Implementing an Integrative Multi-agent Clinical Decision Support System with Open Source Software

Jelber Sayyad Shirabad[1] Szymon Wilk[2,*] Wojtek Michalowski[1] and Ken Farion[3]

[1] University of Ottawa, Ottawa, Canada
[2] Poznan University of Technology, Poznan, Poland
[3] Children's Hospital of Eastern Ontario, Ottawa, Canada

Abstract

Clinical decision making is a complex multi-stage process. Decision support can play an important role at each stage of this process. At present, the majority of clinical decision support systems have been focused on supporting only certain stages. In this paper we present the design and implementation of MET3 – a prototype multi-agent system providing an integrative decision support that spans over the entire decision making process. The system helps physicians with data collection, diagnosis formulation, treatment planning and finding supporting evidence. MET3 integrates with external hospital information systems via HL7 messages and runs on various computing platforms available at the point of care (e.g., tablet computers, mobile phones). Building MET3 required sophisticated and reliable software technologies. In the past decade the open source software movement has produced mature, stable, industrial strength software systems with a large user base. Therefore, one of the decisions that should be considered before developing or acquiring a decision support system is whether or not one could use open source technologies instead of proprietary software. We believe MET3 shows that the answer to this question is positive.

Keywords Clinical decision support, Agent based systems, Open source systems, JADE, Protégé

1 Introduction

Clinical decision making follows a complex multi-stage process, which involves data collection, diagnosis formulation and evaluation, and finally treatment planning. Figure 1 represents a hypothesis-driven model of this process. We selected this particular model because of evidence that it has been followed by physicians who use patient data in electronic form [1]. Extensive use of electronic patient data is common in clinical decision support systems (CDSSs). In our research we are concerned with the first three stages of the model, and the follow-up stage (assessment of the impact of implemented treatment on patient's condition) is presented just for completeness.

[*] Research described in this paper was conducted while Dr. Wilk was a postdoctoral fellow at the Telfer School of Management, University of Ottawa.

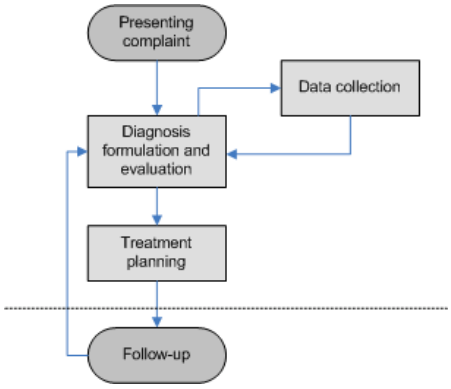


Figure 1. Hypothesis-driven model of decision making process

The physician starts with formulating a set of possible diagnoses based on the presenting complaint. This set is evaluated and revised iteratively by collecting additional information about the patient's condition, until the final diagnosis is established. The actual collection of patient data relies on the knowledge of what to collect and how to collect it effectively and accurately. This stage can be supported through a variety of structured data collection tools [2]. Establishing the final diagnosis allows developing treatment plans for the patient. This stage is usually supported by clinical practice guidelines (CPGs) [3] that summarize and encapsulate knowledge for the use of physicians at the point of care. Such knowledge is vast, often resides in external repositories (e.g., The Cochrane Library [4]), and it is associated with the notion of evidence-based medicine [5].

Individual stages from the decision making process are complex and clinical decision support could play an important role during each stage. This notwithstanding, the majority of clinical decision support systems (CDSSs, see [6] for a review) support either individual stages or integrate support for at most two of the stages (e.g., data collection and diagnosis formulation), but not all three. Considering the existing state of CDSS technology and challenges faced by CDSS developers and users [7], we propose a new type of CDSS – an integrative CDSS (ICDSS) [8] that provides decision support spanning the whole decision making process. The use of such a system is optional. In other words, the physician can ignore it when she thinks it is not necessary. We also propose a new design blueprint for ICDSS that combines principles of ontology-driven design with multi-agent architecture.

