of First Workshop on Multi-Agent Systems for Medicine, Computational Medicine, and Bioinformatics, MAS*BIOMED 2005', Utrech, The Netherlands, pp. 16–34.

Fox, J., Alabassi, A., Patkar, V., Rose, T. & Black, E. (2006), 'An ontological approach to modelling tasks and goals', *Computers in Biology and Medicine* **36**, 837–856.

Fox, J., Beveridge, M. & Glasspool, D. (2003*a*), 'Understanding Intelligent Agents: Analysis and Synthesis', *AI Communications* **16**, 139–152.

Fox, J., Beveridge, M. & Glasspool, D. (2003*b*), 'Understanding intelligent agents: Analysis and Synthesis', *AI Communications* **16**(3), 139–152.

Fox, J. & Das, S. (2000), *Safe and Sound*, AAAI and MIT Press.

Fox, J., Glasspool, D., Grecu, D., Modgil, S., South, M. & Patkar, V. (2007), 'Argumentation-Based Inference and Decision Making - A Medical Perspective', *IEEE Intelligent Systems* **22**.

Frank, U. (1998), Evaluating Modelling Languages: Relevant Issues, Epistemological Challenges And A Preliminary Research Framework, Technical Report 15, Universität Koblenz-Landau. Available on-line: http://citeseer.ist.psu.edu/255724.html [Last visit: 23/11/2007].

Gennari, J. H., Musen, M. A., Fergerson, R. W., Grosso, W. E., Crubzy, M., Eriksson, H., Noy, N. F. & Tu, S. W. (2003), 'The Evolution of Protégé: An Environment for Knowledge-Based Systems Development', *International Journal of Human-Computer Studies* **58**(1), 89–123.

Georgeff, M., Pell, B., Pollack, M., Tambe, M. & Wooldridge, M. (1999), The Belief-Desire-Intention Model of Agency, *in* J. Müller, M. P. Singh & A. S. Rao, eds, 'Proceedings of the 5th International Workshop on Intelligent Agents V : Agent Theories, Architectures, and Languages, ATAL 98', number 1555 *in* 'Lecture Notes in Computer Science', Springer Verlag / Heidelberg, pp. 1–10.

Georgiou, A., Williamson, M., Westbrook, J. I. & Ray, S. (2007), 'The impact of computerised physician order entry systems on pathology services: A systematic review', *International Journal of Medical Informatics* **76**, 514–529.

Ghinea, G., Magoulas, G. D. & Frank, A. O. (2004), 'Intelligent multimedia communication for enhanced medical e-collaboration in back pain treatment', *Transactions of the Institute of Measurement and Control* **26**(3), 223–244.

Giorgini, P., Kolp, M., Mylopoulos, J. & Pistore, M. (2004), *Methodologies and Software Engineering for Agent Systems*, Vol. 11 of *Multiagent Systems, Artificial Societies, and Simulated Organizations*, Springer US, chapter The Tropos Methodology: An Overview, pp. 89–106.

Giorgini, P., Kolp, M., Mylopoulus, J. & Castro, J. (2005), *Agent-Oriented Methodologies*, Idea Group, chapter Tropos: A Requirements-Driven Methodology for Agent-Oriented Software, pp. 20–45.

Gómez-Alonso, C., Isern, D. & Moreno, A. (2007), Software Engineering Methodologies to Develop Multi-Agent Systems: State-of-the-art, Research Report DEIM-RR-07-003, Universitat Rovira i Virgili, Tarragona, Catalonia. Available at http://deim.urv.cat/recerca/reports/DEIM-RR-07-003.pdf [last visit 2/4/2008].

Gómez-Pérez, A., Fernández-López, M. & Corcho, O. (2003), *Ontological Engineering: with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web*, Springer-Verlag New York, LLC.

Gómez-Sanz, J. (2002), Modelado de Sistemas Multiagente, PhD thesis, Universidad Complutense de Madrid. Departamento de Sistemas Informáticos y Programación.

Gómez-Sanz, J. (2003), 'Metodologías para el desarrollo de sistemas multi-agente', *Revista Iberoamericana de Inteligencia Artificial* **18**(3), 51–63.

Gómez-Sanz, J., Gervais, M.-P. & Weiss, G. (2004), *Methodologies and Software Engineering for Agent Systems*, Vol. 11 of *Multiagent Systems, Artificial Societies, and Simulated Organizations*, Springer US, chapter A Survey on Agent-Oriented Oriented Software Engineering Research, pp. 33–62.

Gómez-Sanz, J. & Pavón, J. (2006), Implementing Multi-agent Systems Organizations with INGENIAS, *in* R. H. Bordini, M. Dastani, J. D. Amal & E. F. Seghrouchni, eds, 'Proceedings of the Third International Workshop, ProMAS 2005', Vol. 3862 of *Lecture Notes in Artificial Intelligence*, Springer Berlin / Heidelberg, pp. 236–251.

Godó, L., Puyol-Guart, J., Sabater, J., Torra, V., Barrufet, P. & Fàbregas, X. (2003), 'A multiagent system approach for monitoring the prescription of restricted use antibiotics', *AI in Medicine* **27**(3), 259–282.

González Vélez, H., Mier, M., Julià-Sapé, M., Arvanitis, T. N., García-Gómez, J. M., Robles, M., Lewis, P. H., Dasmahapatra, S., Dupplaw, D., Peet, A., Arús, C., Celda, B., Van Huffel, S. & Lluch-Ariet, M. (2008), 'HealthAgents: distributed multi-agent brain tumor diagnosis and prognosis', *Applied Intelligence* **in press**.

Gong, Y., Zhu, M., Li, J., Turley, J. P. & Zhang, J. (2007), Clinical Communication Ontology for Medical Errors, *in* K. A. Kuhn, J. R. Warren & T.-Y. Leong, eds, 'Proceedings of the 12th World Congress on Health (Medical) Informatics, MEDINFO 2007', Vol. 129 of *Studies in Health Technology and Informatics*, IOS Press, Brisbane, Australia, pp. 1007–11.

Grabisch, M., Orlovski, S. & Yager, R. (1999), Fuzzy aggregation of numerical preferences, *in* R. Slowinski, ed., 'Fuzzy sets in decision analysis, operations research and statistics', Kluwer Academic, pp. 31–68.

Gutknecht, O., Ferber, J. & Michel, F. (2001), Integrating tools and infrastructures for generic multiagent systems, *in* E. André, S. Sen, C. Frasson & J. P. Müller, eds, 'Fifth international conference on Autonomous agents, AGENTS 2001', ACM Press, pp. 441–448.

Hajnal, A., Isern, D., Moreno, A., Pedone, G. & Varga, L. Z. (2007), Knowledge Driven Architecture for Home Care, *in* H.-D. Burkhard, G. Lindemann, R. Verbrugge & L. Z.

Varga, eds, 'Multi-Agent Systems and Applications V. 5th International Central and Eastern European Conference on Multi-Agent Systems, CEEMAS 2007', Vol. 4696 of *Lecture Notes in Artificial Intelligence*, Springer Berlin / Heidelberg, Leipzig, Germany, pp. 173–182.

Hannoun, M., Boissier, O., Sichman, J. S. & Sayettat, C. (2000), MOISE: An Organizational Model for Multi-agent Systems, *in* M. C. Monard & J. S. Sichman, eds, 'Advances in Artificial Intelligence: Proceedings of International Joint Conference, 7th Ibero-American Conference on AI, 15th Brazilian Symposium on AI, IBERAMIA-SBIA 2000', Vol. 1952 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 156–165.

Hart, D. (2003), 'Risk Management, Clinical Guidelines, Clinical Pathways and Health Law', *European Journal of Health Law* **10**(3), 219–222.

Hübner, J. F., Boissier, O. & Sichman, J. S. (2006), Programming MAS reorganisation with Moise+, *in* 'Seminar on Foundations and Practice of Programming Multi-Agent Systems, 2006', number 06261 *in* 'Dagstuhl Seminar Proceedings', Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany. Available at http://moise.sourceforge.net/doc/dagsthul06-slides.pdf [Last visit: 23/11/2007].

Hübner, J. F., Sichman, J. S. & Boissier, O. (2002*a*), A Model for the Structural, Functional, and Deontic Specification of Organizations in Multiagent Systems, *in* G. Bittencourt & G. Ramalho, eds, 'Proc. of 16th Brazilian Symposium on Artificial Intelligence, SBIA 2002', Vol. 2507 of *Lecture Notes in Artificial Intelligence*, Springer Berlin / Heidelberg, Porto de Galinhas/Recife, Brazil, pp. 439–448.

