COMPUTER METHODS AND PROGRAMS IN BIOMEDICINE 134 (2016) 267–287

journal homepage: www.intl.elsevierhealth.com/journals/cmpb

A methodology based on *open*EHR archetypes and software agents for developing e-health applications reusing legacy systems

João Luís Cardoso de Moraes [a,*], *Wanderley Lopes de Souza* [a], *Luís Ferreira Pires* [b], *Antonio Francisco do Prado* [a]

[a] *Federal University of São Carlos, Computer Department, Rodovia Washington Luís-Km 235, 13565-905 São Carlos-SP, Brazil*
[b] *University of Twente, Centre for Telematics and Information Technology, Drienerlolaan 5, 7522 NB, Enschede, The Netherlands*

ARTICLE INFO

Article history:
Received 29 March 2016
Accepted 4 July 2016

Keywords:
Pervasive healthcare
Archetypes
*open*EHR
Software agents
Semantic interoperability

ABSTRACT

Background and objective: In Pervasive Healthcare, novel information and communication technologies are applied to support the provision of health services anywhere, at anytime and to anyone. Since health systems may offer their health records in different electronic formats, the *open*EHR Foundation prescribes the use of archetypes for describing clinical knowledge in order to achieve semantic interoperability between these systems. Software agents have been applied to simulate human skills in some healthcare procedures. This paper presents a methodology, based on the use of *open*EHR archetypes and agent technology, which aims to overcome the weaknesses typically found in legacy healthcare systems, thereby adding value to the systems.
Methods: This methodology was applied in the design of an agent-based system, which was used in a realistic healthcare scenario in which a medical staff meeting to prepare a cardiac surgery has been supported. We conducted experiments with this system in a distributed environment composed by three cardiology clinics and a center of cardiac surgery, all located in the city of Marília (São Paulo, Brazil). We evaluated this system according to the Technology Acceptance Model.
Results: The case study confirmed the acceptance of our agent-based system by healthcare professionals and patients, who reacted positively with respect to the usefulness of this system in particular, and with respect to task delegation to software agents in general. The case study also showed that a software agent-based interface and a tools-based alternative must be provided to the end users, which should allow them to perform the tasks themselves or to delegate these tasks to other people.
Conclusions: A Pervasive Healthcare model requires efficient and secure information exchange between healthcare providers. The proposed methodology allows designers to build communication systems for the message exchange among heterogeneous healthcare systems, and to shift from systems that rely on informal communication of actors to a more automated and less error-prone agent-based system. Our methodology preserves significant investment of many years in the legacy systems and allows developers to extend them adding new features to these systems, by providing proactive assistance to the end-users and increasing the user mobility with an appropriate support.

* Federal University of São Carlos, Computer Department, Rodovia Washington Luís-Km 235, 13565-905 São Carlos-SP, Brazil. Fax: +55 16 33518233.
E-mail address: joao_moraes@dc.ufscar.br.

http://dx.doi.org/10.1016/j.cmpb.2016.07.013
0169-2607/

1. Introduction

In most countries, the conventional healthcare model will soon become inadequate, due to the increasing healthcare costs for the growing population of elderly people, the rapid increase in chronic disease, the growing demand for new treatments and technologies, and the decrease in the number of health professionals relative to the population increase [1]. Recently, the United States Census Bureau estimated that the expected number of inhabitants in the United States older than 65 will be approximately 70 million in 2030, twice that in 2000 [2]. In Ontario, the most populous province of Canada, healthcare is expected to represent 66% of government expenditure in 2017, and 100% in 2026 [3].

Currently, most healthcare organizations rely on legacy information systems, which are difficult to maintain and evolve since they are not flexible enough to incorporate the new requirements frequently demanded by the end-users [4]. Many legacy information systems have been designed to support end-users working in static organizations, and they typically do not profit from recent advances in Information and Communication Technologies (ICT) [5]. In addition, information exchange between these systems still requires users' intervention and telephonic conversations between these users, while the expenses for redesigning or replacing these systems are often huge mainly due to their poor structure and complexity [6].

In Pervasive Healthcare, ICT is applied to support the provision of health services anywhere, at anytime and to anyone [7]. In order to interoperate in Pervasive Healthcare environments, heterogeneous Electronic Health Record (EHR) systems require the use of communication standards [8,9]. *open*EHR [10] is a foundation dedicated to the research on interoperable EHR, and has defined an open architecture based on a two-level model that separates information from knowledge. *open*EHR prescribes the use of archetypes for describing clinical knowledge, in which an archetype is a structure employed by a domain expert to represent some specific knowledge within this domain.

Software agents [11] are entities that employ Artificial Intelligence techniques to choose the optimal set of actions to be performed in order to achieve the goals specified by their users. They can communicate with each other and with their users, and they have properties such as sociability, proactivity and autonomy, which allow them to support their users in their daily activities. The cooperation and coordination abilities of two or more agents can be combined through the use of well-defined communication rules for building a Multi-Agent System (MAS) to cope with complex tasks [12]. In healthcare, software agents can help healthcare professionals exchange healthcare information during their routine tasks [13,14].

This paper proposes a methodology that employs *open*EHR archetypes and software agents to cater for the interoperability between legacy healthcare systems. In this work, we explored the challenges of creating archetypes in the cardiology domain using the archetype methodology and tools to formalize the representation of clinical information within the EHR. Furthermore, we investigated the challenges of importing and integrating archetypes into the healthcare applications. We studied several healthcare environments and identified the following main requirements that make agent technology suitable to be used in these environments: deal with the distributed clinical information to be shared by the stakeholders; provide a fast and safe communication among healthcare professionals and patients; keep the autonomy of healthcare providers; and gather clinical information proactively from heterogeneous healthcare information systems. Our challenge has also been to keep the original legacy application running while moving to a pervasive healthcare model, making use of new technologies and skills.

As a case study, we designed a system using our methodology that supports the preparation of a cardiac surgery by reusing legacy Hospital Information Systems (HIS). The remainder of this paper is organized as follows: Section 2 introduces the organization and problems of the Cardiac Department of a hospital in the city of Marília (São Paulo, Brazil) that inspired our research; it provides some background on the *open*EHR model and archetypes; it discusses the software agents employed in this methodology; and it discusses some related work. Section 3 presents the proposed methodology; Section 4 describes the case study and discusses our evaluation results; and Section 5 gives some concluding remarks and topics for future work.

2. Background

2.1. *Initial situation*

Cardiac surgery is one of the best examples of teamwork in surgery, since it requires the full integration of individual efforts with maximum efficiency to make sure that each action plan is performed successfully [15]. Cardiac surgery is performed by a work group of highly trained staff, here named *Heart Team*, which consists of a cardiovascular surgeon, who leads the surgical team; an assistant surgeon, who follows the instructions of the cardiovascular surgeon; a cardiovascular anesthesiologist, who administers the drugs to keep patients asleep during surgery; a perfusionist, who operates the cardiopulmonary bypass machine; and cardiovascular nurses, who are specially trained to assist during the cardiac surgery.

The Center of Cardiac Surgery of Marília (CCCM) is the Cardiology Department of the Santa Casa Hospital that provides ongoing follow-up care for the cardiac surgery procedure in Marília and surrounding cities, and it has a clinical HIS, named CCCMSys, to keep track of the care provided to its patients. The scheduling of cardiac surgeries involves a procedure that consists of the following steps: (1) check the availability of resources in Santa Casa Hospital, such as blood bank, Intensive Care Unit (ICU) and surgical center operating room, since Santa Casa has one blood bank, eight ICU beds, and two operating rooms; (2) set a date for a meeting of the Heart Team to discuss the surgery, for which medical reports containing information on the patient's EHR need to be obtained in advance from the cardiology clinics in which the patient has been treated; and (3) notify the patient about the time slot when the surgery is expected to take place. Quite often, the actors involved in this procedure (and in other clinical activities) rely on informal offline communication (telephone or fax) to perform their tasks.

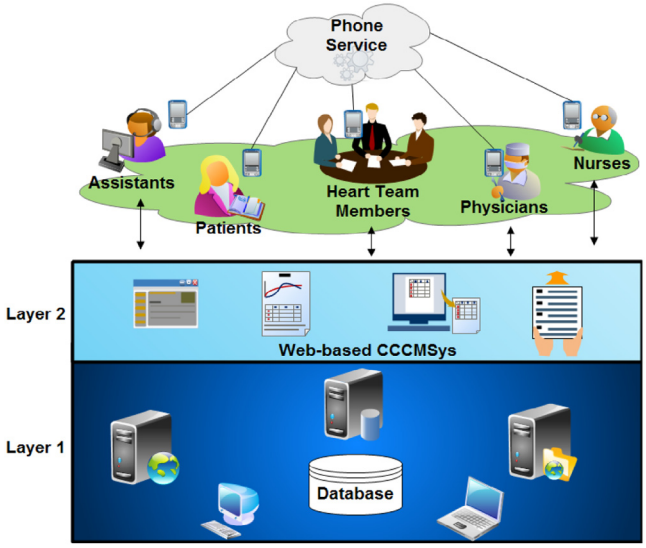

Fig. 1 – Current CCCMSys.

Fig. 1 shows the original CCCMSys as a two-layered system: (1) an information system infrastructure (back-end) consisting of heterogeneous hardware, software and databases running in a computer network; and (2) the Web-based CCCMSys front-end. In addition, human actors, including the healthcare professionals of the cardiology clinics and Santa Casa Hospital units, exchange information related to their clinical activities via telephone or fax using the phone service.

We investigated the legacy healthcare information system (CCCMSys) and the healthcare organizations, which exchange clinical information with the Cardiology Department (CCCM) in order to properly understand their daily tasks and organizational processes. In the daily tasks of the CCCM, we observed several facts concerning its scheduling of cardiac surgeries:

(1) the CCCMSys does not inform the Heart Team members automatically when the resources necessary for the prevention and treatment of diseases become available, such as blood of proper type, the surgical center room and ICU beds;
(2) the healthcare professionals who use the legacy healthcare information systems in the Cardiology Department of the hospital spend extra time searching and retrieving clinical information about their patients when using the legacy system. Therefore, Heart Team members have less time available for their main medical activities due to the extra time spent by using these legacy systems;
(3) in synchronous communication forms, such as in a phone call, the participating parties must both be present for the communication to take place and, therefore, time is wasted if one party cannot reply to the other;
(4) even though some cardiology clinics keep medical reports online, the quality of these reports depends on human factors. For instance, any error by an assistant, such as exchanging clinical information of two namesake patients when requesting medical reports from a physician, may have consequences for the patient's treatment; and
(5) assistants are responsible for the scheduling of cardiovascular treatments and the notification of appointment dates to patients. If patients miss their appointments, time is wasted, obstructing the provision of healthcare and decreasing the number of patients that can be attended.

2.2. openEHR *and archetypes*

The *openEHR* architecture was developed based on a two-level modeling approach, as shown in Fig. 2. On the first level, a common *Reference Model* (RM) was defined in terms of a predefined set of classes that model the structure of an EHR; and on the second level, specific concepts were defined by restricting the RM classes in terms of so-called *archetypes*, expressed in the Archetype Definition Language (ADL) [10,16,17].

An archetype consists of a formal model of a domain concept, designed by a domain expert and not by an ICT professional, which can be translated to any implementation

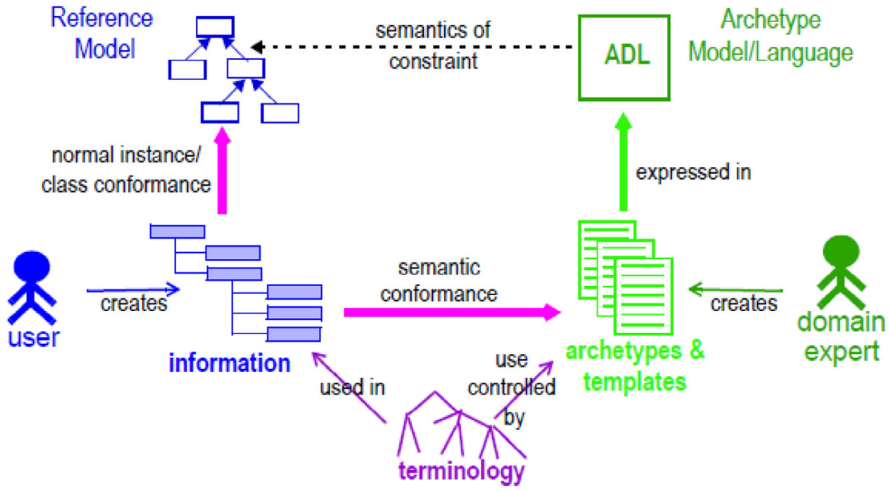

Fig. 2 – *open*EHR two-level model [Reprinted with permission of Thomas Beale and Sam Heard] (http://www.openehr.org/releases/BASE/latest/docs/architecture_overview/diagrams/archetype_meta_architecture.png).

```
1  archetype (adl_version=1.4) openEHR-EHR-ACTION.cardiac_surgery.v1
2  [...]
3  definition
4    ACTION[at0000] {--Cardiac Surgery Procedure
5      description{
6        ITEM_TREE[at0001]{--Tree
7          items cardinality {0..*; unordered}{
8            CLUSTER[at0008] occurrences{0..1}{--Procedures and Cavity
9              items cardinality {0..*; unordered}{
10               ELEMENT[at0002] occurrences{0..*}{    -- Procedures
11                 value matches {
12                   DV_TEXT matches {*}}}
13 [...]
14 ontology
15   term_definitions = <
16     ["en"] = <
17       ["at0000"] = <
18         text = <"Cardiac Surgery Procedure">
19         description = <"Procedure to treat heart disease[...]">
20 [...]
```

Fig. 3 – Excerpt of the cardiac surgery archetype.

language. In accordance with the two-level modeling approach [18], data from users are stored according to the RM, but must also comply with the concepts expressed by the archetypes. This separation should facilitate the interpretation of the knowledge extracted from the messages exchanged between health systems in various medical applications.