In this paper we describe our implementation of the MET3 system – a prototype of an ICDSS. MET3 is a successor of our two earlier CDSSs, MET1 [9] and MET2 [10]. MET1 was a CDSS aimed at supporting diagnosis of selected acute pain presentation. It was based on the client-server architecture with the client running on handheld computers. In MET2 we introduced ontological models that describe the essential aspects of system functionality (processed data, user interface, decision support). This significantly improved the flexibility and extensibility of the system, resulting in a customizable client that runs on various computing platforms such as mobile devices and desktop computers. MET2 relied on the client-server architecture and provided decision support for diagnosis formulation. In MET3, we moved to the new multi-agent architecture and added support for the remaining stages in the decision making process, particularly for the treatment planning stage which includes provision of clinical evidence.

In our research we have focused primarily on the decision support functionality and fully implemented it for selected clinical problems (presenting complaints). The remaining functionality such as user management and access control was implemented at a level which allowed the system to be tested in a laboratory setting. Specifically, our primary goal was to

implement a prototype ICDSS that meets the following requirements:

- Provides integrated support for data collection, diagnosis formulation and treatment planning (including evidence retrieval).
- Runs on different platforms to provide better availability at the point of care, where it is difficult to take any specific platform for granted.
- Is capable of interacting with hospital information systems (HIS), especially with an electronic patient record (EPR) system, via HL7 messages to provide realistic integration with existing healthcare systems.

It is obvious that implementing such an ICDSS (even at a prototype level) requires a fair amount of investment, both in its initial procurement and throughout its lifetime of service. Therefore it is important to decide about the characteristics of such a system. In the past this meant deciding about the functionality provided by such systems. However, the world of software development has seen a strong movement towards development of open source software (OSS) and freely available systems. This has motivated many researchers and software developers to take greater advantage of OSS as an alternative for expensive and often limiting commercial products. As a result one should decide whether the ICDSS system is proprietary (or consists of such components) or it is built using available OSS. This paper advocates the second choice. We believe the current OSS offers a wide range of feature rich and stable libraries and systems to build an ICDSS. Furthermore, considering the requirements mentioned above and the novelty of MET3, it is hard to implement such a system using commercial alternatives that are flexible and provide the needed functionality, while supporting required technologies. Therefore, a secondary goal of this paper is to show how one can build a relatively complex ICDSS by only using OSS and free software.

Being concerned with implementation issues, this paper focuses on later phases of software development life cycle and provides fair amount of technical design and implementation details regarding MET3 system. The paper is organized as follows. In section 2 we present related research while section 3 discusses the design of MET3. Third party OSS software used by MET3 is presented in section 4. In section 5 we present a sample scenario with clinical relevance which demonstrates the use of MET3 and discusses resulting interactions among agents. This is followed in section 6 by the details of implementing agent behaviours and interactions. In section 7 we discuss simulated testing of MET3's functionality. In section 8 we discuss the implications of the use of OSS in implementing an ICDSS. Our conclusions are presented in section 9.

2 Related Work

In this section we provide examples of the use of multi-agent systems in healthcare applications.

PalliSys is a prototypical agent-based system designed to facilitate collection, analysis and access to palliative patient data [11]. Palliative patients submit self assessment reports about ten predefined health indicators to the system. This information is managed by a *Database Wrapper* agent which also controls access to the data. This is akin to the role of the Data Manager agent in MET3. Similar to MET3 there is one agent per doctor which allows them to access the patient's data. Through the same agent a doctor can define one or more alarms. If the conditions defined in an alarm hold, a notification is sent to the doctor's agent. For each patient there is an agent which continually monitors submitted reports and evaluates them against alarms defined for that patient. Since there are well defined criteria as to when the agent should send an alarm, patient agents can act proactively. In MET3 the doctor decides when agents such as Diagnosis Suggester, Treatment Suggester or Evidence Provider should be consulted. MET3 users interact

directly with patients, and they decide when to use a particular agent.

