

THE ROLE OF KNOWLEDGE IN DESIGNING AN AGENT PLATFORM FOR HOME CARE*

ÁKOS HAJNAL, GIANFRANCO PEDONE, LÁSZLÓ ZSOLT VARGA

*Computer and Automation Research Institute of the Hungarian Academy of Sciences, Kende u. 13-17
Budapest, H-1111, Hungary*

ANTONIO MORENO, DAVID RIAÑO

*Intelligent Technologies for Advanced Knowledge Acquisition Computer Science and Mathematics Department, University Rovira
i Virgili, Av.Paisos Catalans 26, Tarragona, 43007, Spain*

There is a great demand on the application of Artificial Intelligence techniques and Multi-Agent Systems in health care, since traditional techniques are often not suitable to manage complex tasks that highly change in time or to adapt to unexpected events. The codification of health care treatments as well as the formalization of domain, application knowledge served as an explicit, a priori asset for the agent platform to be implemented. However, the system is required having the capability of applying new, implicit knowledge emerging from physicians on the fly. This paper presents a methodology in modeling and implementing an agent system for home care that is also able to admit and apply new medical knowledge.

1. Introduction

The work presented in this paper is part of the Knowledge for Care (K4Care) European project, whose aim is to provide a *Home Care Model*, and develop a prototype system based on Web technology and intelligent agents. The percentage of old and chronically ill people in European countries is putting a very heavy economic and social pressure on all national health care systems. This problem can be somehow palliated if home care services are improved and used as a valid alternative to hospitalization. In the context of the *K4Care* project, we are targeting a software engineering method that (at least partially) automates the development of an agent platform for this knowledge-intensive application. The intended solution has two basic features. On the one hand, actors are members of well defined organizations. On the other hand, there is extensive domain knowledge to be considered.

Developing software for medical applications, in particular for home care, is special, since the software needs to incorporate complex and sometimes intuitive knowledge present in the mind of the medical staff. Moreover the medical staff is organized into a well-structured model, where roles and responsibilities of each participant are well-defined.

The most relevant aspect of our architecture is the separation of knowledge description from software implementation granting a high level of interoperability and independence among elements of the system. Key elements of the architecture are shown in Figure 1 and will be described in details throughout this paper.

The declarative and procedural knowledge form the *Explicit Knowledge Layer* (detailed in Section 3), where domain entities and actor capabilities can be formally described. Agent-oriented code generation is the core function of the *Implementation Layer*, the deployment of agents and the end-users are illustrated in the *Application Layer* (described in Section 5). The *Implicit Knowledge Layer* embeds the capability of the architecture to capture implicit knowledge (detailed in Section 4). This is the functional point where new medical knowledge is tacitly created and formalized by a proper description mechanism.

The agent paradigm advances the modeling of software systems by embodying a stronger notion of autonomy and control than objects, including the notion of reactive, proactive and social behaviors, as well as assuming inherent multi-threaded control. This allows to handle the complexity by powerful abstractions in engineering software systems. In order to be able to build complex and reliable systems, we need not only new models and technologies, but also an appropriate

* This paper was prepared in the context of the K4CARE project, funded under the 6th Framework Programme of the European Community (Contract N. 026968).