In the *open*EHR RM [18], the COMPOSITION class refers to one or more instances of the SECTION class, each containing ENTRY objects. The ENTRY class represents the clinical content recorded during a patient Observation, Examination, Assessment or Intervention. ENTRY is defined as an abstract type that has four concrete subclasses: (1) OBSERVATION, which can be used to represent clinical observations, such as the measurement of blood pressure; (2) EVALUATION, which can be used to represent assessments made after a clinical observation is completed, such as risk assessment; and (3) INSTRUCTION and (4) ACTION, which are both typically used to represent surgical procedures, medication and other clinical interventions, and the taken actions, respectively. The ACTION subclass describes what was done and committed to the EHR as the result of an INSTRUCTION.

The Clinical Knowledge Manager (CKM)[1] is the *open*EHR archetype repository, and contains a set of archetypes that can be reused in various healthcare applications. In our experiments, we have reused some CKM archetypes, such as *Device Details*, *Result Report*, *ECG Recording*, *Patient Admission* and *Clinical Synopsis*, but we have also developed new archetypes to represent clinical concepts within the cardiology domain, such as *Cardiac Surgery*, *Pacemaker Implantation*, *Angioplasty Cardiac* and *Pacemaker Evaluation* [8].

According to the *open*EHR specifications [10], an archetype consists of three sections: (1) *header* contains a unique identifier for the archetype, and includes some descriptive information, such as author, version, and status; (2) *definition* expresses the restrictions in a tree structure created from the RM, and this structure ensures that both the cardinality and the content of the information model instances comply with the archetype; and (3) *ontology* contains the code that represents the meaning of nodes, the constraints on text and terms, and the bindings to terminologies, such as, for example, SNOMED-CT.[2]

Fig. 3 shows an excerpt of the *Cardiac Surgery* archetype, which was defined according to *open*EHR: *header* includes the name of the archetype (line 1); *definition* (lines 3–13) contains the structure and restrictions associated with the clinical concept defined by the archetype; and *ontology* (line 14–20) includes the terminological definitions and associates the linguistic expression “Cardiac Surgery Procedure” with the code “at0000”. In this example, Cardiac Surgery specializes the ACTION class (line 4) of the RM, the CLUSTER part (line 8) refers to the procedures and opening cavity for cardiac surgery and consists of an ELEMENT (line 10) with a value of type DV_TEXT (line 12) that should contain some textual description.

The archetype formalism has been used to represent clinical practice guidelines, allowing healthcare professionals to access clinical information during consultations. Studies have shown that archetypes help reduce medical errors, standardize clinical concepts and improve the quality of care [19–22].

[1] http://www.openehr.org/ckm/.

[2] http://www.ihtsdo.org/snomed-ct/.

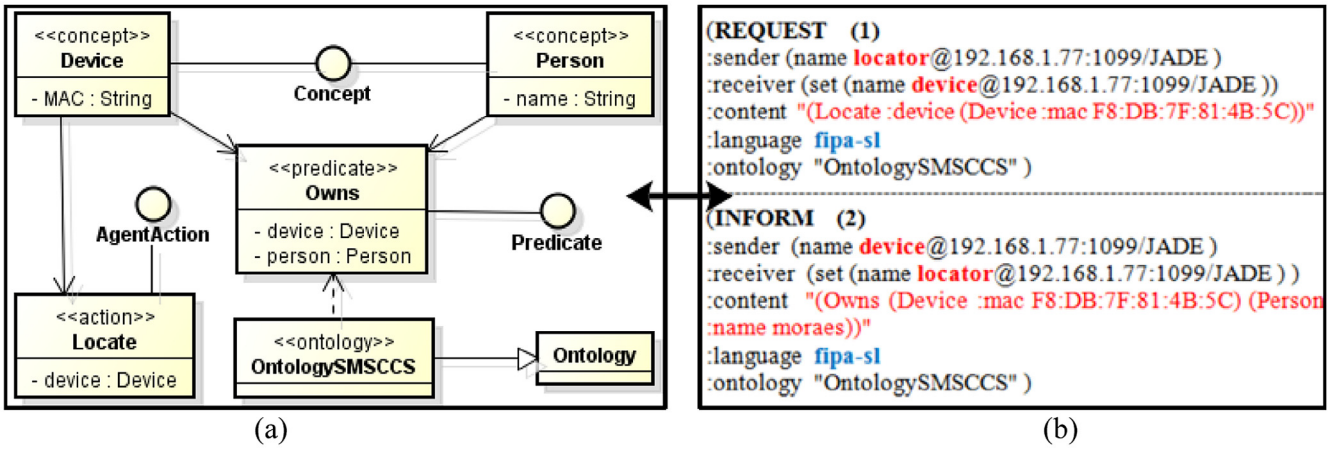

Fig. 4 – Example of ontology for *Device Location* and related messages [42].

2.3. *Software agents*

An agent is a computational entity with autonomous behavior, which allows it to take decisions and perform its own actions [11]. A software agent is an entity that operates continuously and autonomously in a particular environment and is capable of intervening in this environment without requiring constant human guidance. According to its intelligence level, an agent can be classified as *cognitive*, if it has significant cognition abilities, or *reactive* otherwise. An agent is able to move in its environment, e.g., by traveling between several hosts in a computer network. Although multiple agents can inhabit a common environment, each agent must have its own purpose and characteristics in order to ensure its autonomy [23,24].

Agents in a Multi-Agent System (MAS) require the use of a language they all understand to guarantee proper information exchange. The Agent Communication Language (ACL) [25], proposed by the Foundation for Intelligent Physical Agents (FIPA) [26], is often used for this purpose. The FIPA specifications prescribe the rules for developing multi-agent systems. FIPA-ACL is based on the theory of speech acts, and benefits from the formal definition of its *performative* library [27] with communicative acts, such as *INFORM*, *REQUEST*, *REFUSE*, *QUERY-IF*, *CONFIRM* and *AGREE*. A FIPA-ACL message is encapsulated as one of the message attributes (*content*), and the FIPA Semantic Language (FIPA-SL)[3] can be used to represent message content, since it is based on first-order logic, enabling the agents to precisely express the properties and relations between objects of a specific domain.

Multi-agent platforms, like JADE [28], JASON [29] and JACK [30], are software frameworks that support the implementation and execution of MAS. Regardless of the differences in scope and objectives, they mostly provide facilities to developers, such as agent communication languages and monitoring tools. In our work, we employed the JADE platform since it provides an efficient messaging service, and it offers a better performance for running a MAS in a distributed environment, if compared with most of the other available agent platforms [31–34]. JADE is FIPA-compliant and allows an interoperable MAS to be implemented in Java [35]. The JADE platform allows the coordination of agents and enables them to communicate with each other using the standard FIPA-ACL communication language. According to Refs. [36, 37], "JADE uses the *Behavior* abstraction to model the tasks that agents are able to perform, allowing agents to instantiate behaviors according to their needs and capabilities".

JADE supports ontologies, which help define the names and types of data to be used in the communication between agents, according to the syntactic rules of the related content language (FIPA SL) [38]. Agents can declare the ontology they apply to interpret the message they exchange. Exchanged messages can be interpreted unambiguously by all participants by using an ontology, since each message encodes some particular information expressed in the ontology, and agents ascribe the same meaning to the content of the messages [39–41]. Several types of ontology schemes can be defined by applying the following stereotypes [38]: *concept* denotes ontology concepts; *predicate* denotes expressions that relate concepts to one another and can be evaluated as true or false; and *action* denotes actions that can be performed by an agent.

In JADE ontologies [38], *concepts* are expressions used only for interpretation, which represent entities of the real world and correspond to unary predicates in first-order logic, while *predicates* and *actions* are represented in the exchanged messages. Fig. 4 shows an ontology example used for device location [42]. Fig. 4(a) shows the *Device* and *Person* concepts, the *Owns* predicate, which indicates that a device has a given owner, and the *Locate* action, which indicates that an agent must locate the owner of a device. Fig. 4(b) shows two FIPA-ACL messages indicated with the FIPA-performative REQUEST and INFORM. The *locator* agent sends a request message (REQUEST) to the *device* agent. The message is written in FIPA-SL and its content is related to *OntologySMSCCS*. The message indicates that the MAC address *F8:DB:7F:81:4B:5C* of a device was located. In its turn, the *device* agent returns an informative message (INFORM) to the *locator* agent. The message indicates that the owner of the device with MAC address *F8:DB:7F:81:4B:5C* is the person called *moraes*.

[3] http://www.fipa.org/specs/fipa00008/.

Software agents have been applied in the healthcare domain to deal with several issues [43–48], such as: to maintain the autonomy of the team members; to integrate the clinical information obtained from heterogeneous sources; to adapt user interfaces based on user requirements and clinical information; and to coordinate clinical information exchange in distributed healthcare environments. In this work we advocate the use of agent technologies to automate part of the healthcare professionals' daily tasks, giving support to the decision-making steps, and to share clinical knowledge related to their patients.

2.4. *Related work*

Many e-health applications built with software agents and complying with healthcare standards have been reported in the literature.

An MAS for controlling the medication of patients as well as the available stock of medicines is proposed in Ref. [49]. That work has some similarities with ours, since both use MAS to control clinical tasks. However, in Ref. [49] healthcare standards are not employed to enforce interoperability at the message level between agents, which brings difficulties when sharing and reusing legacy systems. This solution does not solve the interoperability problem.

A distributed coordination framework for dynamically exchanging EHRs between health communities is proposed in Ref. [50]. These communities are connected by means of a P2P model, and they use a multi-agent platform and a set of distributed rules to coordinate the agents in the search of specific health records. This work has some similarities to ours, since both use MAS for coordinating the exchange of EHR content and focus on integration healthcare information systems, but their work does not employ a healthcare standard for the exchange of clinical concepts. Our work employs the *open*EHR dual model for exchanging patient clinical information in order to improve the semantic interoperability between heterogeneous healthcare information systems.

In Ref. [51], a virtual organization, for the procurement of organs and tissues for transplantation purposes, is proposed as an electronic institution, and described by means of an institution specification language. For managing and processing of the huge amount of data involved in this procurement, an agent-based architecture is also proposed. This work has some similarities with ours, since both use MAS to deal with distributed healthcare information, but it does not employ *open*EHR standards. Furthermore, the main concern in our work was to preserve the investment in the legacy systems by allowing developers to add new features to them, while in Ref. [51] the main concern was to preserve privacy, security and trust in relation to matters such as agents' access to patient records. These are issues that we intend to investigate and incorporate in our system in future work.

A communication system is reported in Ref. [52] in which mobile devices recognize the context in which healthcare professionals perform their tasks. The authors propose an extension of the traditional Instant Messaging paradigm by using the Extensible Messaging and Presence Protocol (XMPP) for exchanging XML messages. These messages contain contextual information, including the essential information that allows the system to deliver messages. Similarly to our work, in this system, agents are responsible for message exchange. However, the system reported in Ref. [52] does not apply any standard language for agents' communication, and it disregards healthcare standards for the exchange of clinical concepts. This system does not apply a standardized ontology for capturing the meaning of the message content exchanged between the agents either, so that interoperability is not guaranteed in this system.

A proposal for the representation and persistence of clinical data of patients as well as contextual information in ubiquitous applications is described in Ref. [53]. This work is based on the *open*EHR two-level modeling, and the persistence solution stores an XML representation of a reference model indexed with data paths defined in archetypes. This work has some similarities to ours, since both use the *open*EHR two-level modeling. However, this work does not take advantage of an agent communication language for the exchange of patient clinical data.

An approach to provide interoperability between self-care systems when they exchange non-clinical information alongside clinical data is proposed in Ref. [54]. This work is based on the use of web services for the communication between heterogeneous systems, where SOAP messages are defined for transporting the Personal Health Record (PHR) content in structures of the so-called Health Diary Entry (HDE). This allows the use of external vocabularies and ontologies in order to achieve semantic interoperability. This work has some similarities to ours, since we both deal with the interoperability of heterogeneous systems by using healthcare standards. However, this work does not use software agents, which could be provided with a flexible interaction mechanism for allowing them to cooperate by carrying the PHR content.

K4CARE[4] [55–57] is a large European Union (EU) project whose main goal is to design, implement and validate a new Home Care Model (HCM) for helping manage the home care assistance required by the increasing EU senior population. Two specific goals of this project are the development of an Electronic Home Care Record (EHCR) with a structure and communication based on HL7 standards, and a multi-agent platform for allowing the actors to interact through the HCM and to access the EHCR. A translator agent enables the communication between this platform and external HISs employing different standards, since a special translation schema is provided for this particular communication. This work has several similarities with ours, since both use MAS for controlling clinical tasks and healthcare standards for message exchanging, but it focuses on the specific Home Care domain while our work targets the healthcare domain in general. Furthermore, our Agent-based CCCMSys employs *open*EHR dual model standards, and uses an *open*EHR Gateway for enabling communication with external HISs.

3. Development methodology for e-health applications

Fig. 5 gives an overview of our proposed development methodology represented according to the Structured Analysis and

[4] http://www.k4care.net/.