MADIP is a prototype implementation of a mobile Multi-agent based Distributed Information Platform [12]. The aim is to notify a health care provider in case of an abnormality, offer distance medical advice, and continuously monitor the patients. MADIP's design includes *User* agents and a *Resource* agent whose roles are close to MET'3 Encounter Assistant (or user) and Data Manager agents. User agents allow for launching *Physician* agents that act on behalf of a physician and are capable of monitoring a patient's condition. These physician agents pass patient's data to a diagnosis agent which, upon detecting anomalies, can request an *External Services* agent to inform the physician through technologies such SMS texts. MET3 communicates with the outside world via a dedicated agent and standard HL7 messaging. MADIP also includes a *Knowledge-based Data Server* which maintains the patient monitoring data, electronic data records, and threshold values for monitored values. MET3 on the other hand does not encode threshold values in its knowledgebase Such thresholds are captured instead in diagnosis models. Similar to MET3, the system is implemented in Java and uses JADE and MySQL. MET3 uses MySQL for storing the evidence data while MADIP uses it to implement the knowledge-based data server.

In [13] the authors describe an agent based system designed to help the user find information about available healthcare services in a city, book an appointment, provide access to patient data during a visit and schedule the needed medical tests. The system uses JADE to implement the MAS and Protégé to create and access an ontology. Similar to MET3, this ontology contains patient-related information, such as visits and medical treatments.

Intelligent agents have been used to aid in diagnosing cardiac disorders [14]. The system consists of 5 agents which use different technologies (e.g., rule based system, neural networks, signal analysis) and radiologist input to assist with the diagnosis. A physician is in charge of collecting and inputting the relevant data. A *Task Manager* assigns subtasks to individual agents and combines the results obtained from them to generate an overall response. The *Communication Manager* is in charge of conversion of agent inputs and outputs, so that the relevant agents can understand them. Evidence based medicine is used in developing a knowledge base for the rule based agent. MET3 does not employ an explicit centralized control agent. Each agent knows what to request from other agents to accomplish its goals. MET3 provides the evidence in the form of reviews and paper summaries directly to the physician, as she is the final authority to accept or reject it. MET3 uses a standard protocol and agent communication language provided by JADE.

[15] describes the development of an agent-based decision support system to improve the treatment of infectious diseases which are initially being treated with antibiotics of restricted use. The goal is to automate the process of revising and proposing alternative antibiotics which is typically done manually by the pharmacy department. The system includes agents representing nurses and physicians. These agents accept input from the users they represent and act as means for contacting them. A *Guardian Agent* is attached to a patient to monitor the given prescriptions. The *Pharmacy Expert Agent* is in charge of proposing possible changes in a medical order. This agent uses a knowledge-based approach for decision making.

Intelligent Healthcare Knowledge Assistant (IHKA) is a client-server system that incorporates agents for knowledge gathering, filtering, adaptation and acquisition from databases storing empirical knowledge, case-bases storing experiential knowledge, scenario-bases storing tacit knowledge and document-bases storing explicit knowledge [16]. The user enters her knowledge

specification queries through a *Web Interface* agent. These queries refer to terms and entities maintained in IHKA's Ontology. A *Knowledge Retrieval* agent is in charge of retrieving relevant knowledge from the above mentioned sources. A *Knowledge Procurement* agent will search the web to satisfy the user query, if the requested knowledge could not be found in the system's internal sources. Finally, a *Presentation* agent formats the results so that they can be presented to the user via the web agent. Formatting instructions are captured in MET3's interface models, which makes their creation and maintenance much easier. This feature has eliminated the need for a separate presentation agent. MET3 uses a single database that contains medical evidence knowledge. The user does not have to enter a query to retrieve the evidence as for each presenting complaint there is a support provider in the knowledge that allows the evidence provider agent to link together the relevant patient data, the diagnosis and the existing evidence.