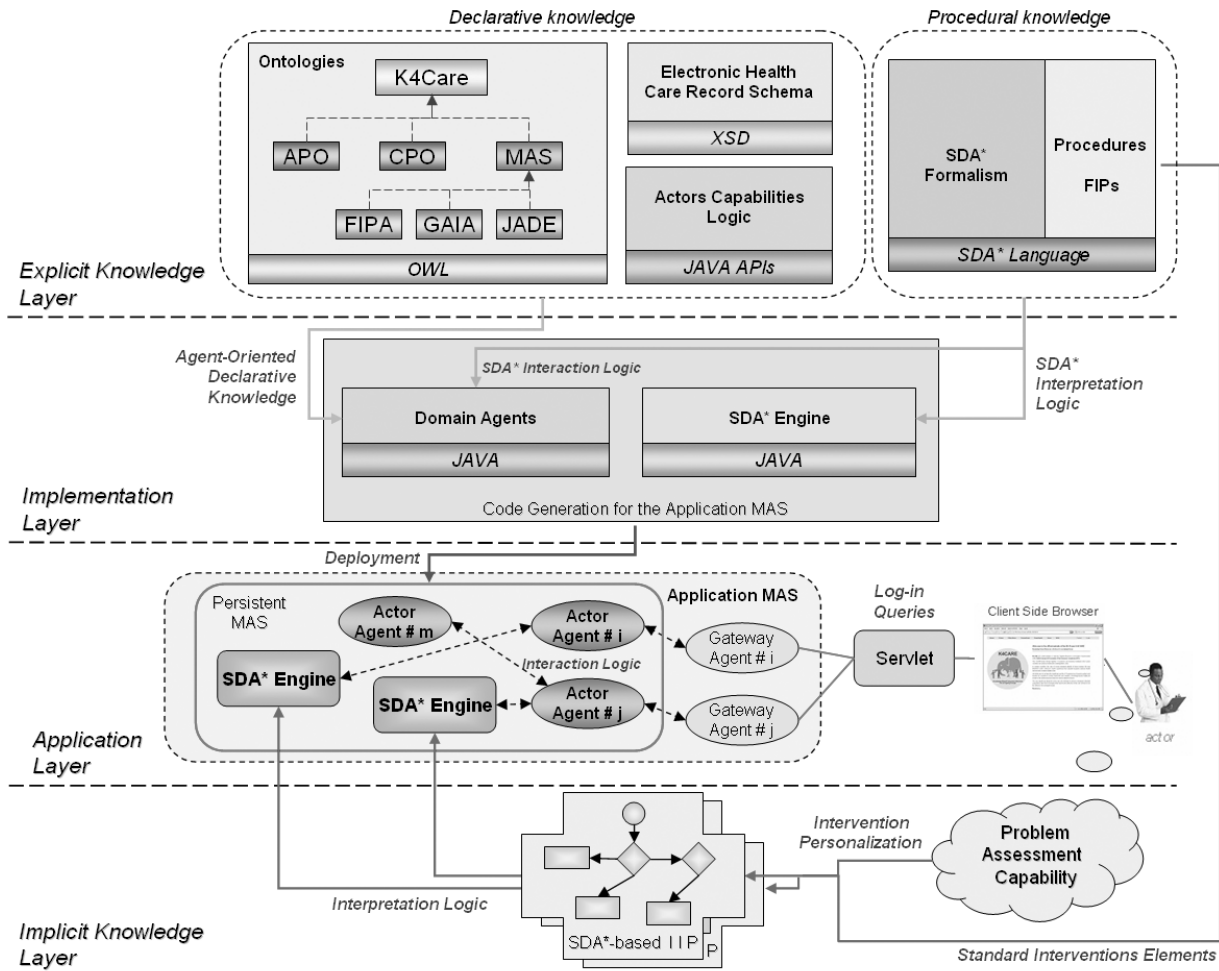


Figure 1. Knowledge-Driven Architecture of the Home Care Platform

set of software engineering abstractions that can be used as a reference for system analysis, and as the basis for methodologies that enable developers to engineer software systems in a robust, reliable, and repeatable fashion.

The result of our investigations is an architecture based and driven by the knowledge, which is able to generate agent code from ontologies and codified treatments. Moreover the architecture enables the creation and embedment of new medical knowledge (referred to as *implicit knowledge*).

The rest of the paper is organized as follows. Section 2 overviews the methodological background for deriving agent-based platform from high-level knowledge descriptions. Section 3 describes the role of the explicit knowledge in the architecture. Section 4 presents the capability of the platform to collect and apply implicit knowledge. Section 5 briefly describes

how the implementation of the platform can be supported by automation. Finally, Section 6 concludes the paper.

2. Methodological Outlines

A well accepted approach to engineer object oriented software systems is the Model Driven Architecture (MDA) ([1]) that provides a new way of writing specifications based on a high-level platform-independent model. The complete MDA specification consists of a base UML model, platform-specific models and interface definition sets, each describing how the base model is implemented on different target platforms.