Hübner, J. F., Sichman, J. S. & Boissier, O. (2002*b*), Moise+: Towards a Structural, Functional, and Deontic Model for MAS Organization, *in* 'Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems, AAMAS 2002', ACM Press, pp. 501–502.

Hübner, J. F., Sichman, J. S. & Boissier, O. (2005), S-Moise+: A Middleware for developing Organised Multi-Agent Systems, *in* O. Boissier, J. A. Padget, V. Dignum, G. Lindemann, E. T. Matson, S. Ossowski, J. S. Sichman & J. Vázquez-Salceda, eds, 'Workshops on Agents, Norms and Institutions for Regulated Multi-Agent Systems, ANIREM 2005, and Organizations in Multi-Agent Systems, OOOP 2005. Revised Selected Papers.', Vol. 3913 of *Lecture Notes in Computer Science*, pp. 64–78.

Hübner, J. F., Sichman, J. S. & Boissier, O. (2007), 'Developing organised multi-agent systems using the Moise+ model: Programming issues at the system and agent levels', *International Journal of Agent-Oriented Software Engineering* **in press**.

Henderson-Sellers, B. & Giorgini, P. (2005), *Agent-Oriented Methodologies*, Idea Group, chapter Agent-Oriented Methodologies: An Introduction, pp. 1–19.

Herrera, F. & Herrera-Viedma, E. (2000), 'Linguistic decision analysis: steps for solving decision problems under linguistic information', *Fuzzy Sets and Systems* **115**, 67–82.

Herrera, F., Herrera-Viedma, E. & Martínez, L. (2002), Representation Models for Aggregating Linguistic Information: Issues and Analysis, *in* T. Calvo, G. Mayor & R. Mesiar, eds, 'Aggregation Operators. New Trends and Applications', Vol. 97 of *Studies in Fuzziness and Soft Computing*, Physica Verlag, pp. 245–259.

Herrler, R. & Puppe, F. (2005), Evaluation and Optimization of Clinical Processes with Multi-Agent Simulation, *in* A. Moreno, ed., 'In Proc. of IJCAI05 Workshop on Agents Applied in Health Care', Edimburgh, Scotland, pp. 26–32.

Hill, R., Polovina, S. & Beer, M. (2005), Managing Community Healthcare Information in a Multi-Agent System Environment, *in* 'In Proc. of First Workshop on Multi-Agent Systems for Medicine, Computational Medicine, and Bioinformatics, MAS*BIOMED 2005', Utrech, The Netherlands, pp. 35–49.

Hájek, P. & Valdés, J. (1994), 'An analysis of MYCIN-like expert systems', *Mathware and Soft Computing* **1**, 45–68.

Holbrook, A., Keshavjee, K., Troyan, S., Pray, M. & Ford, P. T. (2003), 'Applying methodology to electronic medical record selection', *International Journal of Medical Informatics* **71**, 43–50.

Hommersom, A., Groot, P., Lucas, P. J., Balser, M. & Schmitt, J. (2007), 'Verification of Medical Guidelines Using Background Knowledge in Task Networks', *IEEE Transactions on Knowledge and Data Engineering* **19**(3), 832–846.

Horling, B. & Lesser, V. (2005), 'A Survey of Multi-Agent Organizational Paradigms', *The Knowledge Engineering Review* **19**(4), 281 – 316.

Hospers, M., Kroezen, E., Nijholt, A., Akker, R. & Heylen, D. (2003), An Agent-Based Intelligent Tutoring System for Nurse Education, *in* A. Moreno & J. Nealon, eds, 'Applications of Software Agent Technology in the Health Care Domain', Whitestein Series in Software Agent Technologies, Birkhäuser Verlag, Basel, Switzerland, pp. 143–160.

Hoyt, R., Sutton, M. & Yoshihashi, A. (2007), *Medical Informatics. Practical Guide for the Healthcare Professional 2007*, University of West Florida, Pensacola, Florida, USA.

Hrabak, K. M., Campbell, J. R., Tu, S. W., McClure, R. & Weida, R. (2007), Creating Interoperable Guidelines: Requirements of Vocabulary Standards in Immunization Decision Support, *in* K. A. Kuhn, J. R. Warren & T.-Y. Leong, eds, 'Proceedings of the 12th World Congress on Health (Medical) Informatics, MEDINFO 2007', Vol. 129, IOS Press, Brisbane, Australia, pp. 930–934.

Hripcsak, G., Clayton, P. D., Jenders, R. A., Cimino, J. J. & Johnson, S. B. (1996), 'Design of a clinical event monitor', *Computers and Biomedical Research* **29**(3), 194–221.

Huget, M. F. (2004), 'Agent UML Notation for Multiagent System Design', *IEEE Internet Computing* **8**, 63–81.

Huget, M.-P. & Odell, J. (2004), Representing Agent Interaction Protocols with Agent UML, *in* P. Giorgini, J. P. Müller & J. Odell, eds, '5th International Workshop, AOSE 2004. Revised Selected Papers', Vol. 3382 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 16–30.

Huget, M.-P., Odell, J. & Bauer, B. (2004), *Methodologies and Software Engineering for Agent Systems*, Multiagent Systems, Artificial Societies, and Simulated Organizations, Springer US, chapter The AUML Approach, pp. 237–257.

IAPO (2006), Iapo patient-centred healthcare review, Technical report, International Alliance of Patients' Organizations. Available on-line at http://www.patientsorganizations.org/pchreview [last visit 22/04/2008].

IBM (1997), 'Object Constraint Language Specification, v.1.1', Resource page. Available at ftp://ftp.omg.org/pub/docs/ad/97-08-08.pdf [last visit [Last visit: 11/11/2007]].

Iglesias, C. A. & Garijo, M. (1999), UER Technique: Conceptualisation for Agent-Oriented Development, *in* '3rd World Multiconference on Systemics, Cybernetics and Informatics, SCI 1999', Vol. 5, International Institute of Informatics and Systemics, pp. 535–540.

Iglesias, C. A. & Garijo, M. (2005), *Agent-Oriented Methodologies*, Idea Group, chapter The Agent-Oriented Methodology MAS-CommonKADS, pp. 46–78.

Iglesias, C. A., Garijo, M. & González, J. C. (2000), A Survey of Agent-Oriented Methodologies, *in* J. P. Müller, M. P. Singh & A. S. Rao, eds, '5th International Workshop on Intelligent Agents V : Agent Theories, Architectures, and Languages, ATAL 1998', Vol. 1555 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 317–330.

InferMed (2007), Arezzo Technical White Paper, Technical report, InferMed, Ltd. Available on-line http://www.infermed.com/ [last visit 10/10/2007].

Isern, D., Gómez-Alonso, C. & Moreno, A. (2008), 'Methodological development of a multiagent system in the healthcare domain', *Communications of SIWN* **3**, 65–68.

Isern, D., Millan, M., Moreno, A., Pedone, G. & Varga, L. (2008*a*), Home Care Individual Intervention Plans in the K4Care Platform, *in* '21st IEEE International Symposium on Computer-Based Medical Systems, IEEE CBMS 2008', IEEE Press, Jyväskylä, Finland, pp. 455–457.

Isern, D., Millan, M., Moreno, A., Pedone, G. & Varga, L. (2008*b*), Home Care Personalisation with Individual Intervention Plans, *in* D. Riaño, ed., 'Proceedings of the Knowledge Management for Healthcare Processes Workshop in conjunction with the 18th European Conference on Artificial Intelligence, ECAI 2008', Patras, Greece, pp. 50–54.

Isern, D., Millan, M., Moreno, A., Pedone, G. & Varga, L. Z. (2008*c*), Agent-based execution of Individual Intervention Plans, *in* A. Moreno, R. Annichiaricho & U. Cortés, eds, 'Workshop Agents Applied in Health Care, collocated in AAMAS 2008', Estoril, Portugal, pp. 31–40.

Isern, D. & Moreno, A. (2004*a*), Agent-based careflow using CPGs, *in* J. Vitrià, P. Radeva & I. Aguiló, eds, 'Research and Development (Proceedings of Setè Congrés Català d'Intel·ligència Artificial, CCIA 2004', Vol. 113 of *Recent Advances in Artificial Intelligence*, IOS Press, Barcelona, Catalonia, pp. 293–300.