COSMOA [17] is a prototype multi-agent system that has been designed to support the decision-making process during medical response to a large-scale incident. The system simulates the monitoring of news feeds and emergency service reports of incidents to determine if an incident has occurred, and its nature. Similar to MET3, this system was designed following an ontology-centric approach to multi-agent system design. Rules, heuristics and statistical attributes that define an agent's behaviour are placed in an ontology layer. This allows for changing the behaviour of the agents that rely on these rules and heuristics by changing the ontology. This provides a flexibility in design similar to the one offered by MET3. However, MET3's decision models are not stored in the ontology. Unlike MET3 the user agents (one per hospital) need to negotiate the scheduling conflicts that may arise among them. Both systems are Java based. They use JADE for implementing the agent system and Protégé for the creation of ontologies.

Multi-Agent System for Integration of Data (MAID) is an MAS used as part of a deployed Virtual Electronic Patient Record (VEPR) system [18]. VEPR makes patient records available at all points of care. MAID is responsible for automatic document retrieval from a number of different departmental information systems (DIS). This system is another example of using JADE for implementing an MAS. Corresponding to each DIS there is one *List* agent for retrieving the list of new reports from that DIS. A dedicated *Balancer* agent distributes the load of retrieving actual documents among a number of *File* agents. To do this the balancer and list agents implement the FIPA-Contract-Net negotiation protocol. In contrast, MET3 uses a FIPA-Request interaction protocol for communication among all its agents. These documents are stored in a centralized document repository. If a specific document is requested by a user but it does not exist in the repository at the time, an *Express* agent directly retrieves it from the DIS that holds the document. Authentication of the user and his interactions with the system are done through a web interface. MET3 on the other hand uses an agent for these services.

While this section does not present a complete list of agent-based healthcare systems, it provides a snapshot of the current landscape in this area. A study of open source clinical systems by Hogarth and Turner [19] indicates that the largest segments of open source clinical applications are clinical information systems and imaging systems. They made up of nearly half of all open source clinical software in their study. Decision support systems were present but constituted only about 8.4% of investigated systems. Agent based health care systems are a relatively recent development. Not all of these systems are based on open source technologies. As the above review suggests, many of these systems are stand alone prototypes and do not communicate with an existing hospital system or do not use industry standards such as HL7. MET3 integrates some of the less supported functionalities in clinical software — namely diagnosis, treatment planning, and provision of evidence in support of these plans. Furthermore, the architecture of

the system allows for the easy expansion of support for multiple illnesses (or presenting complaints). Most clinical systems are designed to be used in a conventional desktop environment. By contrast, MET3 is designed to be used on multiple platforms, including mobile devices.

3 MET3's Design

In this section we discuss the design of the MET3 ICDSS. Section 3.1 briefly introduces multi-agent systems and the particular methodology used to arrive at the design of MET3. Section 3.2 presents the resulting design. Modeling the domain of interest is an essential part of developing any CDSS. We discuss the major concepts defined in MET3 ontology is section 3.3.

3.1 Background

Complex system functionality should be partitioned into independent and distributed entities that either provide services (corresponding to main functions of the system) or request them [20]. Such partitioning is supported by service-oriented [21] and agent-oriented architectures [22]. While these two architectures slowly converge [23], the agent-oriented architecture demonstrate properties (persistency and intelligent behaviour of entities) that make it especially useful for an ICDSS such as MET3.

A multi-agent system (MAS), i.e., a system based on the multi-agent architecture, is composed of multiple interacting entities known as agents. Typically, the problems addressed by an MAS are too complex to be solved by an individual agent. Therefore, in such systems, agents exchange and share information in order to achieve the overall goal of the system [22]. During these interactions agents react to changes in their environment in a proactive, autonomous and intelligent manner. According to [24] "the key abstraction models that define the agent-oriented mindset are agents, interactions and organizations."

