Knowledge Driven Architecture for Home Care

Ákos Hajnal[1], David Isern[2], Antonio Moreno[2],
Gianfranco Pedone[1], and László Zsolt Varga[1]

[1] Computer and Automation Research Institute of the Hungarian Academy of Sciences,
Kende u. 13-17,
1111 Budapest, Hungary
{ahajnal,gianfranco.pedone,laszlo.varga}@sztaki.hu
[2] ITAKA research group - Intelligent Technologies for Advanced Knowledge Acquisition
Computer Science and Mathematics Department, University Rovira i Virgili
Av.Països Catalans, 26. 43007-Tarragona, Spain
{david.isern,antonio.moreno}@urv.cat

Abstract. Multi-Agent Systems (MAS) in health-care domains are showing a rapid increase, in order to manage complex tasks and adapt gracefully to unexpected events. On the other hand, the lack of well-established agent-oriented engineering methodologies to transform knowledge level descriptions into deployable agent systems slackens MAS development. This paper presents a new methodology in modelling and automatically implementing agents in a home care domain. The representation of the application knowledge together with the codification of health care treatments lead to flexible realization of an agent platform that has the capability to capture new medical knowledge emerging from physicians.

Keywords: knowledge driven architecture, home care, agent-based care system, model-driven agent generation, emerging knowledge management.

1 Introduction

Engineering software systems needs powerful abstractions to handle complexity. The agent paradigm advances the modelling of software systems by embodying a stronger notion of autonomy and control than objects, including the notion of reactive, proactive and social behaviours, as well as assuming inherent multi-threaded control. In order to be able to build complex and reliable systems, we need not only new models and technologies, but also an appropriate set of software engineering abstractions that can be used as a reference for system analysis, and as the basis of the methodologies that enable developers to engineer software systems in a robust, reliable, and repeatable fashion. A well accepted approach to engineer object oriented software systems is the Model Driven Architecture (MDA) [1] which provides a new way of writing specifications, based on a high level platform-independent model. The complete MDA specification consists of a platform-independent base UML model, plus 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

H.-D. Burkhard et al. (Eds.): CEEMAS 2007, LNAI 4696, pp. 173 – 182, 2007.

from high level models into platform-specific models in the way as it is in the MDA approach. In this paper we are advancing agent oriented software engineering methods by applying an MDA style approach to the engineering of the agent platform of the *K4Care* project in order to support home care. We call this software engineering approach as *knowledge driven architecture for home care*. There are two main differences from MDA: one is that the high level description is knowledge based compared to the object oriented model of MDA; the other is that 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.

The aim of the K4Care European project is to provide a *Home Care model*, as well as design and develop a prototype system, based on Web technology and intelligent agents that provide the services defined in the model. The percentage of old and chronically ill people in all European countries has been growing steadily in the last years, putting a very heavy economic and social pressure on all national Health Care systems. It is widely accepted that this problem can be somehow palliated if *Home Care* (HC) services are improved and used as a valid alternative to hospitalisation.

In the context of the K4Care project, we need a software engineering method that (at least partially) automates the development of an agent platform for a knowledge intensive application like home care. The intended application has two basic features: a) actors are members of well defined organizations, b) there is extensive domain knowledge to be considered. Despite similarities with the MDA approach, we have to take into account that MDA is based on UML, which is basically a notation language, not a specification one. Nevertheless, UML does not capture the behavioural aspects (dynamism) of run-time interactions. The main innovations of the paper's work are the following:

- A general methodology proposal in modelling an organizational domain, capturing and classifying the application knowledge, categorized into two main classes: *declarative* (identifying the system compositional elements) and *procedural* (capturing the behavioural logic governing the system).
- The capability of the system to capture the knowledge emerging from the behavioural skills of agents orchestrated by the interaction artefacts (SDA* Language [2]). Procedural knowledge is codified through elementary entities (States – Decisions – Actions) that, combined together, can capture the knowledge manifested at run-time by the physicians.
- Automation in the software engineering of the agent-oriented code.

The result of our investigations is a knowledge driven software engineering architecture that supports agent code generation from ontologies and codified documents for treatments, and follows a specific agent-oriented software engineering methodology.