Isern, D. & Moreno, A. (2004*b*), Distributed guideline-based health care system, *in* '4th International Conference on Intelligent Systems Design and Applications, ISDA 2004', IEEE Press, Budapest, Hungary, pp. 145–150.

Isern, D. & Moreno, A. (2006), Computer-Based Management of Clinical Guidelines: A Survey, *in* '4th Workshop on Agents Applied in Health Care in conjunction with the 17th European Conference on Artificial Intelligence, ECAI 2006', IOS Press, Riva del Garda, Italy, pp. 71–80.

Isern, D. & Moreno, A. (2008*a*), A Distributed Platform of Medical Services: HeCaSe2, *in* 'Proceedings of the System Demonstrations session of 18th European Conference on Artificial Intelligence, ECAI 2008', IOS Press, Patras, Greece, pp. 5–6.

Isern, D. & Moreno, A. (2008*b*), 'Computer-Based Execution of Clinical Guidelines: A Review', *International Journal of Medical Informatics* **77**(12), 787–808.

Isern, D., Moreno, A., Pedone, G. & Varga, L. Z. (2008), An Intelligent Platform to Provide Home Care Services, *in* 'Knowledge Management for Health Care Procedures. From Knowledge to Global Care AIME 2007 Workshop K4CARE 2007', Vol. 4924, Springer Berlin / Heidelberg, pp. 149–160.

Isern, D., Sánchez, D. & Moreno, A. (2007), HeCaSe2: A Multi-Agent Ontology-Driven Guideline Enactment Engine, *in* H.-D. Burkhard, G. Lindemann, R. Verbrugge & L. Z. Varga, eds, 'Multi-Agent Systems and Applications V. 5th International Central and Eastern European Conference on Multi-Agent Systems, CEEMAS 2007', Vol. 4696 of *Lecture Notes in Artificial Intelligence*, Springer Berlin / Heidelberg, Leipzig, Germany, pp. 322–324.

Isern, D., Sánchez, D. & Moreno, A. (2007*a*), Agents and clinical guidelines: Filling the Semantic Gap, *in* C. Angulo & L. Godó, eds, 'Artificial Intelligence Research and Development, CCIA 2007', Vol. 126 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, Andorra la Vella, Andorra, pp. 67–76.

Isern, D., Sánchez, D. & Moreno, A. (2007*b*), An Ontology-Driven Agent-Based Clinical Execution Engine, *in* R. Bellazzi, A. Abu-Hanna & J. Hunter, eds, '11th Conference on Artificial Intelligence in Medicine, AIME 2007', Vol. 4594 of *Lecture Notes in Artificial Intelligence*, Springer Berlin / Heidelberg, Amsterdam, The Netherlands, pp. 49–53.

Isern, D., Sánchez, D., Moreno, A. & Valls, A. (2003*a*), HeCaSe: An Agent-Based System to Provide Personalised Medical Services, *in* 'Proc. of Workshop Agentes Inteligentes en el Tercer Milenio en X Conferencia de la Asociación Española para la Inteligencia Artificial, CAEPIA 2003', Donosti, Vasque Country, pp. 1–10.

Isern, D., Sánchez, D., Moreno, A. & Valls, A. (2003*b*), HeCaSe: Provision of Secure Personalised Medical Services, *in* 'Proceedings of the First European Workshop on Multi-Agent Systems, EUMAS 2003', Oxford, UK.

Isern, D., Valls, A. & Moreno, A. (2006*a*), Learning the user's preferences for multiple criteria ranking, *in* 'XIII Congreso Español sobre Tecnologías y Lógica Fuzzy, ESTYLF 2006', Ciudad Real, Spain, pp. 325–330.

Isern, D., Valls, A. & Moreno, A. (2006*b*), Using aggregation operators to personalize agent-based medical services, *in* B. Gabrys, R. J. Howlett & L. C. Jain, eds, 'Knowledge-Based Intelligent Information and Engineering Systems, KES 2006', Vol. 4252 of *Lec-*

ture Notes in Artificial Intelligence, Springer Verlag, Bournemouth, UK, pp. 1256–1263.

Jacobson, I., Booch, G. & Rumbaugh, J. (1999), *The unified software development process*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Jennings, N. R. (1996), Coordination techniques for distributed artificial intelligence, *in* G. M. P. O'Hare & N. R. Jennings, eds, 'Foundations of distributed artificial intelligence', John Wiley Sixth-Generation Computer Technology Series, John Wiley and Sons, Inc, New York, NY, USA, pp. 187–210.

Kamisalic, A., Riaño, D., Real, F. & Welzer, T. (2007), Temporal Constraints Approximation from Data about Medical Procedures, *in* '20th IEEE International Symposium on Computer-Based Medical Systems, CBMS 2007', IEEE Press, Maribor, Slovenia, pp. 581–588.

Kelly, S., Lyytinen, K. & Rossi, M. (1996), MetaEdit+: A Fully Configurable Multi-User and Multi-Tool CASE and CAME Environment, *in* P. Constantopoulos, J. Mylopoulos & Y. Vassiliou, eds, 'Advanced Information Systems Engineering. Proc. of 8th International Conference, CAiSE 1996', Vol. 1080 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, Heraklion, Crete, Greece, pp. 1–21.

Kim, H. & Chan, P. (2008), 'Learning implicit user interest hierarchy for context in personalization', *Applied Intelligence* **28**, 153–166.

Kirn, S., Heine, C., Herrier, R. & Krempels, K.-H. (2003), Agent-Hospital - a Framework for Clinical Applications in AgentCities, *in* A. Moreno & J. Nealon, eds, 'Applications of Software Agent Technology in the Health Care Domain', Whitestein Series in Software Agent Technologies, Birkhäuser Verlag, Basel, Switzerland, pp. 67–85.

Kolp, M., Giorgini, P. & Mylopoulos, J. (2006), 'Multi-Agent Architectures as Organizational Structures', *Autonomous Agents and Multi-Agent Systems* **13**(1), 3–25.

Kostkova, P., Mani-Saada, J., Madle, G. & Weinberg, J. R. (2003), Agent-Based Up-to-date Data Management in National electronic Library for Communicable Disease, *in* A. Moreno & J. Nealon, eds, 'Applications of Software Agent Technology in the Health Care Domain', Whitestein Series in Software Agent Technologies, Birkhäuser Verlag, Basel, Switzerland, pp. 105–124.

Kumar, A., Ciccarese, P., Smith, B. & Piazza, M. (2004), *Ontologies in Medicine*, Vol. 102 of *Studies in Health Technology and Informatics*, IOS Press, chapter Context-Based Task Ontologies for Clinical Guidelines, pp. 81–94.

Kumar, A., Quaglini, S., Stefanelli, M., Ciccarese, P. & Caffi, E. (2003), 'Modular representation of the guideline text: An approach for maintaining and updating the content of medical education', *Medical Informatics and the Internet in Medicine* **28**(2), 99–115.

Kumar, A., Quaglini, S., Stefanelli, M., Ciccarese, P., Caffi, E. & Boiocchi, L. (2002), 'A framework for representing and executing a clinical practice guideline for the management of high blood pressure in pregnancy', *Technology and Health Care* **10**, 517–519.

Laleci, G. B., Dogac, A., Olduz, M., Tasyurt, I., Yuksel, M. & Okcan, A. (2008), SAPHIRE: A Multi-Agent System for Remote Healthcare Monitoring through Computerized Clinical Guidelines, *in* R. Annicchiarico, U. Cortés & C. Urdiales, eds, 'Agent Technology and e-Health', Whitestein Series in Software Agent Technologies and Autonomic Computing, Birkhäuser Verlag, Basel, Switzerland, pp. 25–54.

Lanzola, G., Gatti, L., Falasconi, S. & Stefanelli, M. (1999), 'A framework for building cooperative software agents in medical applications', *Artificial Intelligence in Medicine* **16**(3), 223–249.

Lau, L. M. & Shakib, S. (2005), Towards Data Interoperability: Practical Issues in Terminology Implementation and Mapping, *in* 'Proceedings of Thirteenth National Health Informatics Conference, HIC 2005', Health Informatics Society of Australia, Melbourne, Australia, pp. 208–213.