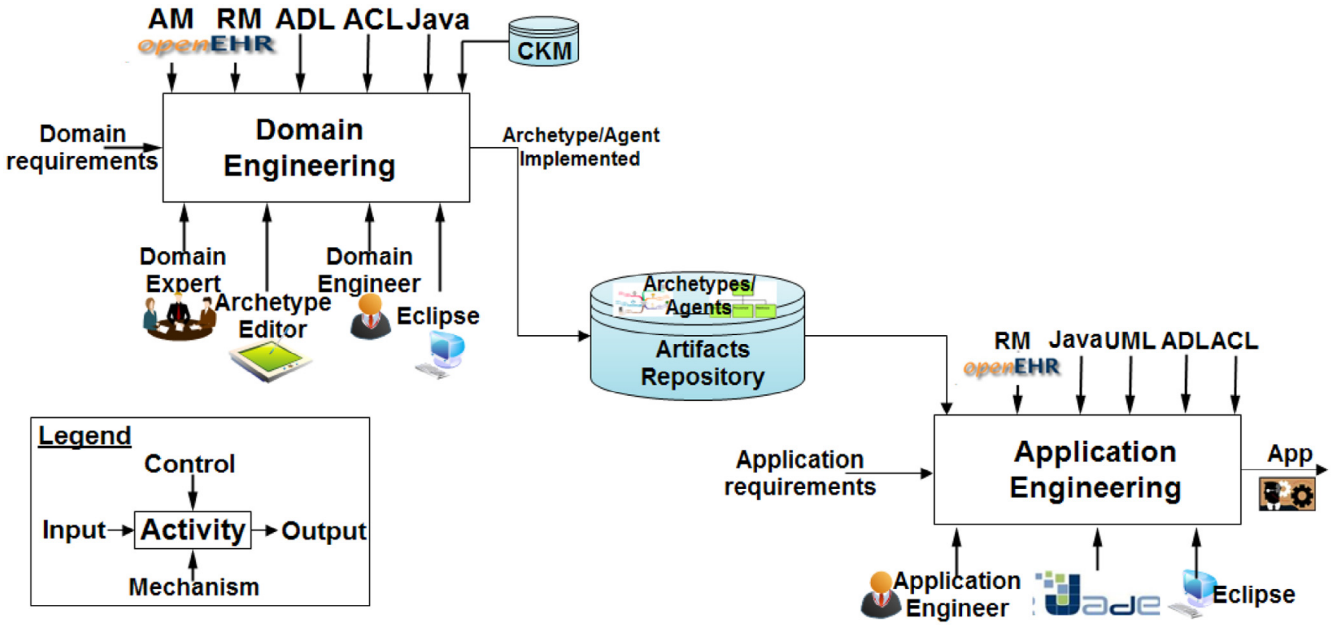

Fig. 5 – Methodology overview [9].

Design Technique (SADT) [58]. In Fig. 5 the activity arrows have the following semantics: *Input* represents entries to be transformed by the activity; *Output* represents results produced by the activity; *Control* represents means that influence activity execution but are not consumed (i.e., languages, standards); and *Mechanism* represents means for executing the activity (i.e., machine, computer, person or tool) [9].

Fig. 5 shows that our methodology comprises two phases [59,60]:

(1) *Domain Engineering (DE)*, where healthcare domain models are defined using *open*EHR archetypes, and agents are modeled in terms of *behaviors*, which are specified in terms of action operations that represent the tasks an agent is able to perform [37]. These models are used to support application modeling, allowing code to be generated that handles the agents in combination with the *open*EHR archetypes. The reusable domain artifacts generated in DE phase are the specified archetypes and modeled agents. These artifacts are stored in an Artifacts Repository to be reused in the next phase; and
(2) *Application Engineering* (AE), where applications are developed according to the application requirements specification. In this phase, new applications are derived from the reusable artifacts produced in the DE phase. The application artifacts include all models designed according to the application requirements, and the source code generated by the *Application Engineer* [61].

The advantage of this separation is that two concerns are kept separated [62], namely, (1) to build reusable and flexible domain artifacts and (2) to build specific applications by reusing available artifacts, adding in this way value to the legacy systems of the related domain.

3.1. *Domain engineering (DE)*

The DE phase involves the *Domain Specification* (DS), *Domain Design* (DD), and *Domain Implementation* (DI) activities, as shown in Fig. 6.

3.1.1. *Domain specification*

In the DS activity, the healthcare domain requirements are elicited, specified, analyzed and represented in models that express knowledge about this domain. Examples of healthcare domain requirements are requesting laboratory results, making an appointment and checking the availability of resources. The DS main actors are the Domain Expert and the Domain Engineer. *Domain Experts* are the stakeholders who have adequate clinical knowledge to ensure that the developed archetype models are aligned with the end-users' requirements.

For modeling clinical concepts, all items to be represented should be known. Any stakeholder working in the healthcare environment can gather the initial clinical information used in clinical practice. In order to support the premise that "an archetype is defined as the maximal data set for generic clinical concept" [63], item categories and their value ranges have to be identified from existing sources, such as datasets, healthcare information systems, electronic-based and paper-based publications and healthcare standards. These categories have to be combined into significant clinical concepts.

In this activity, the *Domain Expert* analyzes the previous documentation system in order to identify all the relevant items, which are represented as concepts. The goal of this activity is to define a hierarchical structure that classifies these items depending on their clinical meaning. From this hierarchical structure, the concepts are examined to identify overlapping items and structures. The *Domain Expert* retrieves the relevant archetypes from the CKM repository and specifies new

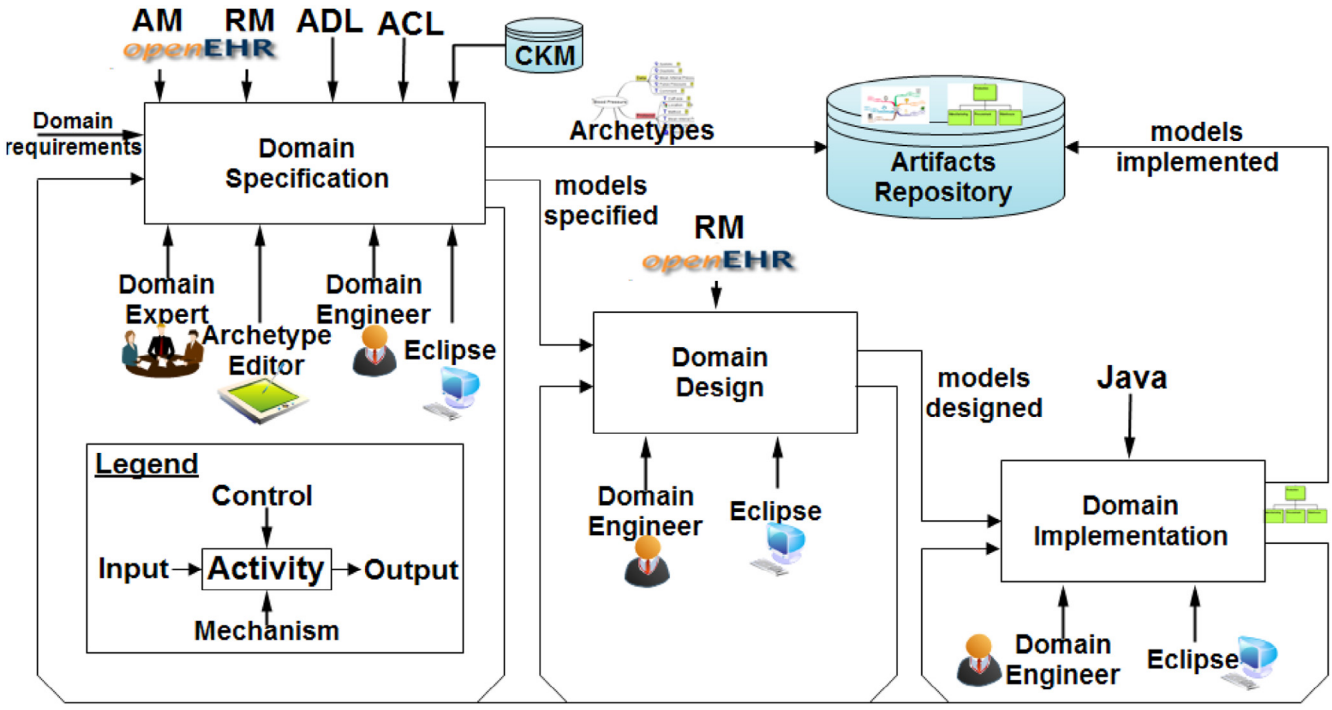

Fig. 6 – Domain engineering phase.

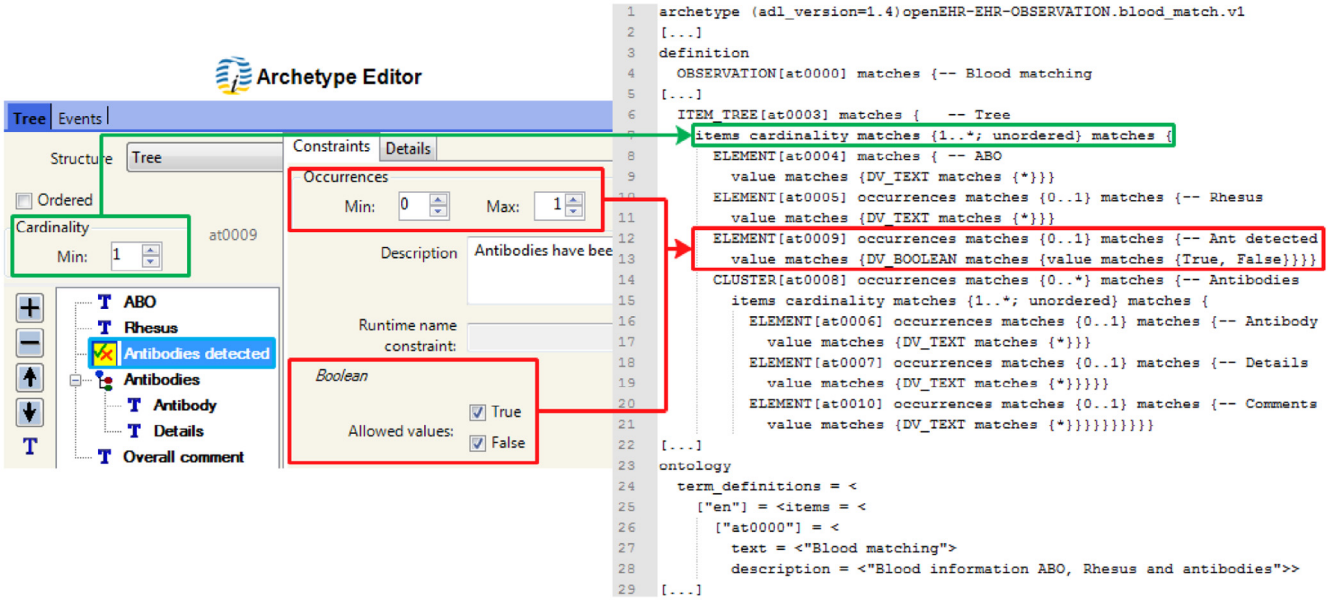

Fig. 7 – *Blood Match* archetype.

archetypes using the Archetype Editor,[5] guided by the *open*EHR AM specifications in ADL.

The outputs of the DS activity are the specified archetypes, which conform to the *open*EHR reference and archetype model and represent existing standard specifications and technical attributes, such as data constraints, occurrences and cardinalities. Fig. 7 shows the specified cardinality of the *Blood Match* archetype designed by the *Domain Expert*. Fig. 7 also shows the constraints expressed in this reusable archetype, such as its occurrence (0..1), type (Boolean) and values (true or false).

The *Domain Engineer* analyzes the current system in order to understand its goals, limitations and requirements, and then describes the activities and use cases of the system, by

[5] http://oceaninformatics.com/.

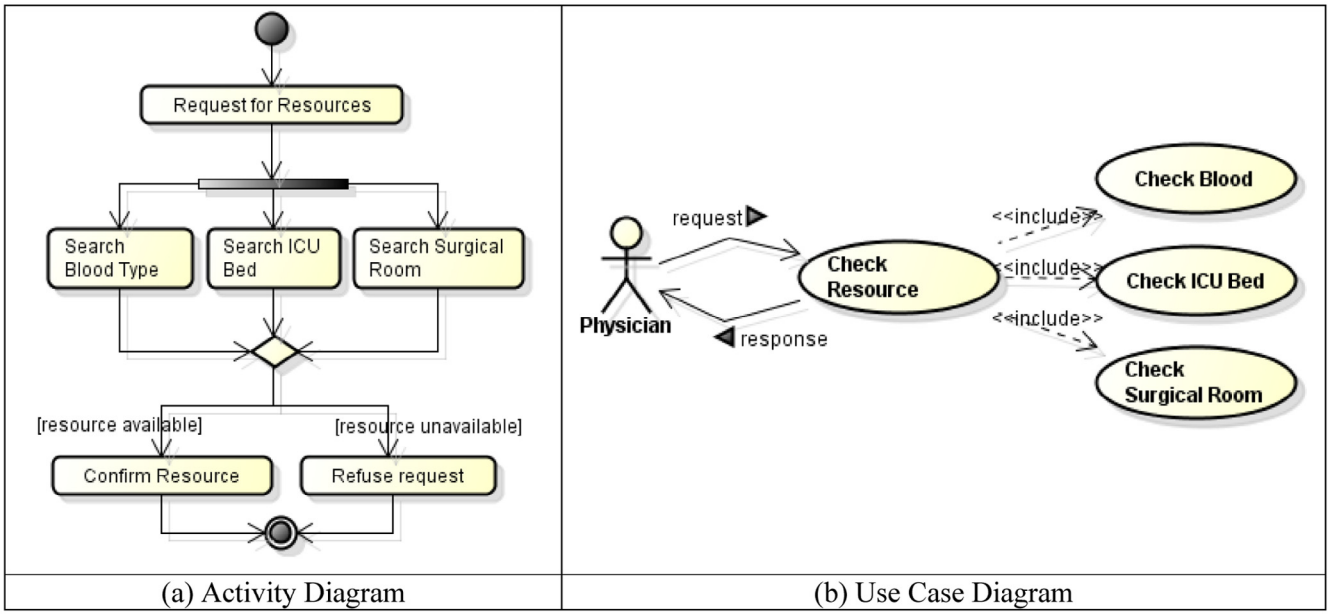

Fig. 8 – Domain specification model—request for resources.

identifying the agent types. The *Domain Engineer* can model these agents with the Eclipse Agent Modeling Project (AMP)[6] plugin. The outputs of the DS activity are the activity diagrams for the agents' behavior specification and use case diagrams that represent the actor's interaction with the environment. Fig. 8(a) shows an excerpt of the activity diagram that describes the sequence and conditions for coordinating agent behaviors for requesting resources, and Fig. 8(b) shows an excerpt of the diagram that provides a high-level view of the use case in which the physician actor requests for resources in a healthcare environment.