The rest of the paper is structured as follows. Section 2 comments the most well-known paradigms in agent-oriented software engineering and the contribution of this work together with agent code generation from knowledge descriptions. Section 3 describes the *knowledge-driven architecture*. Section 4 reports the state of the realization of the proposed knowledge driven architecture in the K4Care project. The paper finishes with a brief conclusion.

2 Software Technology for Agent Systems

In the actual state of the art there is still a lack of well-established Agent-Oriented Software Engineering (AOSE) methodologies to support the automatic transformation of system description into deployable agent system. Several AOSE methodologies [3] have been proposed. Some of them offer specification-based formal verification, allowing software developers to detect errors at the beginning of the development process (e.g. Formal Tropos [4], [5]). Others inherit Object-Oriented (OO) testing techniques to be exploited later in the development process, upon a mapping of agent-oriented abstractions into OO constructs (e.g. PASSI [6], INGENIAS [7]). In addition, a structured testing process for AOSE methodologies that complements formal verification is still missing. From a theoretical point of view, different metaphors have also been proposed in the literature. The ant algorithms metaphor ([8], [9]) has shown to be useful in efficiently solving complex distributed problems. The physical metaphors ([10]) focus on the spontaneous reshaping of a system's structure and may have useful applications in pervasive and mobile computing. The societal metaphors have been effectively applied in robotics applications ([11]) and in the understanding and control of highly decentralized systems ([12]).

A formal AOSE modelling technique has been provided by the GAIA methodology ([13]), whose approach focuses on the development of large size systems that have to guarantee predictable and reliable behaviours. The metaphor adopted by GAIA is the *human organization*, which may be the most appropriate for highly populated domains ([14], [15]). A computational paradigm must be enabled by the proper technology: Multi-Agent System (MAS) platforms are the necessary infrastructures to support agents and their behavioural model. In the K4Care project we use JADE ([16]), which is a FIPA ([17]) compliant middleware for the development and run-time execution of agent-based applications. Mutually accepted standards in MAS conceptualization and implementation have not reached an adequate level of diffusion. In this context of methodological uncertainty ontologies (expressed in terms of OWL) may play an important role at modelling time, targeting the possibility to automate the engineering of the agent software. In this paper we propose the development of a complete Agent-based System for the home care domain, taking into account knowledge coming from different ontological sources (the domain model, the project requirements, the implementation and technological issues) and from medical procedures representation. 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, interpretable descriptions, and then generating the deployable code. The interpretable descriptions contain the following knowledge: the *domain*, the *agent-paradigm*, the *standards* (GAIA methodology, FIPA) and the *technology* enabling the agent paradigm (JADE). It is important to highlight the complexity and the innovation of the paper's work. Others in the literature ([18], [19]) have demonstrated that the problem of deriving code from an ontology is not a trivial task due to different factors: the *higher semantics* of the OWL language in comparison with programming languages (like JAVA, in this project); the *complexity of the domain* representation, the relations and the knowledge contained, and the *run-time code dependencies* that must be captured and included in the automation.

3 Knowledge Driven Architecture for Home Care

Developing software for medical applications and in particular for home care is special in the sense that the developed software needs to incorporate and support complex and sometimes intuitive knowledge present in the head of the medical staff, and also the medical staff is organized into a well structured organisation where the roles and responsibilities of each participant are well defined.