While the primary focus of this paper is covering implementation aspects of MET3, software development efforts start with capturing system requirements, and are followed by analysis and design steps. Thus, we first needed to use a methodology appropriate for development of an MAS.

There are different methodologies for analysis and design of MAS, including Gaia [25], Tropos [26], MAS-CommonKADS [27], and O-MASE [24], to name a few. We used the O-MASE methodology in designing MET3 because of the following benefits:

- Easy customization of the analysis and design process
- Flexibility in the agents' design – from passive to intelligent
- Use of the Agent UML (AUML), which helps with the general readability of the produced design.

O-MaSE views MAS as an organization of agents. This organization has a specific purpose that defines its overall goal, which itself is decomposed into sub–goals. To achieve these goals agents are assigned specific roles to play. The assignment of roles to agents can be either static (during design time) or dynamic (during run time). In order to communicate with each other, agents use a common language. Agents perceive or sense objects in the environment via actions. This environment is modeled by a Domain Model, which defines the types of objects in the environment and the relations between them [24].

The tasks in the O-MASE methodology fall into one of the three categories: requirements engineering, analysis, and design. Requirements engineering is concerned with translating

requirements into goals and is summarized by a goal model. The outcome of analysis is the definition of an organization model, possible roles and their interactions, and a domain model. The outcome of design is the definition of agent classes, the communication protocols used by agents, and plans that agents follow in order to achieve specific goals. For systems such as ours, where agents are reactive and they are assigned static roles and goals at design time, a basic O-MaSE process is sufficient [24]. The tasks in the basic process are: goal modeling and refinement, domain modeling, agent modeling, protocols modeling and plans modeling. The products of this basic process are the system goals (capturing what the system aims to achieve), agent classes (capturing different types of agents in the system and the goals they achieve), agent plans (capturing the algorithms used by agents to achieve the goals assigned to them), and inter-agent protocols (capturing how agents communicate with each other).

3.2 Design Outcome

Since the focus of this paper is the implementation of MET3, our starting point is the *Agent class model* as specified in the O-MaSE process. The agent class model shows the agents composing MET3 ICDSS.