Lenz, R., Blaser, R., Beyer, M., Heger, O., Biber, C., Bã´lein, M. & Schnabel, M. (2007), 'IT support for clinical pathways – Lessons learned', *International Journal of Medical Informatics* **76**(S3), S397–S402.

Lenz, R. & Reichert, M. (2005), IT Support for Healthcare Processes, *in* W. M. van der Aalst, B. Benatallah, F. Casati & F. Curbera, eds, '3rd International Conference, BPM 2005', Vol. 3649 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, Nancy, France, pp. 354–363.

Lenz, R. & Reichert, M. (2007), 'IT support for healthcare processes – premises, challenges, perspectives', *Data and Knowledge Engineering* **61**, 39–58.

Leong, T.-Y., Kaiser, K. & Miksch, S. (2007), 'Free and Open Source Enabling Technologies for Patient-Centric, Guideline-Based Clinical Decision Support: A Survey', *IMIA Yearbook of Medical Informatics, Methods Inf Med* **46**, 74–86.

Lhotska, L. & Prieto, L. (2005), A Multi-agent System for Information Retrieval, *in* R. Moreno Díaz, F. Pichler & A. Quesada Arencibia, eds, 'In Proc. 11th International Conference on Computer Aided Systems Theory, EUROCAST 2007', Vol. 4739 of *Lecture Notes in Computer Science*, Las Palmas de Gran Canaria, Spain, pp. 337–344.

Lhotska, L. & Stepankova, O. (2004), 'Agent architecture for smart adaptive systems ', *Transactions of the Institute of Measurement and Control* **26**(3), 245–260.

Lhotska, L. & Stepankova, O. (2005), Agent Architecture for Diagnostics and Monitoring in Medicine, *in* A. Moreno, ed., 'In Proc. of IJCAI05 Workshop on Agents Applied in Health Care', Edimburgh, Scotland, pp. 42–50.

Lin, C.-E., Kavi, K. M., Sheldon, F. T., Daley, K. M. & Abercrombie, R. K. (2007), A Methodology to Evaluate Agent Oriented Software Engineering Techniques, *in* '40th Annual Hawaii International Conference on System Sciences, HICSS 2007', IEEE Computer Society, Los Alamitos, CA, USA, p. 60a.

Linkov, I., Satterstrom, F., Kiker, G., Batchelor, C., Bridges, T. & Ferguson, E. (2006), 'From comparative risk assessment to multi-criteria decision analysis and adaptive management: Recent developments and applications ', *Environment International* **32**(8), 1072–93.

Lluch-Ariet, M., Estanyol, F., Mier, M., Delgado, C., González-Vélez, H., Dalmas, T., Robles, M., Sáez, C., Vicente, J., van Huffel, S., Luts, J., Arús, C., Candiota Silveira, A., Julià-Sapé, M., Peet, A., Gibb, A., Sun, Y., Celda, B., Martínez Bisbal, M., Valsecchi, G., Dupplaw, D., Hu, B. & Lewis, P. (2008), HealthAgents: Agent-Based Distributed Decision Support System for Brain Tumour Diagnosis and Prognosis, *in* R. Annicchiarico, U. Cortés & C. Urdiales, eds, 'Agent Technology and e-Health', Whitestein Series in Software Agent Technologies and Autonomic Computing, Birkhäuser Verlag, Basel, Switzerland, pp. 5–24.

López, J., Bustos, F. & Julián, V. (2007), 'Tourism Services Using Agent Technology: A MultiAgent Approach', *INFOCOMP - Journal of Computer Science* **Special edition**, 51–57.

Luck, M., McBurney, P., Shehory, O. & Willmott, S., eds (2005), *Agent Technology: Computing as Interaction. A Roadmap for Agent Based Computing*, AgentLink.

Mallya, A. U. & Singh, M. P. (2005), Incorporating Commitment Protocols into Tropos, *in* J. P. Müller & F. Zambonelli, eds, '6th International Workshop Agent-Oriented Software Engineering VI, AOSE 2005. Revised and Invited Papers', Vol. 3950 of *Lecture Notes in Computer Science*, Springer Verlag / Heidelberg, pp. 69–80.

Mansour, E., Wu, B., Dube, K. & Li, J. X. (2006), An Event-Driven Approach to Computerizing Clinical Guidelines Using XML, *in* 'Proceedings of the IEEE Services Computing Workshops, SCW 2006', IEEE Computer Society, Chicago, Illinois, USA, pp. 13–20.

Maviglia, S. M., Zielstorff, R. D., Paterno, M., Teich, J. M., Bates, D. W. & Kuperman, G. J. (2003), 'Automating complex guidelines for chronic disease: Lessons learned', *Journal of the American Medical Informatics Association* **10**(2), 154–166.

McClure, R., Campbell, J., Nyman, M. & Glasgow, J. (2006), Guidelines and Standard Terminology: Making Standard Terminology Work In Clinical Guidelines, *in* 'Proceedings of 78th Annual American Health Information Management Association Conference, AHIMA 2006', AHIMA Press, Denver, USA, pp. 481–491.

McGuinness, D. & Harmelen, F. v. (2004), 'OWL Web Ontology Language'. URL: http://www.w3.org/TR/owl-features/ [last access: 26/04/08].

Mersmann, S. & Dojat, M. (2004), SmartCareTM - Automated Clinical Guidelines in Critical Care, *in* R. L. d. MÃ ntaras & L. Saitta, eds, '16th European Conference on Artificial Intelligence, ECAI 2004', Vol. 110 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, Valencia, Spain, pp. 745–749.

MHP (2006), A New Era for Patient-Centred Health Care, Technical report, Ministry of Health Planning. Government of British Columbia. Available on-line at http://www.health.gov.bc.ca/socsec/publications.html [last visit 23/04/2008].

Miksch, S., Shahar, Y. & Johnson, P. (1997), Asbru: A Task-Specific, Intention-Based, and Time-Oriented Language for Representing Skeletal Plans, *in* E. Motta, F. V. Harmelen, C. Pierret-Golbreich, I. Filby & N. Wijngaards, eds, '7th Workshop on Knowledge Engineering: Methods and Languages, KEML 1997', Milton Keynes, UK.

Moraïtis, P., Petraki, E. & Spanoudakis, N. I. (2002), Engineering JADE Agents with the Gaia Methodology, *in* R. Kowalczyk, J. P. Müller, H. Tianfield & R. Unland, eds, 'Agent Technologies, Infrastructures, Tools, and Applications for E-Services. Agent-Related Workshops, NODe 2002', Vol. 2592 of *Lecture Notes in Artificial Intelligence*, Springer Verlag, pp. 77–91.

Moraïtis, P. & Spanoudakis, N. I. (2004), Combining Gaia and JADE for Multi-agent Systems development, *in* R. Trappl, ed., '17th European Meeting on Cybernetics and Systems Research, EMCSR 2004', Austrian Society for Cybernetic Studies.

Moreno, A., Isern, D. & Sánchez, D. (2003), 'Provision of Agent-Based Health Care Services', *AI Communications* **16**(3), 167–178.

Moreno, A., Sánchez, D. & Isern, D. (2003), Security Measures in a Medical Multi-Agent System, *in* I. Aguiló, L. Valverde & M. T. Escrig, eds, 'Artificial Intelligence Research and Development', Vol. 100 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, pp. 244–255.

Moreno, A., Valls, A. & Bocio, J. (2001), 'A multi-agent system to schedule organ transplant operations', *Revista Iberoamericana de Inteligencia Artificial* **13**(3), 36–44.

Moreno, A., Valls, A., Isern, D. & Sánchez, D. (2003), GruSMA1: Experience on the Deployment of Agent-Based Health Care Services, *in* A. Moreno & J. Nealon, eds, 'Applications of Software Agent Technology in the Health Care Domain', Whitestein Series in Software Agent Technologies, Birkhäuser Verlag, Basel, Switzerland, pp. 49–66.

Moreno, A., Valls, A., Isern, D. & Sánchez, D. (2006), 'The GruSMA Experience', *IEEE Intelligent Systems* **21**(6), 63–67.

Moreno, A., Valls, A. & Riaño, D. (2004), Improving palliative care with agent technology, *in* J. Nealon, A. Moreno, J. Fox & U. Cortés, eds, 'In Proc. of 2nd Workshop on Agents Applied in Health Care in the 16th European conference on Artificial Intelligence, ECAI 2004', Valencia, Spain, pp. 1–7.