3.1.2. *Domain design*

In the DD activity, the deliverables of the DS activity are used for defining the domain terminology, in which the *Domain Engineer* focuses on concepts, actions, predicates and their relationships in order to design the ontologies for the different agent components. We adopt the following conventions to design the domain terminology: "concepts" are denoted with substantives, such as *Patient*, *Physician*, *Blood type* and *Resource*; "actions" are denoted with verbal phrases, such as *MakeAppointment*, *RequestResource*, *RequestForMeeting*, *BloodMatch* and *RequestMedicalReport*; and "predicates" are denoted with Boolean assertions that are evaluated as true or false, such as *hasBloodType*, *hasICUBed*, and *hasOperatingRoom*. Fig. 9 shows the ontology class diagram, which describes the structure of the concept *BloodType*, action *BloodMatch* and predicate *HasBloodType*, which are used in the content of the messages exchanged by agents in this domain. In the DD activity, the agent types and their roles are identified, especially the agent behavior. For example, an agent of type *PhysicianAgent* can play the roles of surgeon or anesthesiologist. The agent interactions are defined to determine how, what and when the various agents communicate.

[6] http://eclipse.org/amp/.

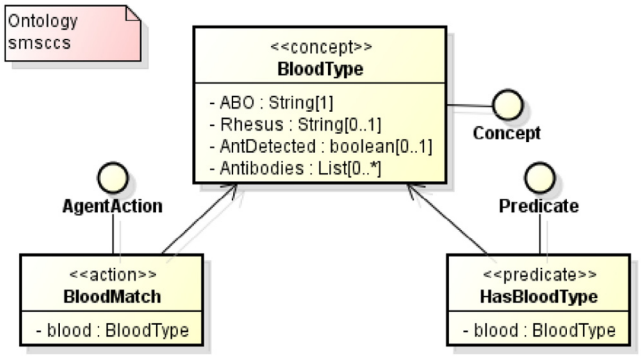

Fig. 9 – Class diagram representing a simple ontology for *Requesting Blood Type*.

The DD outputs are the sequence diagrams that model the interactions between agents and the class diagrams that represent the internal structure, behavior and relationships among the agent types.

Fig. 10 shows the sequence diagram for requesting resources that represents the negotiation between the agents involved.

For instance, a message is annotated with the *cfp* (call for proposal) FIPA-performative indicating that it should be multicast from an Initiator (*ResourceAgent*) to *n* Participants (*IntensiveUnitAgent*, *SurgicalCenterAgent* and *BloodBankAgent*). In this example, the *BloodBankAgent* participant sends a refusal message to this call for proposal, while *SurgicalCenterAgent* and *IntensiveUnitAgent* send proposals. A *proposal* can be accepted, which has been the case for the proposal from *IntensiveUnitAgent*, or rejected, which has been the case for the proposal from *SurgicalCenterAgent*.

Fig. 11 shows the class diagram of agents and behaviors involved in requesting resources. *PatientAgent* is responsible for

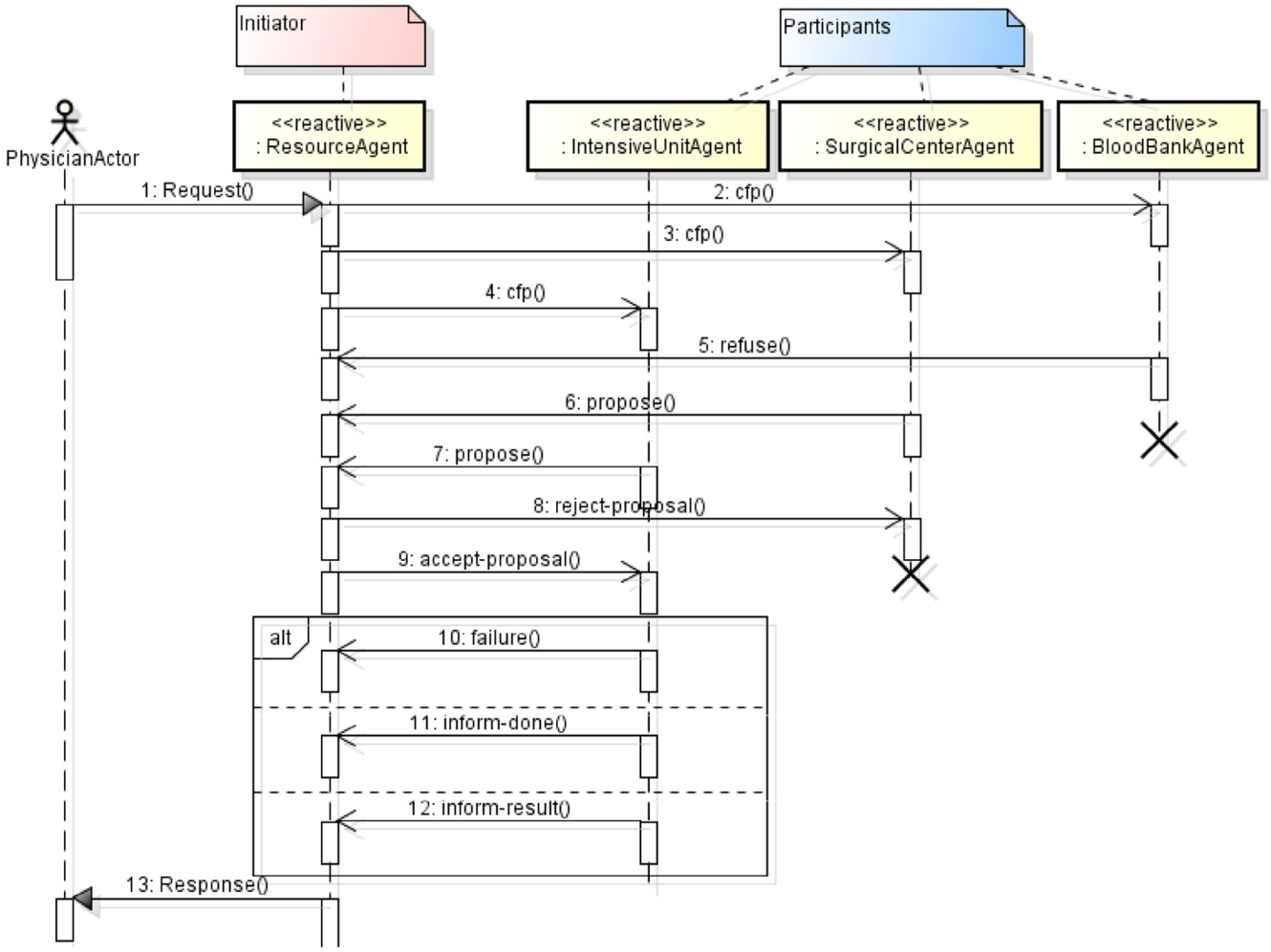

Fig. 10 – Sequence diagram for *Requesting Resources*.

the continuous monitoring of the evolution of a patient, and can send and receive messages to and from a *PhysicianAgent*. *PhysicianAgent* is a mobile agent endowed with intentionality that helps the medical staff monitor the tasks performed during a workday, and obtain information about patients and the availability of resources without requiring the intervention of healthcare professionals. For example, information related to bedridden patients is obtained via

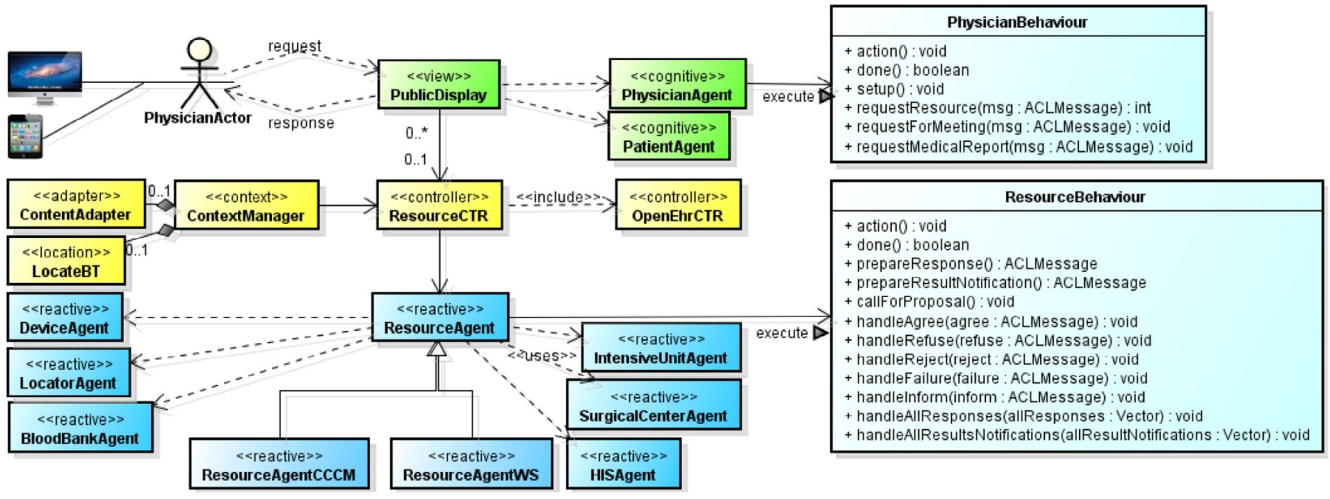

Fig. 11 – Class diagram—agents and behaviors.

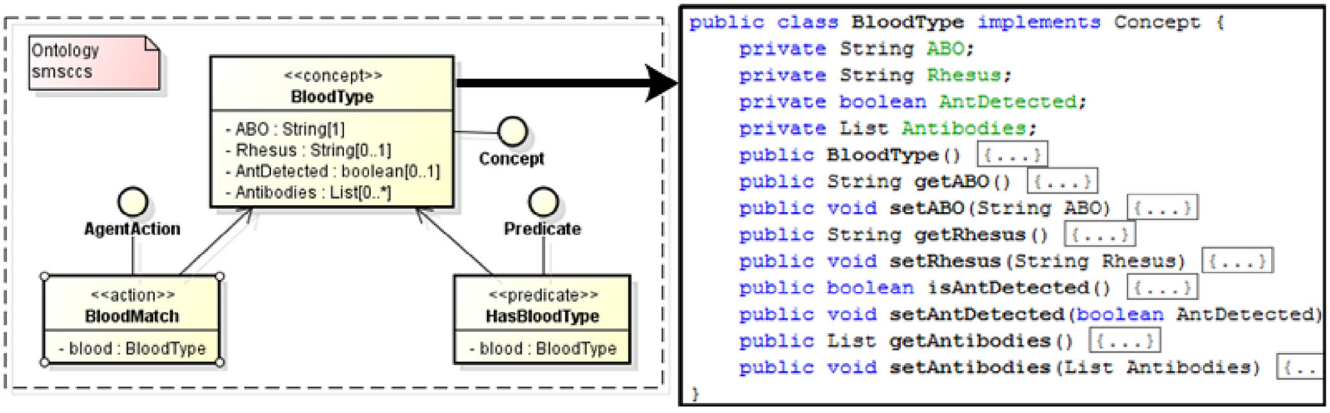

Fig. 12 – Excerpt of Java code for implementing the ontology *Requesting Blood Type*.

ResourceAgent. *ResourceAgent* is a static agent that runs on a remote server and is responsible for mediating the access to resources related to *HISAgent*. *ResourceCTR* provides interfaces to *PatientAgent* and *PhysicianAgent*, which are the main agents that communicate with *ResourceAgent* for obtaining clinical information. *PhysycianBehavior* and *ResourceBehavior* represent the behaviors executed by the agents *PhysicianAgent* and *ResourceAgent*, respectively. These behaviors include checking of received ACL message's performative (i.e., *inform*, *request* and *cfp*), requesting for information (i.e., *requestResource*, *requestForMeeting* and *requestMedicalReport*) and sending reply ACL messages to the requestor agents (i.e., *prepareResponse* and *prepareResultNotification*).

3.1.3. *Domain implementation*

In the DI activity, deliverables of the DD activity are used to identify and create the software agents that are assigned to the actors, so that their behaviors are defined according to their roles in the domain. The *Domain Engineer* uses the Eclipse IDE for implementing the software agents in Java, which are the DI outputs. Fig. 12 shows the Java code that can be produced in this activity, such as the *BloodType* (concept), *BloodMatch* (action) and *HasBloodType* (predicate) classes, which are implemented according to the ontology class diagram for *Requesting Blood Type* designed in the DD activity. At the end of the DI activity, tests are performed to determine whether it is necessary to iterate on the previous DE activities, otherwise the DE phase can be concluded.

The main goal of the *Domain Engineering* phase is to extract the knowledge necessary for instantiating a particular agent in *Application Engineering* phase from the generic models. This knowledge corresponds to the archetypes, agent behaviors, interactions with other agents and some other knowledge specific to a particular agent. We defined a set of ontologies that allow actors of the domain to communicate with each other in a language they understand.

3.2. *Application engineering (AE)*

Application Engineering is a process in which a specific application is developed by reusing the artifacts obtained during the *Domain Engineering* phase. The *Application Engineer* analyzes the legacy systems functions, procedures and terminologies, and additional user requirements for the target application, selecting on appropriate domain model and completing the application development by reusing software components in a bottom-up manner. The AE phase consists of the *Application Analysis* (AA), *Application Design* (AD), and *Application Implementation* (AI) activities, as shown in Fig. 13.