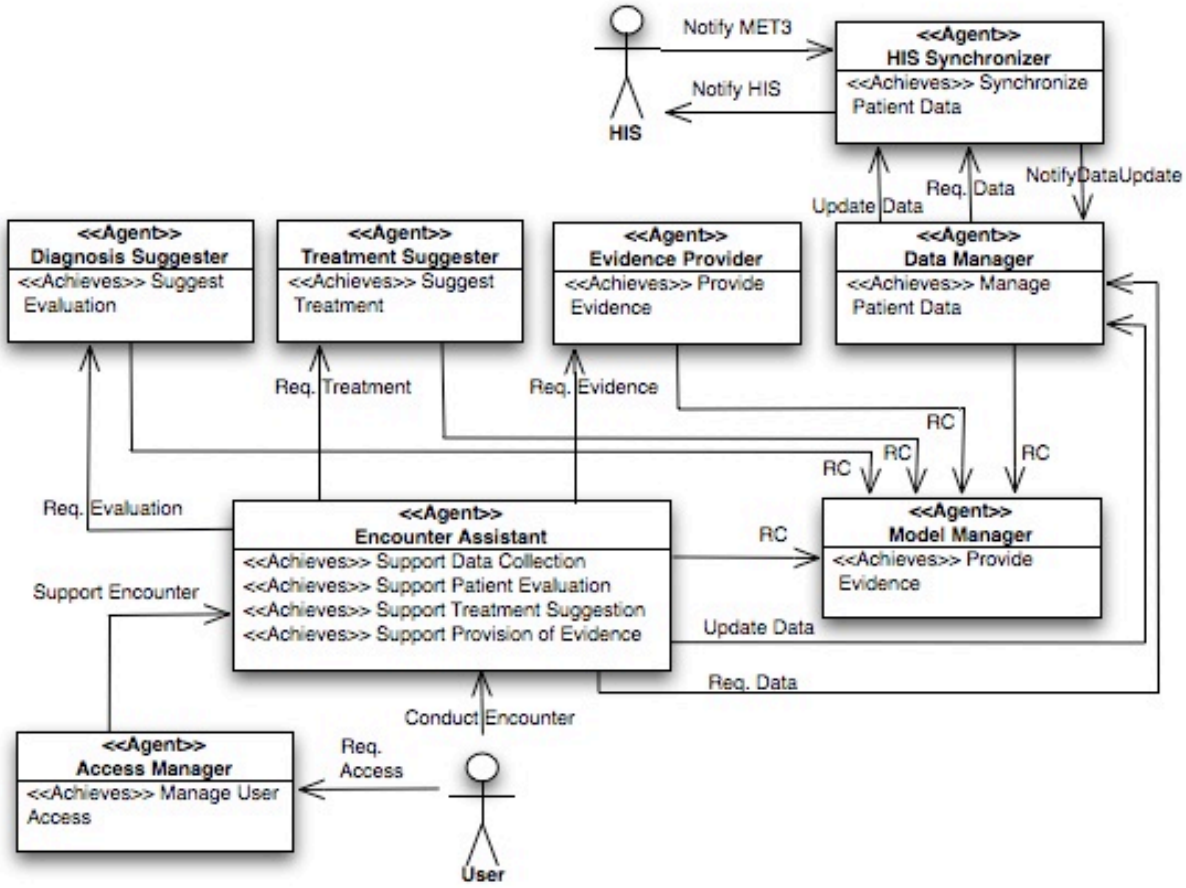


Figure 2. Agent class diagram

Figure 2 shows the the Agent class model for MET3 system. Individual agent classes are represented using rectangles and denoted by «Agent» keyword. The goals that an agent achieves are denoted using the «achieves» keyword. Communications between agent classes are identified by arrows, where the direction of an arrow identifies the initiating and receiving ends. RC in this figure stands for *Request Concept*. As can be seen in Figure 1, MET3 consists of the following agents:

- *Access Manager Agent (AMA)*: Controls access to the MET3 system.
- *Model Manager Agent (MMA)*: Acts as a gateway to MET3's knowledge base and retrieves models upon requests from other agents (e.g., interface model, diagnosis decision model, treatment decision model), required to achieve individual goals.
- *Data Manager Agent (DMA)*: Is in charge of retrieving and storing patient data locally.
- *HIS Synchronizer Agent (HSA)*: Acts as an exchange gateway for synchronizing the data updates between external systems; e.g. HIS, and MET3.
- *Diagnosis Suggester Agent (DSA)*: Suggests a diagnosis for a patient's presenting complaint based on the available data.
- *Treatment Suggester Agent (TSA)*: Suggests a treatment plan for a patient using the treatment model developed for a specific complaint.
- *Evidence Provider Agent (EPA)*: Provides clinical evidence; e.g. systematic reviews, related to a treatment plan for a specific patient.
- *Encounter Assistant Agent (EAA)*: Allows the user to access MET3's functionality through appropriate graphical user interface.

Most of MET3's agents are reactive; i.e., they wait for requests and respond to them. Fully proactive and autonomous behavior is demonstrated by HSA that actively monitors information generated by HIS (in form of HL7 messages), and reacts accordingly by notifying other agents or external hospital systems. Arguably this system could also have been implemented as a collection of web services. However, as stated in [28], every agent can be considered a service but not every service can be considered an agent; e.g., a service does not necessarily exhibit an autonomous behaviour. Having an agent based infrastructure in place (complemented with a knowledge base) should allow for the easier introduction of potentially more autonomous features in the future. The limited autonomy of MET3 agents results from the specificity of the decision making process, where the physician acts as the "super-agent" controlling other agents and requesting their services, if needed. Furthermore, having reactive agents such as DSA, TSA, and EPA, which provide services that are focused on decision support, is in line with the idea of defining and standardizing decision support services put forth by HL7 initiative [29].