There are several agent-oriented software engineering methods, however they do not contain the transformations from high level models to platform-specific models in the way as the MDA approach. In

this paper we focus on the importance of the knowledge asset in providing support to home care, and particularly, on its role in different phases of the system development. We are moreover advancing agent oriented software engineering methods by applying an MDA style approach to the engineering of the agent platform of the K4Care project.

There are two main differences from MDA. First, the high level description is completely knowledge based compared to the object oriented model of MDA. Second, the target is a platform concerning the agent paradigm. These advancements are perceived as a considerable support when implementing software for a knowledge intensive application. MDA is based on UML, which is basically a notation language. UML, however, does not capture behavioral aspects (dynamism) of runtime interactions.

In the actual state of the art of agent technology, there is still a lack of well-established Agent-Oriented Software Engineering (AOSE) ([2]) methodologies to support the transformation of system description to deployable agent system. Several AOSE methodologies have been proposed: [3], [4], [5], [6].

A formal AOSE modeling technique is the GAIA methodology ([7]), whose approach focuses on the *human organization* metaphor, which seems to be the most appropriate for highly populated domains ([8], [9]) with a required reliable behavior. Multi-Agent Systems (MASs) are necessary infrastructures that support agent deployment. In this project we use JADE ([10]), which is a FIPA ([11]) compliant middleware for development of agent-based applications.

In this context of methodological uncertainty, an important support can derive from the use of ontologies at modeling-time. In this paper, we develop a complete agent-based system for the home care domain, taking into account the knowledge coming from different ontological sources, such as domain model, project requirements, implementation standards (GAIA methodology, FIPA) and technological issues (JADE), and from medical procedure representations.

Instead of separately using an AOSE methodology to model the MAS and consequently developing the inherent code, we merged the two aspects by conceptualizing all inherent elements into ontologies and interpretable documents, and then generate the deployable agent code. Others in the literature ([12], [13]) have demonstrated that the problem of deriving

code from an ontology is not a trivial task because of different factors: the *higher semantics* of the OWL language in comparison with programming languages (like JAVA); the *complexity of the domain* representation, the relations and the knowledge contained, and the *runtime code dependencies* that must be captured and included in the automation.

We can highlight key methodological aspects of the paper as follows:

1. A general methodology is proposed in modeling an organizational domain, capturing and classifying all the explicit application knowledge, categorized into two main classes: declarative (identifying the system compositional elements) and procedural (capturing the behavioral logic governing the system).
2. The possibility to introduce ontological inference and deduction in validating the architectural model.
3. The system capability to capture the implicit knowledge deriving from physicians. The behavioral skills of agents are orchestrated by the interaction artifacts. Implicit knowledge (as well as the procedural one) is codified through elementary entities (states, decisions, actions), whose interpretation permits to capture knowledge manifested by physicians at runtime.
4. Automation in the generation of the agent-oriented code.

3. Explicit Knowledge in K4care

This section presents the classification of the explicit knowledge that can be formally described in the platform. First, we introduce the declarative knowledge of the Home Care domain (Section 3.1), then we describe how the procedural knowledge can be formalized (Section 3.2).

3.1. Declarative Knowledge

In Artificial Intelligence *ontologies* are a standard knowledge representation mechanism [14]. In K4Care, they are the catalysts for the agent behavioral model as well as for the agent code generation.

We have divided the *application ontologies* into different sources in relation to their application coherence. They have been classified as follows:

- **Domain ontology.** It is divided into Actor Profile Ontology (APO) and patient-Case Profile Ontology (CPO). APO represents the profiles of the subjects and contains the capabilities (skills) of the actors (e.g. physicians, nurses, patients). CPO represents

the relevant medical concepts. Domain ontologies describe “know-what” knowledge about actors and pathologies.