3.2.1. *Application analysis*

In the AA activity, the *Application Engineer* analyzes the legacy applications and data, which have been possibly implemented using languages, platforms and techniques older than the technologies considered in this work. Existing systems are analyzed so that potentially reusable components can be identified, such as source code, software documentation and end-users' requirements. Afterwards, the *Application Engineer* analyzes the additional requirements of the target application, and works with the end-users in order to define the functionality that most resembles or accommodates what the end-users really need. To identify the stakeholders and the application business processes, the *Application Engineer* uses UML use case diagrams, which are the AA outputs for documenting the functional requirements and for describing application behaviors.

Fig. 14 shows the use case diagram that provides a high-level view of the relationships between actors (*Physician* and *Patient*) and use cases, such as *Laboratory Test*, *ECG Test* and *X-Ray Test*.

3.2.2. *Application design*

The AD activity aims at identifying the potential reusable components of the domain problem, including the entities and relationships between the agents. To achieve this goal, the specifications are refined to model the application by considering hardware and software platforms, such as, e.g., Java EE and JADE. Using the deliverables of the AA activity, the *Application Engineer* models concepts as instances of the models built in the DE phase, and selects the classes and behaviors that are relevant to the application domain. Fig. 15(a) shows the GUI designed by the *Application Engineer* based on the constraints imposed by the *Blood Match* archetype. *Antibodies Detected*

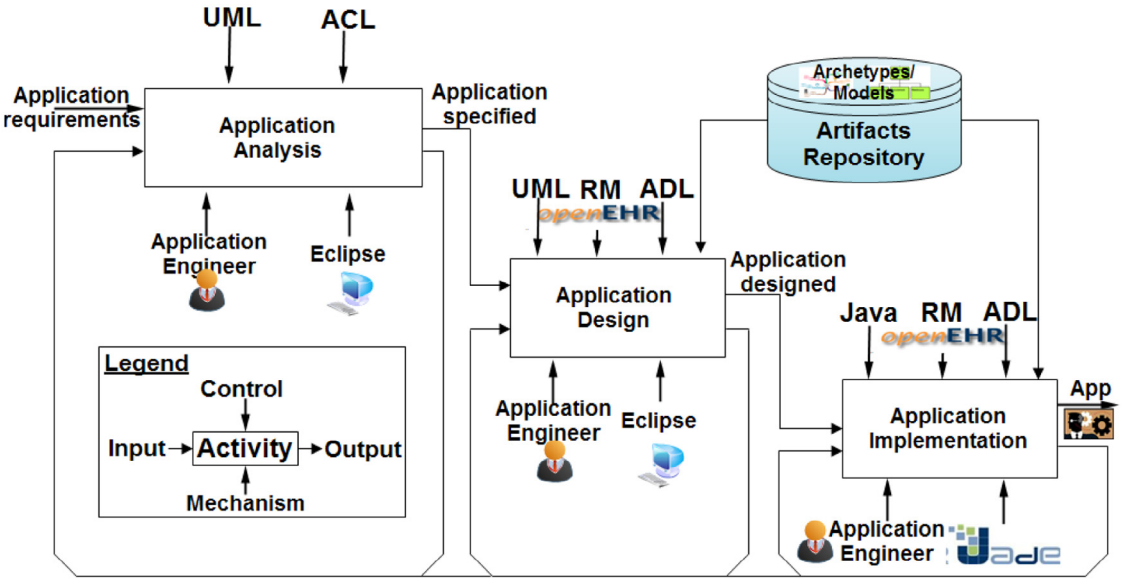

Fig. 13 – Application engineering phase.

(DV_BOOLEAN type) is modeled as option box (True or False) and *ABO* (DV_TEXT type) is modeled as text field (String). Fig. 15(b) shows the constraints expressed in the reusable archetype, such as constraints on type, values, cardinality, existence and occurrence. *Application Engineer* refines the class diagrams that describe the application structure obtained in DE phase, showing the relationship between the agent classes and their behaviors.

3.2.3. *Application implementation*

In the AI activity, the application is implemented. This includes the communication interface designed in the AD activity,

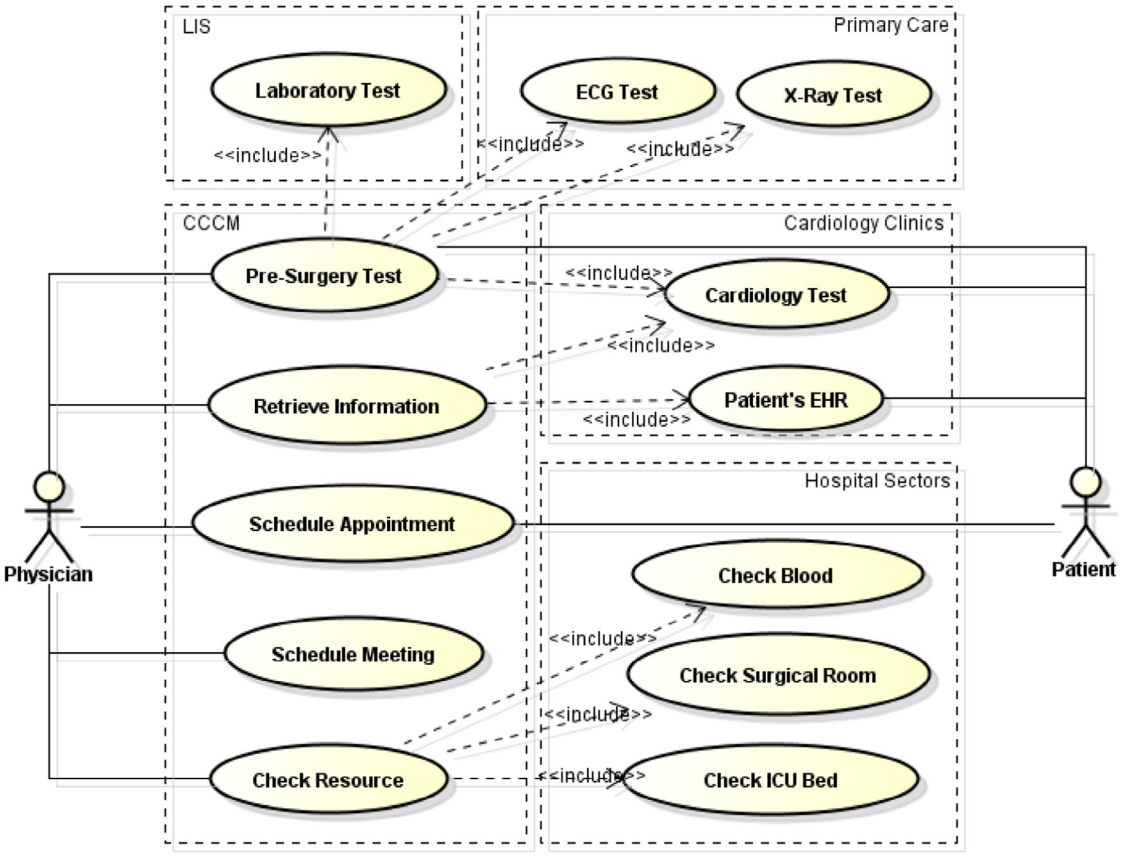

Fig. 14 – Use case diagrams.

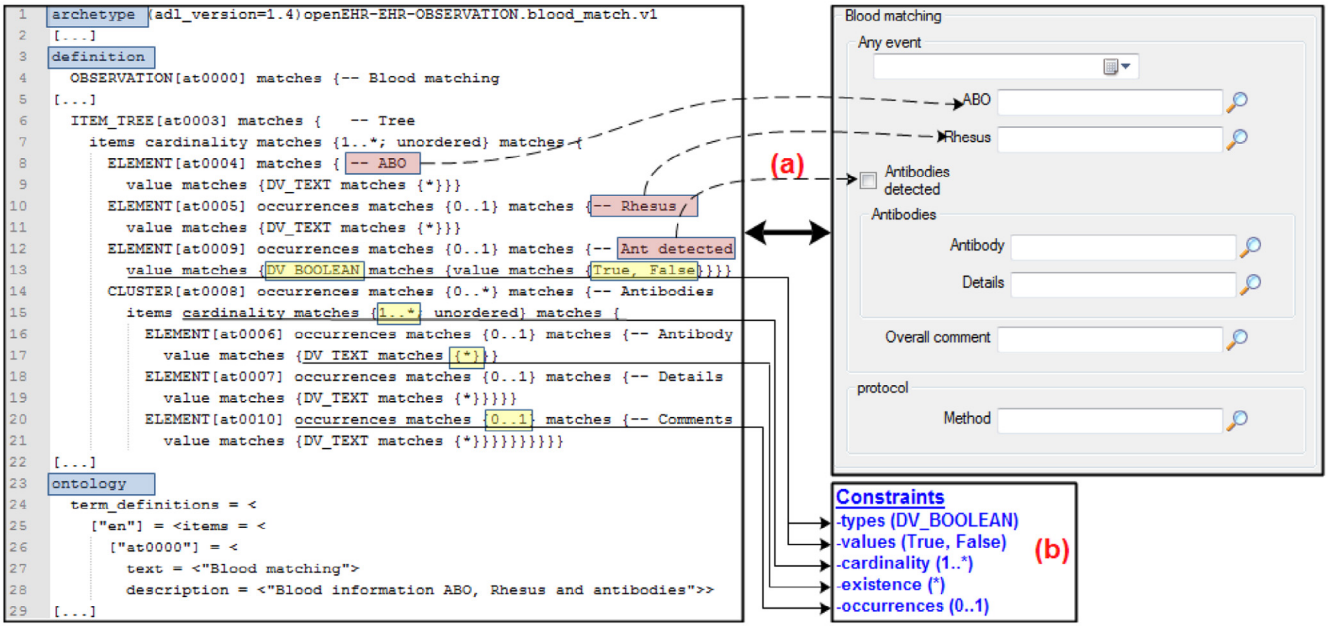

Fig. 15 – Interface designed based on *Blood Match* archetype.

which is built by integrating archetypes to *open*EHR messages. The application is implemented according to the models produced before, and since the application is implemented in Java the Eclipse IDE is the main tool used in this activity. Moreover, other components are developed for application integration, such as the user integration interfaces and data persistence services implemented according to data access object (DAO) and data transfer object (DTO) patterns [64]. These components make use of a standard SQL relational database, incorporating a Java-based Hibernate layer in order to associate table data to class instances. This offers efficient access to concrete clinical information, such as the results of specific laboratory tests and electrocardiograms.

We applied JADE [36] to implement the agents, since JADE enables the development of a multi-agent system by providing a standard implementation of the FIPA-ACL communication language. The agent and behavior classes that were refined and designed on AD activity are implemented in this activity, and the agents communicate through JADE. Fig. 16 shows the *ResourceAgent* behavior implemented in Java according to the *Requesting Resource* use case. The *BloodType* concept (Line 63–66) and the *HasBloodType* predicate (Line 67–69) are instantiated, and the request *ACLMessage* (Line 71–77) is instantiated to query the *BloodBankAgent* for the availability of blood of the requested blood type (Line 80–81).

At the end of the AI activity, tests are performed to determine whether it is necessary to iterate on the previous AE activities, otherwise the AE phase can be concluded.

4. Case study and mode of availability of software

Using our methodology, we designed a system that allowed us to shift from the legacy CCCMSys that relies on informal offline communication between actors to an agent-based system. The Heart Team members are highly qualified professionals characterized by their experience and skills; however, these team members still have the limitations inherent to human beings. Our proposed methodology aims to design software agents that work on behalf of Heart Team members, delegating daily activities performed by human actors to software agents.

4.1. *Design*

In our system, a software agent is assigned to each actor working in the healthcare environment in order to perform the assigned tasks in a cooperative multi-agent environment. These software agents are placed on an additional layer (Layer 3) with respect to the legacy CCCMSys, as shown in Fig. 17.

The *open*EHR Gateway is the interface that enables the communication between the Agent-based CCCMSys and several HISs by means of the *open*EHR specification. The Agent-based CCCMSys has the following agent types: physician, patient, nurse, assistant, resource, schedule, blood bank, surgical center, ICU, device and locator. These agents cooperate with each other in a multi-agent environment in order to automate the information exchange between the stakeholders to which they have been assigned. To ensure the interoperability of the system with other HISs, we designed an agent with the specific ability to exchange messages containing EHR extracts, which are represented according to the *open*EHR archetype specifications. This agent envelopes messages using ACL that contain serialized objects, and sends it to the requester.

Fig. 18 shows the message exchange when *ResourceAgent* requests blood of some type to *BloodBankAgent* according to the specified ontology. In Fig. 18, (1) shows the ontology classes (*BloodType*, *BloodMatch* and *HasBloodType*) according to the constraints imposed by the reusable archetype (*Blood Match*), such as constraint on type, values, cardinality, existence and occurrence; (2) shows the instances of *BloodMatch* (AgentAction)

```
47  public class ResourceAgent extends Agent {
48
49      class RequestBehaviour extends SequentialBehaviour {
50          Behaviour queryBehaviour = null;
51          Behaviour requestBehaviour = null;
52          // Constructor
53          public RequestBehaviour(Agent myAgent) {
54              super(myAgent);
55          }
56          // This is executed at the beginning of the behaviour
58          @Override
58          @SuppressWarnings("CallToThreadDumpStack")
59          public void onStart() {
60              try {
61                  Ontology o = myAgent.getContentManager().lookupOntology(BloodOntology.NAME);
62                  BufferedReader buff = new BufferedReader(new InputStreamReader(System.in));
63                  //CONCEPT
64                  BloodType blood = new BloodType();
65                  blood.setABO(buff.readLine());
66                  blood.setRhesus(buff.readLine());
67                  //PREDICATE
68                  HasBloodType hasBT = new HasBloodType();
69                  hasBT.setBlood(blood);
70                  //ACL message to query the BloodBank agent if the above fact is true or false
71                  ACLMessage queryMsg = new ACLMessage(ACLMessage.REQUEST);
72                  queryMsg.addReceiver(((ResourceAgent) myAgent).bloodBankAgent);
73                  queryMsg.setLanguage(FIPANames.ContentLanguage.FIPA_SL0);
74                  queryMsg.setOntology(EmploymentOntology.NAME);
75                  try {
76                      myAgent.getContentManager().fillContent(queryMsg, hasBT);
77                  } catch (Exception e) {e.printStackTrace();}
78                  // Create and add a behaviour to query the bloodbank agent whether
79                  // blood already exists in bloodbank following a FIPAQeury protocol
80                  queryBehaviour = new ResourceAgent.CheckAlreadyBloodBehaviour(myAgent, queryMsg);
81                  addSubBehaviour(queryBehaviour);
82              } catch (IOException ioe) {System.err.println("I/O error: " + ioe.getMessage());}
83          }
```

Fig. 16 – *ResourceAgent* behavior—request for resources.

and *HasBloodType* (Predicate) types, which are exchanged when *ResourceAgent* (3) requests a blood type from the *BloodBankAgent*, which (4) informs if the requested blood of the requested type is available or not. The *Application Engineer* refines the class diagrams that describe the application structure obtained in the DE phase, showing the relationship between the agent classes and their behaviors.