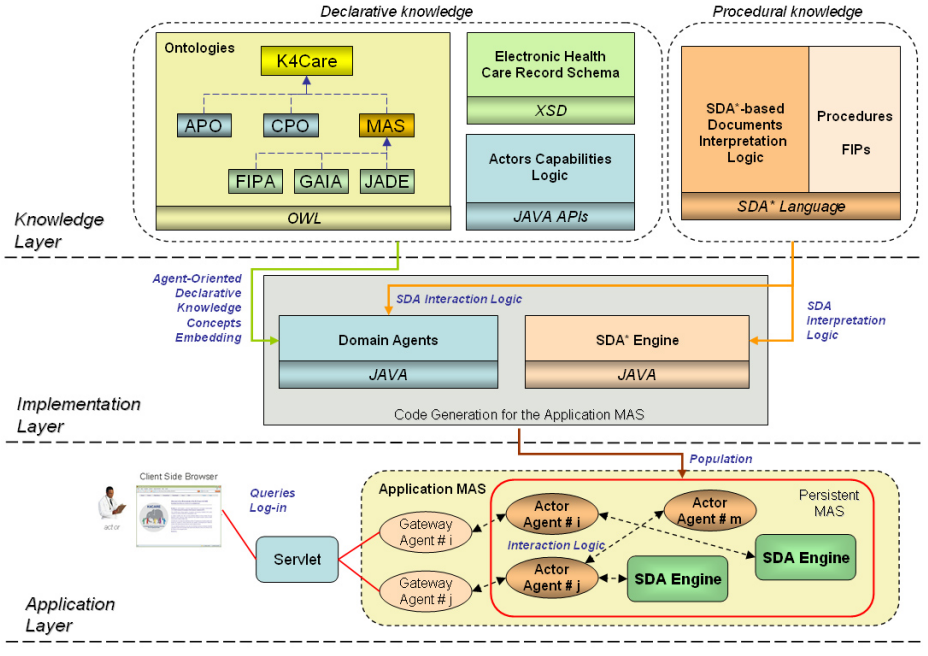

Fig. 1. Knowledge-Driven Architecture for Home Care

This reason led us to implement the software system in the K4Care project by starting from a knowledge based domain description, then deriving the software design using the GAIA methodology to achieve an agent implementation on a JADE agent platform. The most relevant aspect of this knowledge driven architecture is the separation of knowledge description from the software realization, granting a high level of interoperability and, at the same time, high level of independence among elements of the system. Key elements of the architecture are the *declarative knowledge*, the *procedural knowledge*, the *agent-oriented code generation*, the *agent behavioural logic distribution*, and the *interaction between agents and end-users*. The knowledge driven architecture for home care and its elements are shown in Figure 1.

The declarative knowledge and the procedural knowledge form the Knowledge Layer. The agent code generation forms the Implementation Layer. Finally, the agent behavioural logic distribution and the interaction between agents and end-users form

the Application Layer. The Knowledge Layer describes user's knowledge in a high level and the Application Layer is the targeted K4Care agent platform. In the following we are describing these elements.

3.1 Declarative Knowledge

Ontologies are a standard AI knowledge representation mechanism ([20]). Their main components are a set of concepts (or classes) *C*, which are taxonomically related by the transitive *is-a* relation and non-taxonomically related by named object relations $R^* \varepsilon C^*C^*String$. They represent the knowledge assets of the K4Care application and the catalyst for the agent's behavioural model, as well as the fundaments for the agent's code generation. We have divided the *application ontologies* into different sources, in relation to their application coherence. All the ontologies are independent of each other, except of the *K4Care* global one, which includes all the others and contains necessary cross-references to relate concepts coming from different elementary ontologies. The application ontologies are:

- **Domain ontology**, divided into *Actor Profile Ontology* (APO) and *patient-Case Profile Ontology* (CPO). APO represents the profiles of the subjects in the *K4Care* model (healthcare professionals, patients and social organisms) and contains the skills, concerns, aspirations of the people that they represent, together with the healthcare services that those people offer to or receive from the *K4Care* model. CPO represents the relevant medical concepts (e.g. symptoms, diseases, syndromes). Domain ontologies describe "know-what" knowledge about actors accessing the *K4Care* model and pathologies the *K4Care* model gives support to.
- **MAS ontology**, whose double role is to establish a common conceptualization of the Multi-agent System architecture and to enable the automation of its code generation. The MAS ontology includes the following elements:
  - **FIPA ontology.** This is the general conceptualization in the standardization of a MAS development. It represents "know-what" knowledge of a MAS (messages, encodings, communication languages, and so on) and "know-how" knowledge of their interactions (communicative acts, interaction protocols, communication ontologies, and so forth).
  - **GAIA ontology.** It includes the ontological description of the MAS development methodology adopted within the K4Care project, with particular attention to concepts such as *role*, *responsibility*, *permission* ([13]).
  - **JADE ontology.** It expresses the implementation and deployment concepts that must comply with the elected MAS of the project, i.e. JADE agent-types, behaviours, containers, mobility, and so forth ([16]).
- ***K4Care* ontology.** This ontology is necessary in order to model the inevitable ontology-cross references: it represents the way implementation ontologies (MAS) previously mentioned are connected to the domain specific ontologies (APO, CPO). The agent capabilities, for example, are expressed in terms of "*actions*" in the APO; these are considered as "*responsibilities*" or