- **FIPA ontology.** It is the general conceptualization of FIPA standards related with MAS development. It represents “know-what” knowledge of the MAS and “know-how” knowledge of interactions.
- **GAIA ontology.** It includes the description of the adopted MAS development methodology.
- **JADE ontology.** It expresses the implementation concepts complying with JADE.
- **K4Care ontology.** This ontology models the ontology-cross references. Agent capabilities, for example, are expressed in terms of “actions” in the APO; these are considered as “responsibilities” or “permissions” by GAIA; the behavioral logic of an agent inherent to its capabilities is expressed in JADE by “behaviors”, which can contain FIPA compliant “interaction protocols”.

Electronic Health Care Record represents schemes of the patient care profile. *Actor Capabilities* are defined by basic actions representing permanent capabilities of agents. These ontologies are described in OWL, further details are omitted here for brevity.

3.2. Procedural Knowledge

Procedural knowledge codifies complex medical tasks. Although a single actor owns a limited set of behavioral capabilities, these can be combined and orchestrated by physicians in order to obtain much more complex behaviors in the system.

Procedural knowledge is applied in two different areas in home care:

- *Procedures*, which are descriptions of services provided by the platform.
- *Formal Intervention Plans* (FIPs), which are general descriptions of the way in which a particular pathology should be treated.

Procedures relate to general services provided by the platform, such as patient admission. Formal Intervention Plans are the typical treatment procedures for specific pathologies. Both kind of descriptions are expressed using a flowchart-based representation called SDA* ([15]). SDA* represents a sort of plan in terms of *states*, *decisions* and *actions* (SDA* elementary entities). We omit details of procedures and focus on intervention planning in the following.

In a FIP model, *states* are used to describe patient conditions, situations, or statuses that deserve a

particular course of actions, totally or partially different from the actions followed when the patient is in other state. A disease presents alternative degrees of evolution whose treatment must be distinguished. *Decisions* capture the capability of interventions to choose alternative options depending on the available information about the patient, and therefore propose a unified representation of alternative courses that have to be applied to patients. Unlike state conditions, the conditions of a decision do not relate to the degree of evolution of the disease, but to the particular characteristics of the patient. *Actions* are the proper treatment steps of the IIP, which are selected according to the preceding decisions.

Treatments coded in SDA* are interpreted at runtime by a special agent called SDA* Engine. It provides the system with a powerful and elegant flexibility in managing and executing medical treatments.

4. Implicit Knowledge Formalization in K4Care

There is a type of knowledge in the medical context that cannot be explicitly codified in permanent documents. It is because of the strict relation with physicians capability to deduct and improvise treatment solutions. This knowledge is part of the personal skills of an actor, and it is implicitly coupled with his/her mental ability in solving emerging problems. On one side, common problems with well-patterned treatments are contained by FIPs, on the other side they are usually not directly applicable in most of the cases because of the uniqueness of patient’s conditions. In the K4Care platform, new medical knowledge is captured by defining new SDA*-based *Individual Intervention Plans* (IIPs), which are descriptions of the specific treatment that has to be provided to a particular patient.

In practice, there are two possibilities for arranging a personalized treatment: combine different standard FIPs together (illustrated in Figure 1), or model a brand new intervention plan, respectively. Regardless of each approach, physicians will have to model a new IIP by representing it in terms of states, decisions and actions involved in the provision of the new care service.

4.1. IIP Modeling Support

At the time of writing this paper, a graphical environment for supporting SDA*-based modeling is

being developed in the project. Its relevant aspects are to assist physicians and to ease the management of intervention plans (definition, maintenance and combination). An example intervention plan is illustrated in Figure 2, in which the hypertension diagnosis and treatment is emphasized.