4.2. *Usage scenario*

The experiments with the Agent-based CCCMSys were conducted in a distributed environment involving three cardiology clinics (ICM, CRTB, and Prevencor) and the Cardiology Department (CCCM) of the Santa Casa Hospital, all located in Marília (São Paulo, Brazil). We assigned fictitious names to the actors in this paper to make sure the participants remain anonymous. The scenario deals with a meeting to prepare the patient Mr. Silva for cardiac surgery, with the following staff: Dr. Call (cardiac surgeon) and Dr. Day (assisting surgeon), both from CCCM; Dr. John (anesthesiologist) from Santa Casa Hospital; Dr. Marden and Dr. Peter (physicians) from the Department of Hemodynamics of Marília; Mrs. Elienne (nurse); and Mrs. Aline (perfusionist). The interactions between the software agents to plan this cardiac surgery are shown in Fig. 19.

The following notifications are exchanged in these interactions:

(1) *AssistantAgent* asks the required resources to perform the cardiac surgery to *ResourceAgentCCCM*;
(2) *ResourceAgentCCCM* analyzes the conditions for performing the cardiac surgery, and starts the negotiation between *BloodBankAgent*, *IntensiveUnitAgent* and *SurgicalCenterAgent* to confirm the availability of the required blood type at the Blood Bank, an Intensive Care Unit bed and a Surgical Center room, respectively;
(3) *BloodBankAgent*, *IntensiveUnitAgent* and *SurgicalCenterAgent* respond to the *ResourceAgentCCCM* about the availability of the requested resources;
(4) Once all these resources are available, *ResourceAgentCCCM* sends a message to the mobile device of each staff member in order to set a date for the meeting in which the cardiac surgery is expected to be prepared. The meeting room is context-aware, and the staff members are traced by the *DeviceAgent* and *LocatorAgent*;

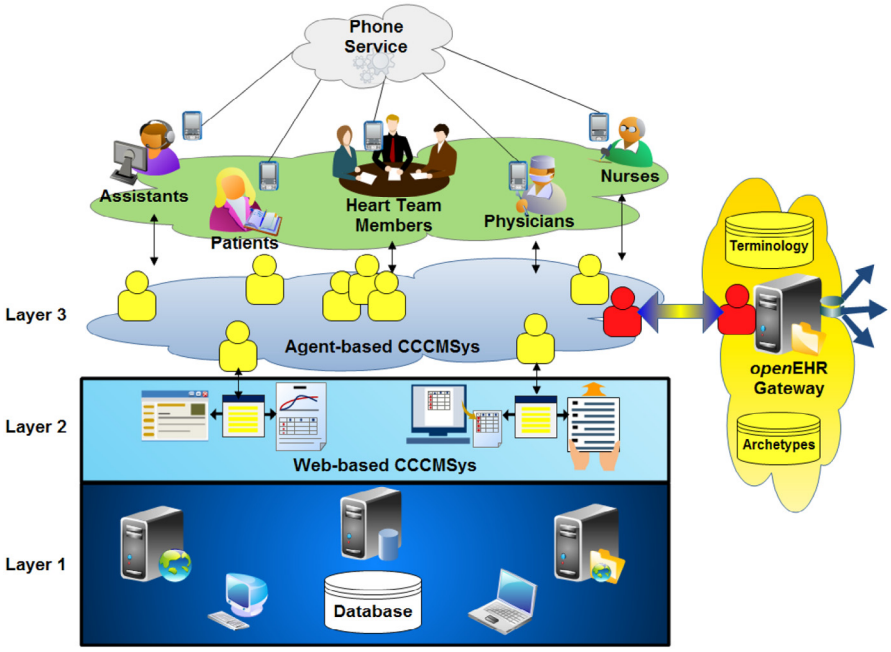

Fig. 17 – Agent-based CCCMSys.

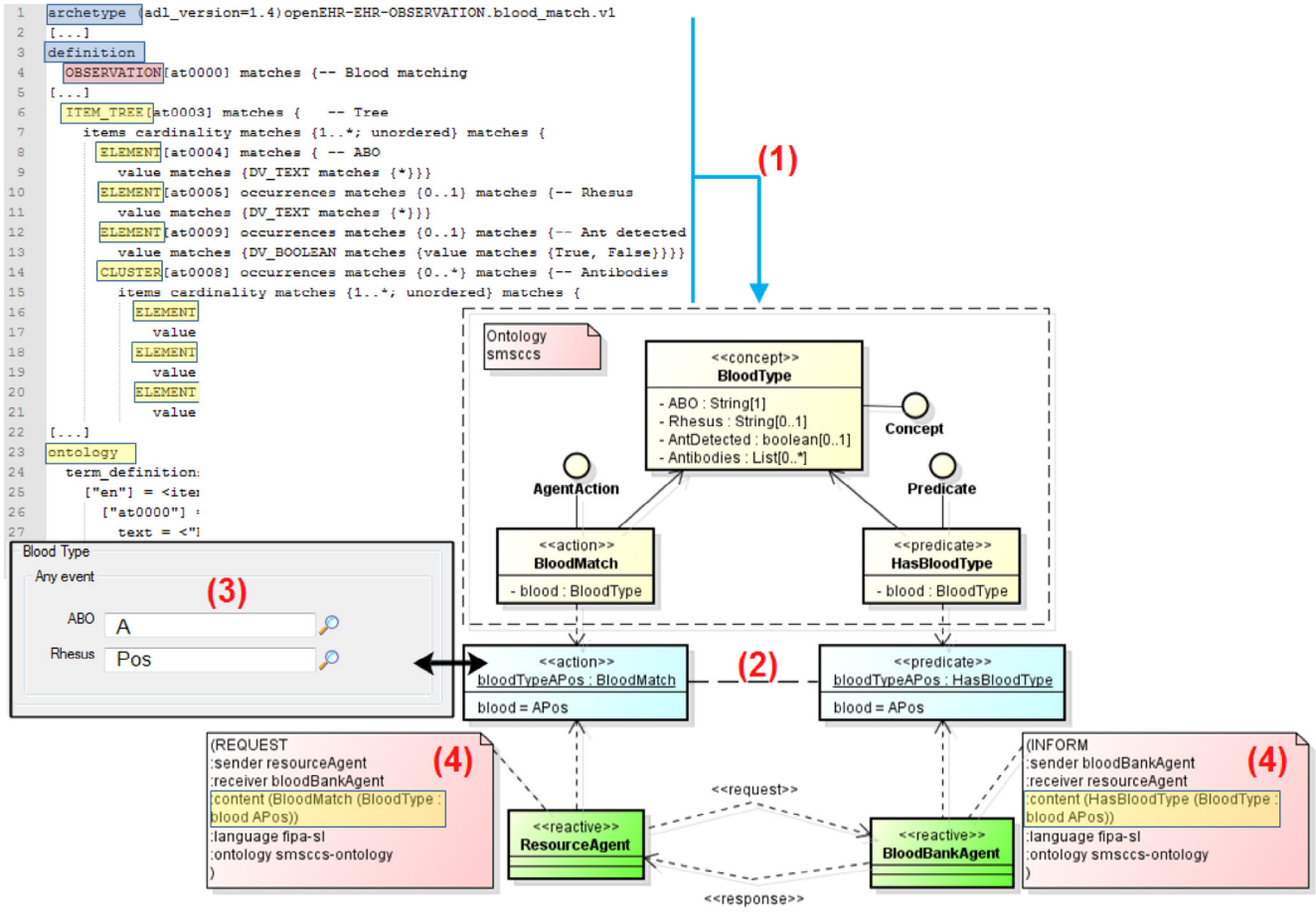

Fig. 18 – Ontology-based message content—*Request for Resources*.

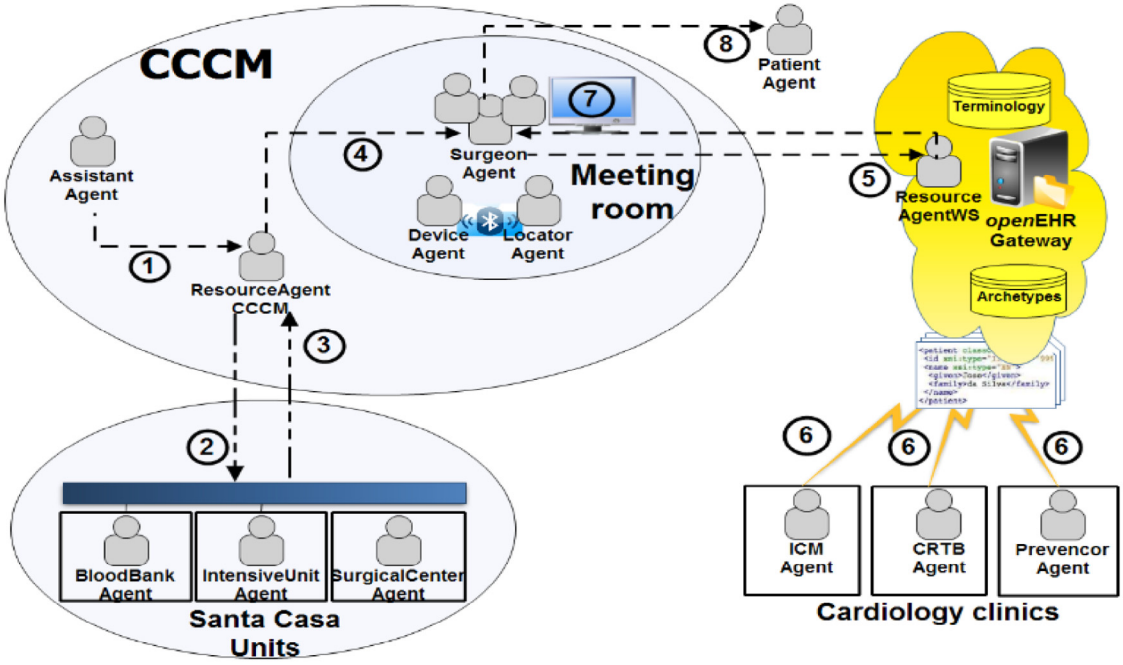

Fig. 19 – Software agents' interactions.

(5) *ResourceAgentCCCM* requests to the *ResourceAgentWS* all the information related to the patient's EHR;
(6) *ResourceAgentWS* receives the messages from the cardiology clinics and serializes the EHR extract containing the patient's information based on the constraints imposed by the *open*EHR archetype;
(7) *ResourceAgentWS* envelops the serialized object in a message, and sends it to *ResourceAgentCCCM* for displaying the patient's information on a big screen to all staff members;
(8) After this meeting has taken place, the *SurgeonAgent* notifies the *PatientAgent* about the surgery time slot.

Fig. 20 shows the GUI of the application, in which *AssistantAgent* Anne requests the required resources to perform a cardiac surgery on patient *Mr. Silva*: the ICUBed is flagged as *Resource Available*, and the Blood Bank and Surgical Center resources are flagged as *Waiting Response* by the *BloodBankAgent* and *SurgicalCenterAgent*, respectively. In Fig. 20, the *Message Content* panel displays the information necessary to perform the cardiac surgery on the patient. Once these resources are available, the staff members receive a message on their mobile devices in order to set a date for the meeting. In Fig. 20, the *Meeting Room* display is disabled, but it becomes enabled during the meeting to discuss the patient cardiac surgery.

4.3. *Evaluation*

In Pervasive Computing, evaluation of new technology is often based on a Technical Laboratory Proof-of-Concept (PoC). In Healthcare, this kind of evaluation is usually not enough for new health technologies, and a methodology named Clinical PoC [65] was proposed for striking a balance between a Technical Laboratory PoC and a full-scale Clinical Trials. A Clinical PoC is characterized as follows:

- It is a controlled experiment;
- It allows the viability of the working prototypes to be assessed in daily use;
- It works inside a real clinical setting; and
- It is relatively short term (typically around 6 months).

To evaluate the acceptance of the Agent-based CCCMSys in a realistic healthcare environment, two groups of users were selected for the proposed scenario: 57 healthcare professionals, including physicians, nurses and medical students; and 122 patients. The healthcare professionals were grouped together because they work as a team in this scenario.

First, we applied a Technical Laboratory PoC during 3 months in the proposed scenario to enable the use of the Agent-based CCCMSys in the daily activities of the selected users. After this period, we applied a Clinical PoC during 6 months to collect evidence from this scenario, employing the Technology Acceptance Model (TAM) [66], since it is generally accepted as an appropriate model to explain technology acceptance in healthcare [67]. TAM assumes that the *Perceived Usefulness (PU)* and *Perceived Ease of Use (PEU)* can predict the use and *Intentions to Use (IU)* of a particular technology.