"*permissions*" by GAIA; the behavioural logic of an agent inherent to its capabilities is expressed in JADE in terms of "*behaviours*", which can contain FIPA compliant "*interaction protocols*", as well. Despite its undoubted importance, this ontology has a small dimension, in terms of relations and items contained.

The *Electronic Health Care Record* in Figure 1 represents the patient's care profile and it is consulted by physicians during the stages of a treatment. *Actor Capabilities*, in addition, are expressed in terms of primitives and they are developed to define an actor's basic skill within the system. They represent the agent's permanent capabilities. The global upper-level ontology (composed in OWL) containing all the relevant concepts in K4Care project consists of about 500 classes, and 70 properties.

3.2 Procedural Knowledge

In the K4Care project we have separated the "*static agent capabilities*", that are defined through the APO ontology in terms of "*actions*", and represents "*what*" an agent is able to do, from the "*dynamic behavioural capabilities*", that emphasize the way static agent capabilities can be combined together in order to define much more complex emerging *interactions* following the care service procedures of the domain. The procedural knowledge is formally described in different fundamental elements:

- *Procedures*: descriptions of the steps to be taken within the *K4Care platform* to provide one of the HC services.
- *Formal Intervention Plans (FIPs)*: general descriptions, defined by health care organisations, of the way in which a particular pathology should be treated.
- *Individual Intervention Plans (IIPs)*: descriptions of the specific treatment that has to be provided to a particular patient.

Procedural knowledge codifies complex medical tasks and is required to define the set of available actions performed by all actors in the platform. This knowledge is expressed in the K4Care project using a flowchart-based representation called SDA* ([2]). A small example of a formal SDA*-based representation (depression with dementia) is shown in Figure 2.

3.3 Agent Behavioural Logic Distribution

A single actor (agent) in the platform owns a limited set of behavioural capabilities that represents its role in the home care organizational context. Despite this limited number of skills, the agents' capabilities can be orchestrated and combined by physicians in order to obtain much more complex global behaviours from the entire system. New treatments are coded into SDA* graphs that are interpreted at run-time by a special agent (SDA* Engine in Figure 1), whose objective is to govern the complexity of the treatments representation and to lead agents interactions (distribution) toward the provision of the requested services. This gives the system a powerful and elegant flexibility in managing and executing new treatments and medical guidelines.

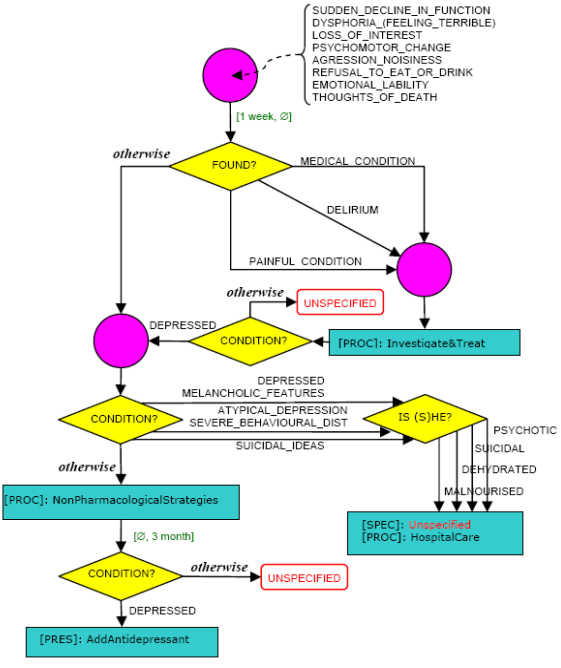

Fig. 2. SDA* to manage the Depression with Dementia

The interaction level of an SDA* graph is determined by evaluating the treatment parameters that characterize a specific node in the graph (patient conditions) first, and then accessing the correct interaction relation. These relations can be imagined as a space of 4-tuples, whose components indicate a *subject* (the actor who is asked to provide an action in the specific node), an *object* (the actor who will receive the benefits of the *subject*'s action), a *service* (the type of service requested) and a *document* (this represents the document that must be officially updated during the patient's treatment, in compliance with a specific procedure).