Muller, M. L., Ganslandt, T., Eich, H. P., Lang, K., Ohmann, C. & Prokosch, H.-U. (2001), 'Towards integration of clinical decision support in commercial hospital information systems using distributed, reusable software and knowledge components', *International Journal of Medical Informatics* **64**(2-3), 369–377.

NCQHC (2006), *CEO Survival Guide. Electronic Health Record Systems*, National Committee for Quality Health Care, Washington, DC (USA). Available at http://www.nqfexecutiveinstitute.org/executiveinstitute/EHRbookfinal.pdf [last visit 01.30.2008].

Nealon, J. L. & Moreno, A. (2003), Agent-Based Applications in Health Care, *in* J. L. Nealon & A. Moreno, eds, 'Applications of Software Agent Technology in the Health Care Domain', Whitestein Series in Software Agent Technologies, Birkhäuser Verlag, Basel, Switzerland, pp. 3–18.

Neches, R., Richard, F., Finin, T., Gruber, T., Patil, R., Senator, T. & Swartout, W. (1991), 'Enabling Technology for Knowledge Sharing', *AI Magazine* **12**(3), 36–56.

NGC (2007), 'National guideline clearinghouse'. URL: http://www.guideline.gov [last visit 19/11/2007].

Nielsen, T. & Jensen, F. (2004), 'Learning a decision maker's utility function from (possibly) inconsistent behaviour', *Artificial Intelligence* **160**, 53–78.

Noy, N. F. & McGuinness, D. L. (2001), Ontology Development 101: A Guide to Creating Your First Ontology, Technical report, Stanford Medical Informatics, Palo Alto. Available at http://protege.stanford.edu/publications/ontology_development/ontology101.pdf [Last visit 26/04/2008].

Odell, J. (2002), 'Objects and Agents Compared', *Journal of Object Technology* **1**, 41–53.

Odell, J., Dyke, H. V. & Bauer, B. (2000), Extending UML for Agents, *in* 'Agent-Oriented Information Systems Workshop at the 17th National conference on Artificial Intelligence, AOIS 2000', pp. 3–17.

Odell, J., Parunak, H. V. D. & Bauer, B. (2001), Representing Agent Interaction Protocols in UML, *in* 'Agent-Oriented Software Engineering: First International Workshop, AOSE 2000. Revised papers', Vol. 1957 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 201–218.

O'Malley, S. A. & DeLoach, S. (2001), Determining When to Use an Agent-Oriented Software Engineering Paradigm, *in* 'Revised Papers and Invited Contributions from the Second International Workshop on Agent-Oriented Software Engineering, AOSE 2001', Vol. 2222 of *Lecture Notes in Computer Science*, Springer-Verlag UK, pp. 188 – 205.

OMG (2007), 'Unified Modeling Language', Resource page. Available at http://www.uml.org [last visit 2007/11/11].

Padgham, L. & Winikoff, M. (2002), Prometheus: A Methodology for Developing Intelligent Agents, *in* 'First International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS 2002', ACM Press, pp. 37–38.

Padgham, L. & Winikoff, M. (2004), *Developing Intelligent Agent Systems: A Practical Guide*, Wiley Series in Agent Technology, John Wiley and Sons.

Padgham, L. & Winikoff, M. (2005), *Agent-Oriented Methodologies*, Idea Group, chapter Prometheus: A Practical Agent-Oriented Methodology, pp. 107–135.

Palau, J., Montaner, M. & López, B. (2004), Collaboration Analysis in Recommender Systems using Social Networks, *in* 'In Proc. of Eighth International Workshop on Cooperative Information Agents, CIA 2004', Springer-Verlag, Erfurt (Germany), pp. 137–151.

Pavón, J. & Gómez-Sanz, J. (2003), Agent Oriented Software Engineering with INGENIAS, *in* V. Marík, J. Müller & M. Pechoucek, eds, 'Multi-Agent Systems and Applications III. Proceedings of 3rd International Central and Eastern European Conference on Multi-Agent Systems, CEEMAS 2003', Vol. 2691 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 394–403.

Pavón, J., Gómez-Sanz, J. J. & Fuentes, R. (2005), *Agent-Oriented Methodologies*, Idea Group, chapter The INGENIAS Methodology and Tools, pp. 236–276.

Pavón, J., Gómez-Sanz, J. J. & Fuentes, R. (2006), Model Driven Development of Multi-Agent Systems, *in* A. Rensink & J. Warmer, eds, 'Model Driven Architecture, Foundations and Applications. Second European Conference, ECMDA FA 2006', Vol. 4066 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 284–298.

Peleg, M., Boxwala, A. A., Tu, S., Zeng, Q., Ogunyemi, O., Wang, D., Patel, V. L., Greenes, R. A. & Shortliffe, E. H. (2004), 'The InterMed approach to sharable computer-interpretable guidelines: motivations and lessons', *Journal of the American Medical Informatics Association* **11**, 1–10.

Peleg, M., Keren, S. & Denekamp, Y. (2008), 'Mapping computerized clinical guidelines to electronic medical records: Knowledge-data ontological mapper (KDOM)', *Journal of Biomedical Informatics* **41**(1), 180–201.

Peleg, M., Tu, S., Bury, J., Ciccarese, P., Fox, J., Greenes, R. A., Hall, R., Johnson, P. D., Jones, N., Kumar, A., Miksch, S., Quaglini, S., Seyfang, A., Shortliffe, E. H. & Stefanelli, M. (2003), 'Comparing Computer-Interpretable Guideline Models: A Case-Study Approach', *Journal of the American Medical Informatics Association* **10**, 52–68.

Peleg, M. & Tu, S. W. (2006), Decision support, knowledge representation and management in medicine, *in* R. Haux & C. Kulikowski, eds, 'International Medical Informatics Association (IMIA) Yearbook of Medical Informatics 2006', International Medical Informatics Association, Schattauer GmbH, Germany.

Pisanelli, D., ed. (2004), *Ontologies in Medicine*, Vol. 102 of *Studies in Health Technology and Informatics*, IOS Press.

Pisanelli, D., Zaccagnini, D., Capurso, L. & Koch, M. (2004), *The New Navigators, from Professionals to Patients*, Vol. 95 of *Studies in Health Technology and Informatics*, IOS Press, chapter An ontological approach to evidence-based medicine and meta-analysis, pp. 543–548.

Pohjonen, R. (2005), Metamodeling Made Easy - MetaEdit+ (Tool Demonstration), *in* R. Gluck & M. Lowry, eds, 'Generative Programming and Component Engineering. Proc. of 4th International Conference, GPCE 2005', Vol. 3676 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, Tallinn, Estonia, pp. 442–446.

Priori, S. G., Klein, W. & Bassand, J. (2003), 'Medical Practice Guidelines: Separating science from economics', *European Heart Journal* **24**(21), 1962–1964.

Quaglini, S. & Ciccarese, P. (2006), 'Models for guideline representation', *Neurological Sciences* **S3**, 240–244.

Quaglini, S., Ciccarese, P., Micieli, G. & Cavallini, A. (2004), Non-Compliance with Guidelines: Motivations and Consequences in a case study, *in* K. Kaiser, S. Miksch & S. Tu, eds, 'Symposium on Computerized Guidelines and Protocols, CGP 2004', Vol. 101 of *Studies in Health Technology and Informatics*, IOS Press, Prague, Czech Republic, pp. 75–87.

Quaglini, S., Stefanelli, M., Cavallini, A., Micieli, G., Fassino, C. & Mossa, C. (2000), 'Guideline-based Careflow Systems', *Artificial Intelligence in Medicine* **20**, 5–22.

Quaglini, S., Stefanelli, M., Lanzola, G., Caporusso, V. & Panzarasa, S. (2001), 'Flexible guideline-based patient careflow systems', *Artificial Intelligence in Medicine* **22**, 65–80.

Raghupathi, W. & Tan, J. (2002), 'Strategic IT applications in health care', *Communications of the ACM* **45**(12), 56–61.

Ram, P., Berg, D., Tu, S. W., Mansfield, J. G., Ye, Q. & Abarbanel, R. M. (2004), Executing Clinical Practice Guidelines using the SAGE Execution Engine, *in* M. Fieschi, E. Coiera & Y.-C. J. Li, eds, 'Congress of the International Medical Informatics Association, MEDINFO 2004', Vol. 107 of *Studies in Health Technology and Informatics*, IOS Press, San Francisco, USA, pp. 251–5.