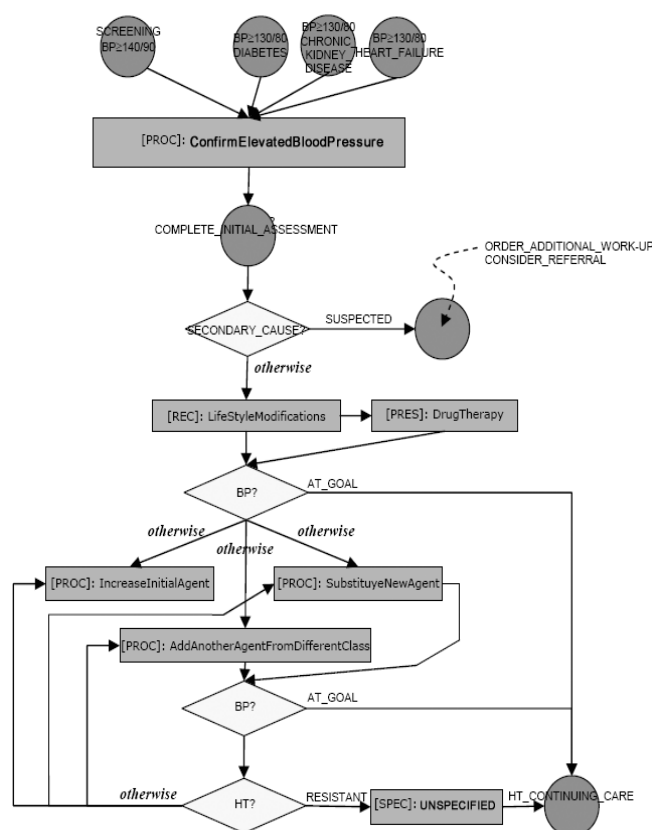


Figure 2. SDA*-based IIP - CSI's Hypertension Diagnosis and Treatment

Considering that the dynamics of an IIP are subsequently mapped to agent actions (their capabilities described in the APO), the modeling tool also provides a semantic check during the intervention definition process. This guarantees both an early avoidance of inconsistencies in domain actors role description and critical events during the treatment provision. The SDA*-based IIP execution is provided by the SDA* Engine.

IIPs are stored in the system, but they are usually not reusable in other circumstances, since they are unique treatments. It is still possible to elect an IIP to a FIP (and consequently consider it as the part of knowledge base of the system).

5. Automation in Implementing the Knowledge Driven Architecture

In this section we briefly describe how the implementation and the deployment of the presented architecture can be supported by automation. These phases are denoted by the *Implementation* and *Application* layers in Figure 1.

The concepts of the domain knowledge are defined by medical experts. Ontologies (APO, CPO) are formalized in OWL using the tool called Protégé [16]. Static procedural knowledge (FIPs) is represented in the form of SDA* descriptions. The construction of the SDA* graphs is also supported by a graphical tool.

Elementary capabilities (actions) of actors are programmed by IT experts according to the descriptions provided by the medical experts. These capabilities are bundled in a JAVA library. Electronic Health Care Record schemes are given by international standards in XML Schema Definition Language (XSD). The general SDA* interpretation logic is implemented by SDA* experts. The application knowledge is introduced by agent experts (MAS ontology) that formalizes agent technology related concepts conforming to FIPA, GAIA, JADE. The domain knowledge is then bound to application concepts in the global (K4CARE) ontology.

By formalizing the above knowledge, codes of agents in the platform can be generated automatically. For each actor in the APO an agent can be created. Capabilities of the actor (APO) are represented by JADE behaviours invoking the implementation of the corresponding action in the JAVA library. The generated SDA* Engine agent integrates the capability of interpreting arbitrary SDA* descriptions (FIPs or IIPs) and orchestrating other agents properly during its execution. The deployment of the architecture then means the assignment of agent types to real persons in the real organization. Actors can initiate procedures, create new IIPs at runtime that are performed by SDA* Engine agents. This solution guarantees the flexibility of the system. This way, when static knowledge changes, the implementation can be easily re-created. In the case of new dynamic knowledge it is applied immediately without restarting the platform.

Details of the ontology-based automated agent code generation (how different OWLs are mapped to concrete JADE-JAVA codes) are omitted here to save space. The empirical study of the resulted system will be published later.

6. Conclusions

We have presented a novel architecture for agent software development in which the role of knowledge and its description formalism are fundamental. The main feature of our architecture is represented by the fact that the development starts from the description of the whole explicit knowledge involved in the home care domain (divided into declarative and procedural knowledge) and then derives the agent system implementation for the K4Care agent platform. Among others, a fundamental aspect of our approach is the capability of the platform to recognize, manage and embed new medical knowledge, expressed in terms of new SDA*-based patient treatments. This aspect extends the actor's capabilities by deriving and combining their starting behavioral model without the necessity to re-plan or re-implement the code structure of the agents. This confers to the platform a high adaptability and flexibility to domain requirements.