3.4 Interaction Between Agents and End-Users

There is a clear separation between the view, the control and the model within the K4Care project. The user interface (*view*) is connected to the MAS (*model*) via a servlet (*controller*) and all communications towards and from the multi-agent system occur by the support of a gateway-agent (one per permanent agent/actor in the MAS), which is strictly coupled with the servlet. The front-end of the interaction is represented by a web-application, whose structure and content are customized in relation with different relevant factors (such as the patient profile's insight or the interaction device – PC, PDA, Notebook).

4 Knowledge Driven Architecture Realization in K4Care

We applied the proposed knowledge driven architecture to the development of the K4Care system. We have realized the knowledge-based description of the domain knowledge (ontologies, procedural documents, agents capabilities description). Our experience demonstrated that medical experts find it sufficiently easy to describe and validate their expertise using ontologies. Despite the undoubted but minimal initial effort in familiarizing with ontology theory and supporting tools, medical experts considered our approach an elegant and powerful method for homogeneously representing medical knowledge (together with treatment guidelines documents). We have already realized all domain ontologies, defined within K4Care: the APO, the CPO and the MAS ontology (FIPA+GAIA+JADE) introduced in the previous section. K4Care global ontology contains the object properties that connect and bridge the different ontological layers (as illustrated in Figure 1).

As already mentioned, an important aspect of our project is the automation in the generation of agent code. Starting from the application knowledge, the accuracy and the level of detail in the code is directly proportional to the formalism's accuracy in describing the domain model. Particularly important is the representation of the agent capabilities, which, in practice, consists of a formal description of their business logic through API invocations. The objective of this automation process was to analyze, recognise and extract all the MAS compliant agent-oriented information: we had to take into account, first of all, the implementation requirements deriving from the elected MAS structure (agent class definition, protocols, message structures, message contents, behaviours class). In order to guarantee flexibility in the ontological traversing and parsing, and to decouple the conceptualization layer from the parsing one, the introduction of a mapping *dictionary* was necessary. This *thesaurus* described to the parser the "root" level of concepts where to start collecting agents' knowledge from. All relevant extracted information has been temporarily represented in an internal model. We have obtained agents code by combining together two categories of elements: *fixed and invariable* (keywords related to programming language, JAVA, and the MAS, JADE), and *knowledge-model-derived*, necessary to embed expressions which comply with the correct code completion of an agent. The agent's capabilities (domain knowledge) are represented by its actions, listed in the APO: from the paradigm's point of view, these must be translated in terms of behavioural models accepted by the MAS. The code was built by taking into account the following aspect: different behaviours have been dedicated to different skills in order to interact in parallel with other entities. Most of the effort in the agent code generation was also dedicated to obtain a harmonious code arrangement of all concepts (elements) characterizing the domain model and the application development.

The actual state in the realization of the described methodology highlights its real feasibility. Application ontologies are formalized in sufficient detail with respect to the goals of the project. Further details of the automated agent code generation based on ontologies are omitted here for space considerations.

5 Conclusions

We have presented a novel knowledge driven architecture for agent software development. The main feature of our architecture is represented by the fact that the development starts from the description of the knowledge involved in the home care domain (divided into declarative/static and procedural/dynamic/emerging) and then automatically derives the agent system implementation for the K4Care agent platform. Among others, a fundamental aspect of our approach is the platform's capability to recognize, manage and embed the emerging of new medical knowledge, expressed in terms of new SDA*-based patient treatments. This aspect extends the actor's (agent) capabilities, by deriving and combining their starting behavioural model, without the necessity to re-plan or re-implement the code structure of the agents. We have defined the knowledge driven architecture and implemented the knowledge layer with the help of medical staff. According to the experiences the knowledge layer is highly flexible and user friendly. This paper described the general concept of the novel knowledge driven architecture, leaving out the details like the agent code generation or the SDA* Engine due to space limitations.