Real, F. & Riaño, D. (2008), Automatic Combination of Formal Intervention Plans Using SDA* Representation Model, *in* D. Riaño, ed., 'Knowledge Management for Health Care Procedures. From Knowledge to Global Care AIME 2007 Workshop K4Care 2007', Vol. 4924 of *Lecture Notes in Artificial Intelligence*, pp. 75–86.

Riaño, D. (2004), Ordered Time-Independent CIG Learning, *in* J. M. Barreiro, F. Martín-Sánchez, V. Maojo & F. Sanz, eds, 'V International Symposium on Biological and Medical Data Analysis, ISBMDA 2004', Vol. 3337 of *Lecture Notes in Computer Science*, Springer Verlag, Barcelona, Catalonia, pp. 117–128.

Riaño, D. (2007), The sda* model: A set theory approach, *in* 'Twentieth IEEE International Symposium on Computer-Based Medical Systems, CBMS 2007', IEEE Press, Maribor, Slovenia, pp. 563–568.

Riaño, D., López-Vallverdú, J. & Tu, S. (2008), Mining Hospital Data to Learn SDA* Clinical Algorithms, *in* D. Riaño, ed., 'Knowledge Management for Health Care Procedures. From Knowledge to Global Care AIME 2007 Workshop K4Care 2007', Vol. 4924 of *Lecture Notes in Artificial Intelligence*, pp. 46–61.

Riaño, D., Moreno, A. & Valls, A. (2004), PalliaSys: Agent-Based Palliative Care, *in* '4th International Conference on Intelligent Systems Design and Applications, ISDA 2004', IEEE Press, Budapest, Hungary, pp. 121–126.

Ricci, S., Celani, M. G. & Righetti, E. (2006), 'Development of clinical guidelines: methodological and practical issues', *Neurological Sciences* **27**, 228–230.

Richard, N., Dojat, M. & Garbay, C. (2004), 'Automated segmentation of human brain MR images using a multi-agent approach', *AI in Medicine* **30**, 153–176.

Ricordel, P.-M. & Demazeau, Y. (2000), From Analysis to Deployment: A Multi-agent Platform Survey, *in* A. Omicini, R. Tolksdorf & F. Zambonelli, eds, 'First International Workshop Engineering Societies in the Agents World, ESAW 2000', Vol. 1972, Springer Berlin / Heidelberg, Berlin, Germany, pp. 93–105.

Roy, B. (1991), 'The outranking approach and the foundations of ELECTRE methods', *Theory and Decision* **31**, 49–73.

Roy, B. (2005), *Paradigms and Challenges*, Vol. 78 of *International Series in Operations Research and Management ScienceLecture Notes in Economics and Mathematical Systems*, Springer New York, pp. 3–24.

Rutten, F., Brouwer, W. & Niessen, L. (2005), 'Practice guidelines based on clinical and economic evidence. Indispensable tools in future market oriented health care', *European Journal of Health Economics* **6**, 91–93.

Sabas, A., Delisle, S. & Badri, M. (2002), A Comparative Analysis of Multiagent System Development Methodologies: Towards a Unified Approach, *in* 'Third International Symposium From Agent Theory to Agent Implementation, AT2AI-3, Sixteenth European Meeting on Cybernetics and Systems Research', Vol. 2, pp. 599–604.

Schadow, G., Russler, D. C. & McDonald, C. J. (2001), 'Conceptual alignment of electronic health record data with guideline and workflow knowledge', *International Journal of Medical Informatics* **64**(2-3), 259–274.

Schreiber, G., Akkermans, H., Anjewierden, A., de Hoog, R., Shadbolt, N., de Velde, W. V. & Wielinga, B. (2000), *Knowledge Engineering and Management: The CommonKADS Methodology*, MIT Press.

Schreiber, G., Akkermans, H., Anjewierden, A., Hoog, R. d., Shadbolt, N., Velde, W. V. d. & Wielinga, B. (1999), *Knowledge Engineering and Management: The CommonKADS Methodology*, MIT Press.

Serban, R., ten Teije, A., van Harmelen, F., Marcos, M. & Polo-Conde, C. (2007), 'Extraction and use of linguistic patterns for modelling medical guidelines', *Artificial Intelligence in Medicine* **39**, 137–149.

Seyfang, A., Miksch, S., Marcos, M., Wittenberg, J., Polo-Conde, C. & Rosenbrand, K. (2006), Bridging the Gap between Informal and Formal Guideline Representations, *in* G. Brewka, S. Coradeschi, A. Perini & P. Traverso, eds, '17th European Conference on Artificial Intelligence', IOS Press, Riva del Garda, Italy, pp. 447–451.

Shahar, Y., Young, O., Shalom, E., Galperin, M., Mayaffit, A., Moskovitch, R. & Hessing, A. (2004), 'A Framework for a Distributed, Hybrid, Multiple-Ontology Clinical-Guideline Library and Automated Guideline-Support Tools', *Journal of Biomedical Informatics* **37**(5), 325–344.

Shahar, Y., Young, O., Shalon, E., Mayaffit, A., Moskovitch, R., Hessing, A. & Galperin, M. (2003), DEGEL: A Hybrid, Multiple-Ontology Framework for Specification and Retrieval of Clinical Guidelines, *in* M. Dojat, E. Keravnou & P. Barahona, eds, 'Proceedings of the 9th Conference on Artificial Intelligence in Medicine in Europe, AIME 2003', Vol. 2780 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, Protaras, Cyprus, pp. 122–131.

Shankar, R., Tu, S. W. & Musen, M. A. (2002), Use of Protégé-2000 to Encode Clinical Guidelines, *in* 'Proceedings of American Medical Informatics Association Annual Symposium, AMIA 2002', American Medical Informatics Association, San Antonio, USA, p. 1164.

Shehory, O. & Sturm, A. (2001), Evaluation of modeling techniques for agent-based systems, *in* 'Proceedings of the fifth international conference on Autonomous agents, AGENTS 2001', ACM Press, pp. 624 – 631.

Shiffman, R. N., Liaw, Y., Brandt, C. A. & Corb, G. J. (1999), 'Computer-based Guideline Implementation Systems: A Systematic Review of Functionality and Effectiveness', *Journal of the American Medical Informatics Association* **6**(2), 104–114.

Shiffman, R. N., Michel, G., Essaihi, A. & Thornquist, E. (2004), 'Bridging the Guideline Implementation Gap: A Systematic, Document-Centered Approach to Guideline Implementation', *Journal of the American Medical Informatics Association* **11**, 418–427.

Sichman, J. S., Dignum, V. & Castelfranchi, C. (2005), 'Agents Organizations: A Concise Overview', *Brazilian Computer Society, Special Issue on Agents Organizations* **11**(1), 1–8.

Simone, F.-H. (2001), *IT-Security and Privacy. Design and Use of Privacy Enhancing Security Mechanisms*, Vol. 1958 of *Lecture Notes In Computer Science*, Springer Berlin / Heidelberg.

Singer, S. J., Enthoven, A. C. & Garber, A. M. (2001), *Medical Informatics. Computer Applications in Health Care and Biomedicine*, Springer Verlag, chapter Health Care and Information Technology: Growing up Together, pp. 663–696.

Singh, S., Ikhwan, B., Haron, F. & Yong, C. (2005), Architecture of Agent-Based Healthcare Intelligent Assistant on Grid Environment, *in* 'In Proc. of Parallel and Distributed Computing: Applications and Technologies conference, PDCAT 2004', Vol. 3320 of *Lecture Notes in Computer Science*, Singapore, pp. 58–61.

Sánchez, D., Isern, D. & Moreno, A. (2006), Integrated Agent-Based Approach for Ontology-Driven Web Filtering, *in* B. Gabrys, R. J. Howlett & L. C. Jain, eds, 'Knowledge-Based Intelligent Information and Engineering Systems, KES 2006', Vol. 4253 of *Lecture Notes in Artificial Intelligence*, Springer Verlag, Bournemouth, UK, pp. 758–765.

Soto, J., Vizcaíno, A., Portillo-Rodriguez, J. & Piattini, M. (2006), Modelling a Knowledge Management System Architecture with INGENIAS Methodology, *in* 'Proceedings of 15th International Conference on Computing, CIC 2006', IEEE Computer Society, pp. 167–173.

Stefanelli, M. (2004), 'Knowledge and process management in health care organizations', *Methods of Information in Medicine* **43**(5), 525–535.