The users have used the Agent-based CCCMSys in their daily activities from 1st January to 30th July 2012, and after that they have been asked to fill in the structured questionnaire shown in Table 1. This questionnaire has been designed based on TAM. Respondents were asked to give their opinion about the statements in Table 1 on a five-point Likert scale [68], ranging from 1 (strongly disagree) to 5 (strongly agree). To control for bias,

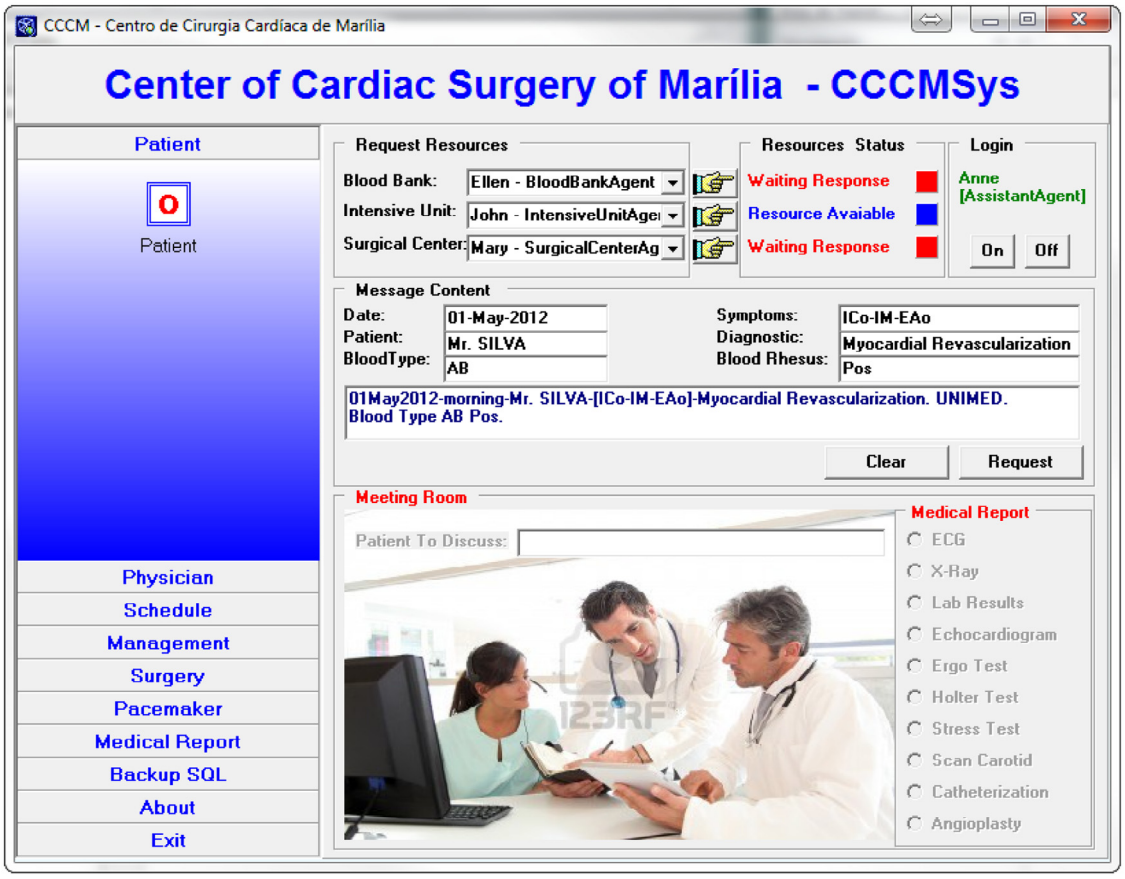

Fig. 20 – GUI of the application for requesting resources.

the questionnaires randomly intermixed items across constructs (PU, PEU, IU), and we conducted a group pre-test to ensure that the scales were appropriate [9,69,70]. About 80% and 73% of the distributed questionnaires were duly completed by the healthcare professionals and patients, respectively.

Our analysis was divided in two parts: (1) the validity and reliability of the measurement model were tested with Cronbach's Alpha [71]; and (2) to examine the research model and the hypotheses, the data were analyzed using Structural Equation Modeling (SEM) [72], which is a statistical method to analyze relationships among variables. Table 2 summarizes the results of this analysis based on the answers to the questionnaire. The constructs have Cronbach's Alpha values close to the limit of 0.700 [71], which is considered acceptable.

Fig. 21 shows the research model employed during this evaluation. The arrows are labeled with the hypotheses and the path coefficients, where the latter measures the relative strength and indicates the causal relationships among the

Table 1 – Structured questionnaire.

Perceived ease of use (PEU)	PEU1	My interaction with the system is clear and understandable.
	PEU2	Interacting with the system does not require a lot of mental effort.
	PEU3	I find the system easy to use.
	PEU4	I find it easy to make the system do what I want it to do.
Perceived usefulness (PU)	PU1	Using the system improves my performance in my job.
	PU2	Using the system increases my productivity in my job.
	PU3	Using the system enhances my effectiveness in my job.
	PU4	I find the system useful in my job.
	PU5	Using the system it becomes easier to perform my tasks.
Intention to use (IU)	IU1	Assuming I have access to the system, I intend to use it.
	IU2	Given that I have access to the system, I predict I will use it.
	IU3	Using the system in my job is a good idea.

Table 2 – Statistics of constructs.

Participants	Constructs	Number of Items	Mean
Healthcare professionals	Perceived ease of use	4	4.03
	Perceived usefulness	5	4.35
	Intention to use	3	4.11
Patients	Perceived ease of use	4	4.12
	Perceived usefulness	5	4.50
	Intention to use	3	4.15

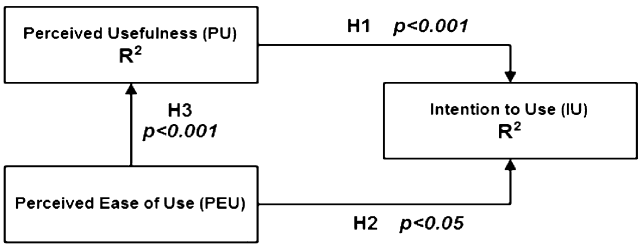

Fig. 21 – Research mode.

variables. R^2 represents the percentage of total variance of the independent variables, and indicates the predictability of the research model.

The following hypotheses were formulated and analyzed for the proposed scenario:

H1. PU positively affects the IU of the system.
- Null hypothesis: H_{1n}: $\mu_{PU} = \mu_{IU}$.
- Alternative hypothesis: H_{1a}: $\mu_{PU} \neq \mu_{IU}$.

H2. PEU positively affects the IU of the system.
- Null hypothesis: H_{2n}: $\mu_{PEU} = \mu_{IU}$.
- Alternative hypothesis: H_{2a}: $\mu_{PEU} \neq \mu_{IU}$.

H3. PEU positively affects PU.
- *Null hypothesis:* H_{3n}: $\mu_{PEU} = \mu_{PU}$.
- Alternative hypothesis: H_{3a}: $\mu_{PEU} \neq \mu_{PU}$.

To verify these hypotheses, we have chosen to apply *Statistical Regression Analysis* [72] to the data collected from the users. We tested H_{3a}, and the regression analysis results were $R^2 = 0.53$ and $p = 0.0005$, which is highly significant because $p < 0.001$ and $\alpha = 0.05$.

According to these results, H_{3n} can be rejected, meaning that PEU positively affects PU, and therefore H_{3a} is strongly confirmed. Since $R^2 = 0.53$, indicating that PEU explains 53% of the variance in PU, the hypotheses H_{1a} and H_{2a} can be confirmed, and we can conclude with a high confidence that PU and PUE positively affect IU.

Therefore, all hypotheses were confirmed at all measurement points in the proposed scenario. Most of the healthcare professionals claimed that the Agent-based CCCMSys was useful in their daily tasks, and it was easy to use. Furthermore, several healthcare professionals and patients mentioned some usability benefits of this system, such as its efficient method for generating notification messages. In general, healthcare professionals confirmed that the new capabilities added to the CCCMSys improved productivity and automated routine tasks in their healthcare environment.

However, our data analysis has some limitations: (1) the questionnaire we applied in our evaluation is not completely free of subjectivity, since each respondent can react to it in a particular way; (2) all healthcare professionals were grouped together and the results were generalized, despite their different kinds of expertise; (3) PU and PEU were the most important factors to explain people's intention of using a given technology; however, other factors may affect this decision, such as the prior experience and their job relevance [9,69,70,73].

5. Conclusion and future plans

This paper presented a methodology that allows healthcare organizations to move from legacy systems that rely on informal offline communication of actors, to an automated agent-based system. This methodology is based on the use of software agents and *open*EHR archetypes, and has been validated with a case study.

In our work, we demonstrated that software agents can support healthcare professionals in their daily practices, since they are able to properly perform communication tasks on behalf of these professionals. Agents can also monitor their environment, since they are able to properly ensure the fulfillment of the contextual requirements. We also observed that these agents are able to take proper decisions about the activities to be performed, and when and how to communicate to perform them.

We have also been able to support the interoperability between heterogeneous HISs in healthcare environments by using *open*EHR archetypes to represent the information contained in clinical EHR extracts. These EHR extracts are enveloped in ACL messages, and which in turn are exchanged in our agent-based system.

The case study demonstrated that we can shift from a legacy system (the original CCCMSys) to an Agent-based system seamlessly, due to the employment of *open*EHR archetypes in order to achieve semantic interoperability. The use and evaluation of the Agent-based CCCMSys by healthcare professionals and patients showed the usefulness and acceptance of this system as perceived by their stakeholders.

The proposed methodology preserves significant investment of many years in the legacy systems and allows developers to extend them by adding new features to these systems, providing proactive assistance to the end-users during their daily tasks and increasing the user mobility with an appropriate support. This methodology had been employed before in the development of communication systems in other realistic healthcare scenarios [42,62,70], and the results obtained so far are highly promising. In principle, the methodology can be also employed in other application domains, since archetypes and software agents can be designed specifically to the target domain, and common components can be encapsulated and specialized, facilitating their reuse.

In further work, the performance of the systems developed according to our methodology can be evaluated, especially their scalability, which is a crucial non-functional requirement for realistic applications and for the simultaneous support of multiple scenarios. We argued that our methodology is reusable, which means that our methodology should be applied in other domains with different properties and challenges, so that the true potential and limitations of the methodology can be analyzed. Privacy and security issues arise when we integrate legacy healthcare systems with the technologies we have used. Healthcare systems are subject to strict regulations, so these issues must be addressed in future work to enable their use in more realistic scenarios and scale. Other issues that must be addressed include ensuring the confidentiality and security of information exchange in a realistic healthcare environment. TAM can be extended or replaced by another

model, to allow the evaluation of the effects of other variables than only perceived usefulness and perceived ease of use (e.g., user experience, job relevance and output quality) on the intention to use the solutions we developed with our methodology.

Acknowledgements

We thank the Brazilian National Council of Technological and Scientific Development (CNPq) for sponsoring our research in the context of the Brazilian National Institute of Science and Technology in Medicine Assisted by Scientific Computing (INCT-MACC). We also thank the Brazilian Federal Agency for Support and Evaluation of Graduate Education (CAPES) for sponsoring the stay of the first author at the University of Twente (Enschede, the Netherlands).

REFERENCES

[1] J.L. Cardoso de Moraes, W. Lopes de Souza, L. Ferreira Pires, A. Francisco do Prado, An architecture for health information exchange in pervasive healthcare environment, in: Springer (Ed.), Enterprise Information Systems – 15h International Conference, ICEIS 2013, Angers, France, July 4–7, 2013, Revised Selected Papers, first ed., Lecture Notes in Business Information Processing 190, Springer International Publishing, 2014, pp. 385–401.

[2] S. Jiang, Y. Xue, A. Giani, R. Bajcsy, Robust medical data delivery for wireless pervasive healthcare, in: Proc. 8th IEEE International Conference on Dependable, Autonomic and Secure Computing, 2009, pp. 802–807.

[3] B. Skinner, M. Rovere, Paying More, Getting Less: Measuring the Sustainability of Government Health Spending in Canada, Fraser Institute, 2009.

[4] J. Bisbal, D. Lawless, Bing Wu, J. Grimson, Legacy information systems: issues and directions, IEEE Softw. 16 (5) (1999) 103–111 http://dx.doi.org/10.1109/52.795108.

[5] M.T. Nguyen, P. Fuhrer, J. Pasquier-Rocha, Enhancing e-health information systems with agent technology, Int. J. Telemed. Appl. (2009) 279091 http://dx.doi.org/10.1155/2009/279091.

[6] B. Jensen, B. Patel, Inheriting the wind—what to look out for when replacing a legacy system, Intech 49 (9) (2002) 26–29.

[7] J.L. Cardoso de Moraes, W. Lopes de Souza, L. Ferreira Pires, A. Francisco do Prado, An architecture for message exchange in pervasive healthcare based on the use of intelligent agents, Jornal Brasileiro de TeleSSaúde 2 (4) (2013) 145–156.

[8] J.L. Cardoso de Moraes, W. Lopes de Souza, L. Ferreira Pires, A. Francisco do Prado, Message generation facilities for interoperability in pervasive healthcare environments, in: Proc. 19th Americas Conference on Information Systems (AMCIS), AIS eLibrary, 2013, p. 12.

[9] J.L. Cardoso de Moraes, Methodological Support to Develop Interoperable Applications for Pervasive Healthcare, EWI, Universiteit of Twente, Enschede, 2014.

[10] T. Beale, S. Heard, Archetype Definition Language, openEHR Foundation, 2007.

[11] M. Wooldridge, An Introduction to MultiAgent Systems, Wiley Publishing, 2009, p. 484.

[12] Fu-Shiung Hsieh, Collaborative workflow management in holonic multi-agent systems, in: Agent and Multi-Agent Systems: Technologies and Applications, Lecture Notes in Computer Science 6682, Springer Berlin Heidelberg, 2011, pp. 383–393.

[13] D.L. Hudson, M.E. Cohen, Intelligent agents in home healthcare, Ann. Telecommun. 65 (9–10) (2010) 593–600 http://dx.doi.org/10.1007/s12243-010-0170-6.