Acknowledgements

The authors would like to acknowledge the work of all the K4Care partners, especially Fabio Campana, Roberta Annicchiarico and the medical partners (K4Care model), David Riaño (SDA* formalism), 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). This work has been funded by the European IST project *K4Care: Knowledge Based Home Care eServices for an Ageing Europe* (IST-2004-026968).

References

1. OMG Architecture Board ORMSC: Model driven architecture (MDA). OMG document number ormsc/2001-07-01 (July 2001), http://www.omg.org
2. Riaño, D.: The SDA model v1.0: A set theory approach. Technical Report (DEIM-RT-07-001), University Rovira i Virgili (2007), http://deim.urv.es/recerca/reports/DEIM-RT-07-001.html
3. Henderson-Sellers, B., Giorgini, P. (eds.): Agent-oriented methodologies. Idea Group Publishing, USA (2005)
4. Fuxman, A., Liu, L., Mylopoulos, J., Pistore, M., Roveri, M., Traverso, P.: Specifying and analyzing early requirements in Tropos. Requirements Engineering 9(2), 132–150 (2004)
5. Dardenne, A., van Lamsweerde, A., Fickas, S.: Goal-directed requirements acquisition. Science of Computer Programming 20(1-2), 3–50 (1993)
6. Cossentino, M.: From requirements to code with the PASSI methodology. In: [3], ch. IV, pp. 79–106 (2005)
7. Pavon, J., Gomez-Sanz, J., Fuentes, R.: The INGENIAS methodology and tools. In: [3], ch. IX, pp. 236–276 (2005)
8. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm intelligence: From natural to artificial systems. Oxford University Press, New York (1999)

9. Babaoglu, O., Meling, H., Montresor, A.: Anthill: A framework for the development of agent-based peer-to-peer systems. In: Proceedings of the 22nd International Conference on Distributed Computing (2002)
10. Mamei, M., Zambonelli, F., Leonardi, L.: Distributed motion coordination with co-fields: A case study in urban traffic management. In: Proceedings of the 6th Symposium on Autonomous Decentralized Systems, p. 63. IEEE Computer Society Press, Los Alamitos (2003)
11. Moses, Y., Tennenholtz, M.: Artificial social systems. Computers and Artificial Intelligence 14(6), 533–562 (1995)
12. Hattori, F., Ohguro, T., Yokoo, M., Matsubara, S., Yoshida, S.: Socialware: Multiagent systems for supporting network communities. Communications of the ACM 42(3), 55–61 (1999)
13. Zambonelli, F., Jennings, N.R., Wooldridge, M.: Developing multiagent systems: the Gaia Methodology. ACM Trans. on Software Engineering and Methodology 12(3), 317–370 (2003)
14. Demazeau, Y., Rocha Costa, A.C.: Populations and organizations in open multi-agent systems. In: Proceedings of the 1st National Symposium on Parallel and Distributed Artificial Intelligence (1996)
15. Zambonelli, F., Jennings, N.R., Omicini, A., Wooldridge, M.: Agent-oriented software engineering for internet applications. In: Wooldridge, M.J., Weiß, G., Ciancarini, P. (eds.) AOSE 2001. LNCS, vol. 2222, pp. 326–346. Springer, Heidelberg (2002)
16. Java Agent DEvelopment Framework, http://jade.tilab.com/
17. Foundation for Intelligent Physical Agents, http://www.fipa.org/
18. Eberhart, A.: Automatic generation of Java/SQL based Inference engines from RDF Schema and RuleML. In: Proceedings of the First International Semantic Web Conference, pp. 102–116 (2002)
19. Kalyanpur, A., Pastor, D., Battle, S., Padget, J.: Automatic mapping of OWL ontologies into Java. In: Proceedings of the 16th International Conference on Software Engineering and Knowledge Engineering, pp. 98–103 (2004)
20. Fensel, D.: Ontologies: A silver bullet for knowledge management and electronic commerce. Springer, Berlin (2001)