We have defined the architecture and implemented the knowledge layer with the help of medical staff. According to our experiences, the knowledge layer is highly flexible. Consulting with physicians regularly, preliminary evaluations show that the representation of the medical knowledge (including intervention plans) is also acquirable, useful and user friendly for other participants of the home care system.

Acknowledgments

The authors would like to acknowledge the work of all the K4Care partners, especially Sara Ercolani, Aïda Valls, Karina Gibert, Joan Casals, Albert Solé, José Miguel Millán, Montserrat Batet and Francis Real (ontologies, data abstraction layer and service execution).

References

1. OMG Architecture Board ORMSC, *Model driven architecture (MDA)*. OMG document number ormsc/2001-07-01, <http://www.omg.org> (2001).
2. B. Henderson-Sellers and P. Giorgini, editors, *Agent-Oriented Methodologies*. Idea Group Publishing (2005).
3. A. Fuxman, L. Liu, J. Mylopoulos, M. Pistore, M. Roveri and P. Traverso, *Specifying and analyzing early requirements in Tropos*. Requirements Engineering, 9(2), pp. 132–150 (2004).
4. A. Dardenne, A. van Lamsweerde and S. Fickas, *Goal-directed requirements acquisition*. Science of Computer Programming, 20(1-2), pp. 3–50 (1993).
5. M. Cossentino, *From requirements to code with the PASSI methodology*. In: [2], chapter IV, pp. 79–106 (2005).
6. J. Pavon, J. Gomez-Sanz and R. Fuentes, *The INGENIAS methodology and tools*. In: [2], chapter IX, pp. 236–276 (2005).
7. M. Wooldridge, N.R. Jennings and D. Kinny, *The Gaia methodology for agent-oriented analysis and design*. Autonomous Agents and Multi-Agent Systems, 3(3), pp. 285–312 (2000).
8. Y. Demazeau and A.C. Rocha Costa, *Populations and organizations in open multi-agent systems*. Proc. of the 1st National Symposium on Parallel and Distributed Artificial Intelligence (1996).
9. F. Zambonelli, N.R. Jennings, A. Omicini and M. Wooldridge, *Agent-oriented software engineering for internet applications*. Coordination of Internet Agents: Models, Technologies, and Applications. Springer-Verlag, Berlin, Germany, pp. 326–346 (2001).
10. F. Bellifemine, A. Poggi and G. Rimassa, *JADE - A FIPA-compliant agent framework*. Proc. of the Fourth International Conference and Exhibition on the Practical Application of Intelligent Agents and Multi-Agents (PAAM'99), pp. 97–108 (1999).
11. Foundation for Intelligent Physical Agents, <http://www.fipa.org>
12. A. Eberhart, *Automatic Generation of Java/SQL based Inference Engines from RDF Schema and RuleML*. I. Horrocks and J. Hendler, editors, Proceedings of the First International Semantic Web Conference, pp. 102–116 (2002).
13. A. Kalyanpur, D. Pastor, S. Battle and J. Padget, *Automatic Mapping of OWL Ontologies into Java*. Proc. 16th Int'l Conf. Software Eng. and Knowledge Eng., pp. 98–103 (2004).
14. D. Fensel, *Ontologies: A silver bullet for knowledge management and electronic commerce*. Springer, Berlin (2001).
15. D. Riaño, *The SDA Model v1.0: A Set Theory Approach*. Technical Report (DEIM-RT-07-001), University Rovira i Virgili, <http://deim.urv.es/recerca/reports/DEIM-RT-07-001.html> (2007).
16. N.F. Noy, M. Sintek, S. Decker, M. Crubezy, R.W. Ferguson and M.A. Musen, *Creating Semantic Web Contents with Protégé-2000*. IEEE Intelligent Systems, 16(2), pp. 60–71 (2001).