[14] D. Sharma, F. Shadabi, An intelligent multi agent design in healthcare management system, in: Proc. Proceedings of the 2nd KES International Conference on Agent and Multi-Agent Systems: Technologies and Applications, Springer-Verlag, 2008, pp. 674–682.

[15] A.F. Merry, J. Weller, S.J. Mitchell, Teamwork, communication, formula-one racing and the outcomes of cardiac surgery, J. Extra Corpor. Technol. 46 (2014) 7–14.

[16] T. Beale, Archetypes and the EHR, Adv. Health Telemat. Telemed. 96 (2003) 238–244.

[17] o. Foundation, openEHR Architecture Overview. http://www.openehr.org/releases/BASE/latest/docs/architecture_overview/architecture_overview.html, 2015.

[18] T. Beale, S. Heard, openEHR Architecture Overview Release 1.0.2, Ocean Informatics, 2008.

[19] D. Isern, A. Moreno, Computer-based execution of clinical guidelines: a review, Int. J. Med. Inform. 77 (12) (2008) 787–808 http://dx.doi.org/10.1016/j.ijmedinf.2008.05.010.

[20] G.M. Bacelar-Silva, H. César, P. Braga, R. Guimarães, OpenEHR-based pervasive health information system for primary care: first Brazilian experience for public care, in: Proc. Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ACM, 2013, pp. 572–873.

[21] G.M. Bacelar-Silva, R. Chen, R.J. Cruz-Correia, From clinical guideline to openEHR: converting JNC7 into archetypes and template, in: Proc. Sociedade Brasileira de Informática em Saúde (SBIS), XIII Congresso Brasileiro de Informática em Saúde, 2012, p. 6.

[22] M. Marcos, B. Martínez-Salvador, Towards the interoperability of computerised guidelines and electronic health records: an experiment with openEHR archetypes and a chronic heart failure guideline, in: Knowledge Representation for Health-Care, Lecture Notes in Computer Science 6512, Springer Berlin Heidelberg, 2011, pp. 101–113.

[23] N.R. Jennings, Agent-oriented software engineering, in: Multiple Approaches to Intelligent Systems, Proceedings 1611, 1999, pp. 4–10.

[24] F. Zambonelli, N.R. Jennings, A. Omicini, M. Wooldridge, Agent-oriented software engineering for internet applications, in: Coordination of Internet Agents, Springer Berlin Heidelberg, 2001, pp. 326–346.

[25] Y. Labrou, T. Finin, Y. Peng, Agent communication languages: the current landscape, IEEE Intell. Syst. Appl. 14 (2) (1999) 45–52 http://dx.doi.org/10.1109/5254.757631.

[26] F. Bellifemine, A. Poggi, G. Rimassa, Developing multi-agent systems with JADE, in: Intelligent Agents VII Agent Theories Architectures and Languages, Lecture Notes in Computer Science 1986, Springer Berlin Heidelberg, 2001, pp. 89–103.

[27] S. Poslad, Specifying protocols for multi-agent systems interaction, ACM Trans. Autonom. Adapt. Syst. 2 (4) (2007) 15 http://dx.doi.org/10.1145/1293731.1293735.

[28] F. Bellifemine, G. Caire, A. Poggi, G. Rimassa, JADE: a white paper, EXP Search Innovation 3 (3) (2003) 6–19.

[29] R. Bordini, J.F. Hübner, M.J. Wooldridge, Programming Multi-Agent Systems in AgentSpeak Using Jason, J. Wiley, 2007.

[30] R. Evertsz, M. Fletcher, R. Jones, J. Jarvis, J. Brusey, S. Dance, Implementing industrial multi-agent systems using JACK (TM), in: Programming Multi-Agent Systems, vol. 3067, 2003, pp. 18–48.

[31] P. Vrba, JAVA-based agent platform evaluation, Holon. Multi Agent Syst. Manufact. 2744 (2003) 47–58.
[32] K. Burbeck, D. Garpe, S. Nadjm-Tehrani, Scale-up and performance studies of three agent platforms, in: Proc. 23rd IEEE International Performance, Computing, and Communications Conference (IPCCC 2004), 2004, pp. 857–863.
[33] E. Cortese, F. Quarta, G. Vitaglione, Scalability and Performance of JADE Message Transport System, Italia Lab, 2002.
[34] L. Mulet, J.M. Such, J.M. Alberola, Performance evaluation of open-source multiagent platforms, in: Proc. Proceedings of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems, ACM, 2006, pp. 1107–1109.
[35] J. Sudeikat, L. Braubach, A. Pokahr, W. Lamersdorf, Evaluation of agent-oriented software methodologies—examination of the gap between modeling and platform, Agent Orient. Softw. Eng. V 3382 (2005) 126–141.
[36] F. Bellifemine, JADE: a FIPA-compliant agent framework, in: Proc. Proceedings of the Fourth Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology, The Practical Application Company Ltd., 1999, pp. 97–108.
[37] F. Bellifemine, A. Poggi, G. Rimassa, Developing multi-agent systems with a FIPA-compliant agent framework, Software 31 (2) (2001) 103–128 http://dx.doi.org/10.1002/1097-024x(200102)31:2<103::aid-spe358>3.0.co;2-o.
[38] G. Caire, JADE Tutorial: Application-Defined Content Languages and Ontologies, Telecom Italia Laboratory, 2002.
[39] N. Khalid, M. Pasha, S. Ur Rehman, H.F. Ahmad, H. Suguri, Ontology services between agents and OWL based web services, in: Proc. Semantics, Knowledge and Grid, Third International Conference on, 2007, pp. 176–181.
[40] F.A. Sperotto, D.F. Adamatti, A proposal for interoperability to agent communication using synonyms, in: Proc. Social Simulation (BWSS), 2012 Third Brazilian Workshop on Social Simulation, 2012, pp. 39–43.
[41] A. Karanastasi, N. Matsatsinis, Agent technology meets the semantic web: interoperability and communication issues, in: Emergent Web Intelligence: Advanced Semantic Technologies, Advanced Information and Knowledge Processing, Springer London, 2010, pp. 105–120.
[42] J.L. Cardoso de Moraes, W. Lopes de Souza, L. Ferreira Pires, A. Francisco do Prado, A novel architecture for message exchange in pervasive healthcare based on the use of intelligent agents, in: Proc. 10th IEEE/ACS International Conference on Computer Systems and Applications (AICCSA), IEEE, 2013, p. 8.
[43] A. Moreno, A. Valls, J. Bocio, Management of hospital teams for organ transplants using multi-agent systems, in: Artificial Intelligence in Medicine, Proceedings 2101, 2001, pp. 374–383.
[44] K.-C. Yeh, Y.C. Chen, C.C. Chen, Reuy-Shun Chen, An RFID-based software agent framework on pervasive health service, in: Proceedings of the 2009 2nd International Conference on Biomedical Engineering and Informatics, vol. 1–4, 2009, pp. 2031–2034.
[45] M.N.K. Boulos, Q. Caib, J.A. Padgetc, G. Rushtonb, Using software agents to preserve individual health data confidentiality in micro-scale geographical analyses, J. Biomed. Inform. 39 (2) (2006) 160–170 http://dx.doi.org/10.1016/j.jbi.2005.06.003.
[46] A. Moreno, C. Garbay, Software agents in health care, Artif. Intell. Med. 27 (3) (2003) 229–232 http://dx.doi.org/10.1016/s0933-3657(03)00004-6.
[47] K.P. Logan, Prognostic software agents for machinery health monitoring, in: Proc. Aerospace Conference Proceedings, IEEE, 2003, pp. 3213–3225.
[48] D. Isern, D. Sánchez, A. Moreno, Agents applied in health care: a review, Int. J. Med. Inform. 79 (3) (2010) 145–166 http://dx.doi.org/10.1016/j.ijmedinf.2010.01.003.
[49] I. Baffo, G. Stecca, T. Kaihara, A multi agent system approach for hospital's drugs management using combinatorial auctions, in: Proc. 8th IEEE International Conference on Industrial Informatics (INDIN), 2010, pp. 945–949.
[50] V. Urovi, A.C. Olivieri, S. Bromuri, N. Fornara, M.I. Schumacher, A peer to peer agent coordination framework for IHE based cross-community health record exchange, in: Proc. Proceedings of the 28th Annual ACM Symposium on Applied Computing, ACM, 2013, pp. 1355–1362.
[51] J. Vazquez-Salceda, J.A. Padgetemail, U. Cortésemail, A. López-Navidademail, F. Caballero, Formalizing an electronic institution for the distribution of human tissues, Artif. Intell. Med. 27 (3) (2003) 233–258 http://dx.doi.org/10.1016/s0933-3657(03)00005-8.
[52] M.A. Munoz, M. Rodriguez, J. Favela, A.I. Martinez-Garcia, V.M. Gonzalez, Context-aware mobile communication in hospitals, Computer 36 (9) (2003) 38–46 http://dx.doi.org/10.1109/mc.2003.1231193.
[53] H. Kashfi, An openEHR-based clinical decision support system: a case study, Stud. Health Technol. Inform. 150 (2009) 348–359.
[54] J. Lahteenmaki, J. Leppanen, H. Kaijanranta, Interoperability of personal health records, in: Annual International Conference of the IEEE Engineering in Medicine and Biology Society, 2009, pp. 1726–1729.
[55] M. Batet, D. Isern, L. Marin, S. Martínez, A. Moreno, D. Sánchez, et al., Knowledge-driven delivery of home care services, J. Intell. Inf. Syst. 38 (1) (2012) 95–130 http://dx.doi.org/10.1007/s10844-010-0145-0.
[56] D. Isern, A. Moreno, D. Sánchez, Á. Hajnal, G. Pedone, L.Z. Varga, Agent-based execution of personalised home care treatments, Appl. Intell. 34 (2) (2011) 155–180 http://dx.doi.org/10.1007/s10489-009-0187-6.
[57] F. Campana, A. Moreno, D. Riaño, L.Z. Varga, K4Care: knowledge-based homecare e-services for an ageing Europe, in: Agent Technology and e-Health, Whitestein Series in Software Agent Technologies and Autonomic Computing, Birkhäuser Basel, 2008, pp. 95–115.
[58] D.T. Ross, Structured analysis (SA): a language for communicating ideas, IEEE Trans. Softw. Eng. SE-3 (1) (1977) 16–34 http://dx.doi.org/10.1109/TSE.1977.229900.
[59] L. Spagnoli, I. Almeida, K. Becker, A.P. Blois, C.M. Werner, Adaptation and composition within component architecture specification, in: Reuse of Off-the-Shelf Components, Proceedings 4039, 2006, pp. 142–155.
[60] R.M.M. Braga, C.M. Werner, M. Mattoso, Odyssey-search: a multi-agent system for component information search and retrieval, J. Syst. Softw. 79 (2) (2006) 204–215 http://dx.doi.org/10.1016/j.jss.2005.05.002.
[61] K. Pohl, G. Böckle, F.J. van der Linden, Software Product Line Engineering: Foundations, Principles and Techniques, Springer-Verlag New York, Inc., 2005.
[62] J.L. Cardoso de Moraes, W. Lopes de Souza, L. Ferreira Pires, A. Francisco do Prado, A novel approach to developing applications in the pervasive healthcare environment through the use of archetypes, in: Proc. 13th International Conference on Computational Science and Its Applications (ICCSA 2013), Springer, 2013, pp. 1–15.
[63] M. Marcos, J.A. Maldonado, B. Martínez-Salvador, D. Boscá, M. Robles, Interoperability of clinical decision-support systems and electronic health records using archetypes: a case study in clinical trial eligibility, J. Biomed. Inform.

46 (4) (2013) 676–689 http://dx.doi.org/10.1016/j.jbi.2013.05.004.
[64] D. Alur, D. Malks, J. Crupi, Core J2EE Patterns: Best Practices and Design Strategies, Prentice Hall PTR, 2001, p. 486.
[65] J.E. Bardram, Pervasive healthcare as a scientific discipline, Methods Inf. Med. 47 (3) (2008) 178–186 http://dx.doi.org/10.3414/me9107.
[66] F.D. Davis, Perceived usefulness, perceived ease of use, and user acceptance of information technology, MIS Q. 13 (3) (1989) 319–340 http://dx.doi.org/10.2307/249008.
[67] P.Y.K. Chau, P.J.H. Hu, Investigating healthcare professionals' decisions to accept telemedicine technology: an empirical test of competing theories, Inf. Manag. 39 (4) (2002) 297–311 http://dx.doi.org/10.1016/s0378-7206(01)00098-2.
[68] R. Likert, A Technique for the Measurement of Attitudes, Science Press, 1932, p. 55.
[69] J.L. Cardoso de Moraes, W. Lopes de Souza, L. Ferreira Pires, A. Francisco do Prado, Using the dual-level modeling approach to develop applications for pervasive healthcare, J. Mob. Multimed. 9 (1–2) (2013) 111–127.
[70] J.L. Cardoso de Moraes, W. Lopes de Souza, L. Ferreira Pires, A. Francisco do Prado, Towards a reusable architecture for message exchange in pervasive healthcare, in: Proc. 15th International Conference on Enterprise Information Systems (ICEIS 2013), SCITEPRESS—Science and Technology Publications, 2013, pp. 393–402.
[71] J.C. Nunnally, Psychometric Theory, McGraw-Hill, 1994, p. 736.
[72] J. Henseler, W.W. Chin, A comparison of approaches for the analysis of interaction effects between latent variables using partial least squares path modeling, Struct. Equ. Modeling 17 (1) (2010) 82–109 http://dx.doi.org/10.1080/10705510903439003.
[73] F.D. Davis, V. Venkatesh, Toward preprototype user acceptance testing of new information systems: implications for software project management, IEEE Trans. Eng. Manag. 51 (1) (2004) 31–46 http://dx.doi.org/10.1109/tem.2003.822468.