Sturm, A. & Shehory, O. (2004*a*), A Framework for Evaluating Agent-Oriented Methodologies, *in* P. Giorgini, B. Henderson-Sellers & M. Winikoff, eds, 'Agent Oriented Information Systems. 5th International Bi-Conference Workshop, AOIS 2003. Revised Selected Papers.', Vol. 3030 of *Lecture Notes in Artificial Intelligence*, Springer Berlin / Heidelberg, pp. 94–109.

Sturm, A. & Shehory, O. (2004*b*), *Methodologies and Software Engineering for Agent Systems*, Vol. 11 of *Multiagent Systems, Artificial Societies, and Simulated Organizations*, Springer US, chapter A Comparative Evaluation of Agent-Oriented Methodologies, pp. 127–149.

Sudeikat, J., Braubach, L., Pokahr, A. & Lamersdorf, W. (2004), Evaluation of Agent - Oriented Software Methodologies - Examination of the Gap Between Modeling and Platform, *in* P. Giorgini, J. P. Müller & J. Odell, eds, 'Agent-Oriented Software Engineering V, Fifth International Workshop AOSE 2004', Vol. 3382 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 126–141.

Supekar, K., Rubin, D., Noy, N. & Musen, M. (2007), Knowledge Zone: A Public Repository of Peer-Reviewed Biomedical Ontologies, *in* K. A. Kuhn, J. R. Warren & T.-Y. Leong, eds, 'Proceedings of the 12th World Congress on Health (Medical) Informatics, MEDINFO 2007', Vol. 129 of *Studies in Health Technology and Informatics*, IOS Press, Brisbane, Australia, pp. 812–816.

Sutton, D. R. & Fox, J. (2003), 'The syntax and semantics of the PROforma guideline modeling language', *Journal of the American Medical Informatics Association* **10**, 433–443.

Tablado, A., Illarramendi, A., Bagües, M. I., Bermúdez, J. & Goñi, A. (2004), Agents in a system for monitoring elderly people, *in* J. Nealon, A. Moreno, J. Fox & U. Cortés, eds, 'In Proc. of 2nd Workshop on Agents Applied in Health Care in the 16th European conference on Artificial Intelligence, ECAI 2004', Valencia, Spain, pp. 47–53.

ten Teije, A., Marcos, M., Balser, M., van Croonenborg, J., Duelli, C., van Harmelen, F., Lucas, P., Miksch, S., Reif, W., Rosenbrand, K. & Seyfang, A. (2006), 'Improving medical protocols by formal methods', *Artificial Intelligence in Medicine* **36**, 193–272.

Terenziani, P., Carlini, C. & Montani, S. (2002), Towards a comprehensive treatment of temporal constraints in clinical guidelines, *in* 'Ninth International Symposium on Temporal Representation and Reasoning, TIME 2002', IEEE Press, Manchester, UK, pp. 20–27.

Terenziani, P., Molino, G. & Torchio, M. (2001), 'A modular approach for representing and executing clinical guidelines', *Artificial Intelligence in Medicine* **23**, 249–276.

Terenziani, P., Montani, S., Bottrighi, A., Molino, G. & Torchio, M. (2005), Clinical Guidelines Adaptation: Managing Authoring and Versioning Issues, *in* S. Miksch, J. Hunter & E. Keravnou, eds, '10th Conference on Artificial Intelligence in Medicine, AIME 2005', Vol. 3581 of *Lecture Notes on Artificial Intelligence*, Springer Verlag, Aberdeen, Scotland, pp. 151–155.

Terenziani, P., Montani, S., Bottrighi, A., Torchio, M., Molino, G., Anselma, L. & Correndo, G. (2003), Applying Artificial Intelligence to Clinical Guidelines: The GLARE Approach, *in* A. Cappelli & F. Turini, eds, 'AI*IA 2003: Advances in Artificial Intelligence, 8th Congress of the Italian Association for Artificial Intelligence', Vol. 2829 of *Lecture Notes in Computer Science*, Springer-Verlag Heidelberg, pp. 536–547.

Terenziani, P., Montani, S., Bottrighi, A., Torchio, M., Molino, G. & Correndo, G. (2004), The GLARE Approach to Clinical Guidelines: Main Features, *in* K. Kaiser, S. Miksch & S. Tu, eds, 'Computer-based Support for Clinical Guidelines and Protocols. Proceedings of the Symposium on Computerized Guidelines and Protocols, CGP 2004', Vol. 101 of *Studies in Health Technology and Informatics*, IOS Press, Prague, Czeck Republic, pp. 162–166.

Thangarajah, J., Padgham, L. & Winikoff, M. (2005), Prometheus Design Tool, *in* 'Proceedings of the fourth International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS 2005', ACM Press, pp. 127–128.

TILab S.p.A. (2002), 'Java Agent Development Framework (JADE)'. More information available on http://sharon.cselt.it/projects/jade.

Tolchinsky, P., Cortés, U. & Grecu, D. (2008), Argumentation-Based Agents to Increase Human Organ Availability for Transplant, *in* R. Annicchiarico, U. Cortés & C. Urdiales, eds, 'Agent Technology and e-Health', Whitestein Series in Software Agent Technologies and Autonomic Computing, Birkhäuser Verlag, Basel, Switzerland, pp. 65–94.

Torra, V. (1997), 'The Weighted OWA Operator', *International Journal of Intelligent Systems* **12**, 153–166.

Torra, V. & Narukawa, Y. (2007), *Modeling Decisions: Information Fusion and Aggregation Operators*, Springer Berlin Heidelberg.

Tran, Q.-N., Low, G. & Williams, M.-A. (2003), A Feature Analysis Framework for Evaluating Multi-agent System Development Methodologies, *in* N. Zhong, ed., 'Foundations of Intelligent Systems. Proceedings of the 14th Int. Symposium on Methodologies for Intelligent Systems, ISMIS 2003', Vol. 2871 of *Lecture Notes in Artificial Intelligence*, Springer Verlag, pp. 613–17.

Tran, Q.-N. N., Low, G. & Williams, M.-A. (2005), A preliminary comparative feature analysis of multi-agent systems development methodologies, *in* P. Bresciani, P. Giorgini, B. Henderson-Sellers, G. Low & M. Winikoff, eds, 'Agent-Oriented Information Systems II. 6th International Bi-Conference Workshop, AOIS 2004. Revised Selected Papers', Vol. 3508 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 386–398.

Tu, S. & Musen, M. (2001), Modeling Data and Knowledge in the EON Guideline Architecture, *in* V. Patel, R. Rogers & R. Haux, eds, 'Proceedings of 10th Triennial Congress of the International Medical Informatics Association, MEDINFO 2001', Vol. 84 of *Studies in Health Technology and Informatics*, IOS Press, London, UK, pp. 280–284.

Tu, S. W., Campbell, J. & Musen, M. A. (2004), SAGE Guideline Modeling: Motivation and Methodology, *in* K. Kaiser, S. Miksch & S. Tu, eds, 'Symposium on Computerized Guidelines and Protocols, CGP 2004', Vol. 101 of *Studies in Health Technology and Informatics*, IOS Press, Prague, Czech Republic, pp. 167–171.

Tu, S. W., Campbell, J. R., Glasgow, J., Nyman, M. A., McClure, R., McClay, J., Parker, C., Hrabak, K. M., Berg, D., Weida, T., Mansfield, J. G., Musen, M. A. & Abarbanel, R. M. (2007), 'The SAGE Guideline Model: Achievements and Overview', *Journal of the American Medical Informatics Association* **14**(5), 589–598.

Tu, S. W., Hrabak, K. M., Campbell, J. R., Glasgow, J., Nyman, M. A., McClure, R., James, M., Abarbanel, R., Mansfield, J. G., Martins, S. M., Goldstein, M. K. & Musen, M. A. (2006), Use of Declarative Statements in Creating and Maintaining Computer-Interpretable Knowledge Bases for Guideline-Based Care, *in* 'Proceedings of American

Medical Informatics Association Annual Symposium, AMIA 2006', American Medical Informatics Association, Washington DC, USA, pp. 784–788.

Veselý, A., Zvárová, J., Peleska, J., Buchtela, D. & Anger, Z. (2006), 'Medical guidelines presentation and comparing with Electronic Health Record', *International Journal of Medical Informatics* **75**(3-4), 240–245.

Vázquez-Salceda, J., Cortés, U., Padget, J., López-Navidad, A. & Caballero, F. (2003), 'The Organ Allocation process: a natural extension of the CARREL Agent-Mediated Electronic Institution', *AI Communications* **16**(3), 71–82.

Vázquez-Salceda, J. & Dignum, V. (2003), Modelling Electronic Organizations, *in* J. G. Carbonell & J. Siekmann, eds, 'Multi-Agent Systems and Applications III: 3rd International Central and Eastern European Conference on Multi-Agent Systems, CEEMAS 2003. Proceedings', Vol. 2691 of *Lecture Notes in Artificial Intelligence*, pp. 1070–80.

Vázquez-Salceda, J., Álvarez, S., Kifor, T., Varga, L., Miles, S., Moreau, L. & Willmott, S. (2008), EU PROVENANCE Project: An Open Provenance Architecture for Distributed Applications, *in* R. Annicchiarico, U. Cortés & C. Urdiales, eds, 'Agent Technology and e-Health', Whitestein Series in Software Agent Technologies and Autonomic Computing, Birkhäuser Verlag, Basel, Switzerland, pp. 55–64.

W3C (2007), 'Extensible Markup Language', Resource page. Available at http://www.w3.org/XML/ [last visit 2007/11/11].

Wang, D. (2003), A generic execution model for sharing of computer-interpretable clinical practice guidelines, PhD thesis, Columbia University, USA.

Wang, D. A., Peleg, M., Tu, S., Shortliffe, E. H. & Greenes, R. A. (2001), Representation of Clinical Practice Guidelines, *in* V. Patel, R. Rogers & R. Haux, eds, 'Proceedings of 10th Triennial Congress of the International Medical Informatics Association, MEDINFO 2001', Vol. 84 of *Studies in Health Technology and Informatics*, IOS Press, London, UK.

Wang, D., Peleg, M., Bu, D., Cantor, M., Landesberg, G., Lunenfeld, E., Tu, S., Kaiser, G., Hripcsak, G., Patel, V. & Shortliffe, E. (2003), GESDOR - A Generic Execution Model for Sharing of Computer-Interpretable Clinical Practice Guidelines, *in* M. Musen, ed., 'Proceedings of American Medical Informatics Association Annual Symposium, AMIA 2003', American Medical Informatics Association, Ottawa, Ontario, CA, pp. 694–698.

Wang, D., Peleg, M., Tu, S. W., Boxwala, A. A., Ogunyemi, O., Zeng, Q., Greenes, R. A., Patel, V. L. & Shortliffe, E. H. (2004), 'Design and Implementation of the GLIF3 Guideline Execution Engine', *Journal of Biomedical Informatics* **37**, 305–318.

Wang, D. & Shortliffe, E. H. (2002), GLEE - A Model-Driven Execution System for Computer-Based Implementation of Clinical Practice Guidelines, *in* 'Proceedings of American Medical Informatics Association Annual Symposium, AMIA 2002', American Medical Informatics Association (AMIA), San Antonio, USA, pp. 855–859.

Weber, J. & Pollack, M. E. (2008), Evaluating User Preferences for Adaptive Reminding, *in* G. Gottlob & T. Walsh, eds, 'In Proceedings of ACM CHI 2008 Conference on Human Factors in Computing Systems', Morgan Kaufmann, Florence, Italy, pp. 2949–2954.

Wiesman, F., Hasman, A., Braun, L. & van den Herik, J. (2006), 'Information Retrieval in Medicine: The Visual and the Invisible', *IT Information Technology* **48**(1), 24–32.

Willmott, S., Dale, J., Burg, B., Charlton, P. & O'Brien, P. (2001), 'AgentCities: A World Wide Open Agent Network', *AgentLink News* **8**, 13–15. Available on-line at http://www.agentlink.org/newsletter/ [last access 17/05/2008].

Wirfs-Brock, R., Wilkerson, B. & Wiener, L. (1990), *Designing Object-Oriented Software*, Prentice Hall.

Witten, I. H. & Frank, E. (2005), *Data Mining: Practical machine learning tools and techniques*, 2nd edition edn, Morgan Kaufmann, San Francisco.

Wood, M. F. & DeLoach, S. (2000), An Overview of the Multiagent Systems Engineering Methodology, *in* 'Agent-Oriented Software Engineering: First International Workshop, AOSE 2000. Revised papers', Vol. 1957 of *Lecture Notes in Computer Science*, pp. 207–222.

Wooldridge, M. (2002), *An Introduction to Multiagent Systems*, John Wiley and Sons, Ltd, West Sussex, England.

Wooldridge, M. & Ciancarini, P. (2000), Agent-Oriented Software Engineering: The State of the Art, *in* 'Agent-Oriented Software Engineering: First International Workshop, AOSE 2000. Revised papers', Vol. 1957 of *Lecture Notes in Computer Science*, Springer, pp. 55–82.

Wooldridge, M. & Jennings, N. R. (1995), 'Intelligents Agents: Theory and Practice', *Knowledge Engineering Review* **10**(2), 115–152.

Woolf, S. H., Grol, R., Hutchinson, A., Eccles, M. & Grimshaw, J. (1999), 'Potential benefits, limitations, and harms of clinical guidelines', *British Medical Journal* **318**(7297), 527–530.

Xanthos, C. (2007), 'Delivering a patient-focused health service', *Health Sociology Review* **16**, 263–279.

Yager, R. (1988*a*), 'Including Importances in OWA Aggregations Using Fuzzy Systems Modeling', *IEEE Transactions on Fuzzy Systems* **6**, 286–294.

Yager, R. R. (1988*b*), 'On Ordered Weighted Averaging Aggregation Operators in Multicriteria Decision making', *IEEE Trans Syst Man Cybern* **18**, 183–190.

Young, O. & Shahar, Y. (2005), The Spock System: Developing a Runtime Application Engine for Hybrid-Asbru Guidelines, *in* S. Miksch, J. Hunter & E. Keravnou, eds, 'Proceedings of the 10th Conference on Artificial Intelligence in Medicine, AIME 2005', Vol. 3581 of *Lecture Notes in Artificial Intelligence*, Springer Berlin / Heidelberg, Aberdeen, Scotland, pp. 166–170.

Young, O., Shahar, Y., Liel, Y., Lunenfeld, E., Bar, G., Shalom, E., Martins, S. B., Vaszar, L. T., Marom, T. & Goldstein, M. K. (2007), 'Runtime application of Hybrid-Asbru clinical guidelines', *Journal of Biomedical Informatics* **40**, 507–526.

Zachewitz, L. (2004), MEDUSA: A Multiagent System for Establishing Electronic Healthcare Records, *in* J. Nealon, A. Moreno, J. Fox & U. Cortés, eds, 'In Proc. of 2nd Workshop on Agents Applied in Health Care in the 16th European conference on Artificial Intelligence, ECAI 2004', Valencia, Spain, pp. 31–37.

Zambonelli, F., Jennings, N. R., Omicini, A. & Wooldridge, M. (2001), *Coordination of Internet Agents*, Springer, chapter Agent-Oriented Software Engineering for Internet Applications, pp. 326–346.

Zambonelli, F., Jennings, N. R. & Wooldridge, M. (2000), Organisational Abstractions for the Analysis and Design of Multi-Agent Systems, *in* 'Agent-Oriented Software Engineering: First International Workshop, AOSE 2000. Revised papers', Vol. 1957 of *Lecture Notes in Computer Science*, pp. 407–422.

Zambonelli, F., Jennings, N. R. & Wooldridge, M. (2005), *Agent-Oriented Methodologies*, Idea Group, chapter Multi-Agent Systems as Computational Organizations: The Gaia Methodology, pp. 136–171.

Zambonelli, F., Jennings, N. & Wooldridge, M. (2003), 'Developing Multiagent Systems: The Gaia Methodology', *ACM Transactions on Software Engineering and Methodology* **12**(3), 317–370.

Zielstorff, R. D. (1998), 'Online Practice Guidelines. Issues, Obstacles, and Future Prospects ', *Journal of the American Medical Informatics Association* **5**(3), 227–236.

For more information, please visit *http://itaka2-deim.urv.cat/hecase2/*

Last update: *December 2, 2008*

```
@PhDThesis{phdthesis:isern:2008,
   title={{Agent-Based Management of Clinical Guidelines}},
   author={David Isern},
   year={2008},
   school={Technical University of Catalonia},
   address={Barcelona, Catalonia}
}
```