As shown in Figure 3, architecturally, MET3 ICDSS can be seen as an agent-based layer and a number of additional components used to provide overall system functionality. In this figure, "Agent System" refers to the environment which includes all eight agents and other infrastructure such as the run time environment and the communication support, provided by JADE framework, which allows these agents to operate. MET3 relies on a knowledge base derived from a domain ontology, which is used to support the system's overall functionality and is accessed through MMA. The same underlying technology is also used by DMA to maintain patient data locally, as we will explain later. In implementing the requirements for providing clinical evidence EPA relies on an evidence database, while DSA and TSA use diagnosis and treatment models and associated solvers. To provide flexible and easily maintainable solutions these two agents delegate the logic of actual diagnosis to external programs created for each decision model. We refer to these programs as solvers. Solvers know how to run patient specific data through a decision model to obtain a suggestion/decision from the model. These decision models are typically built from retrospectively or prospectively collected data, however we do not limit MET3's decision models to be built following a specific approach. Any system that generates a decision model should provide means to use the generated model by other programs. But these means of interaction with a decision model differ from one vendor to the next as well as different versions of these systems themselves.

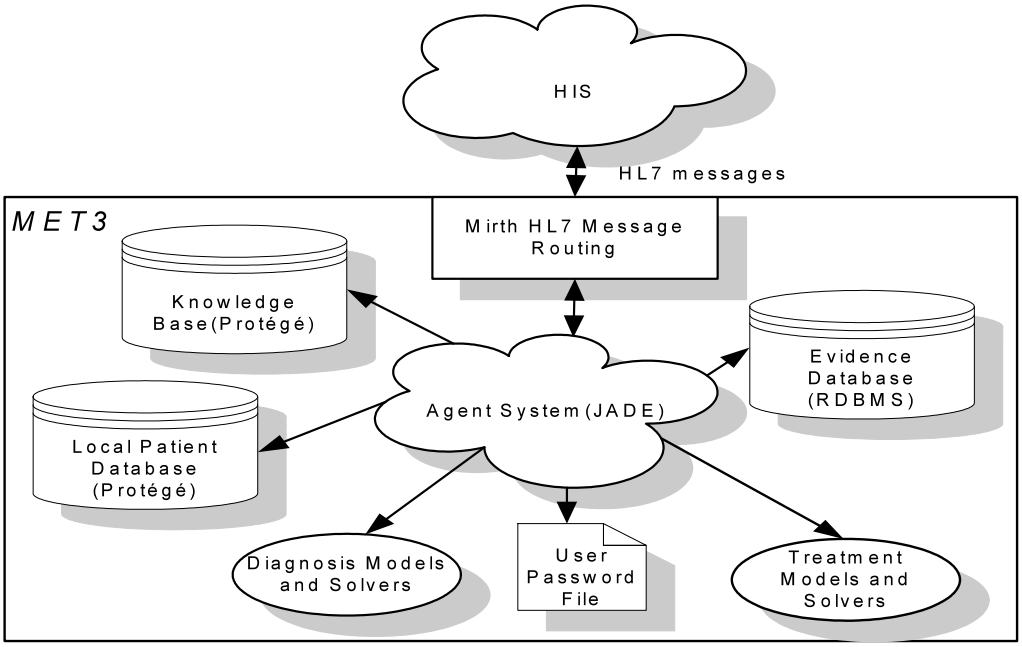


Figure 3. MET3 architecture

Support for integration with the existing HIS systems is achieved by using Mirth Connect[1] HL7 engine. Mirth, as discussed below, acts as a message router between MET3's HSA and the external HIS. Finally, AMA uses a password file to verify user's access rights.

3.3 MET3's Ontology

MET3 ontology plays an essential role in the system architecture as it provides common nomenclature so the agents understand the content of the messages. It also defines other concepts that model the environment in which MET3 operates. All MET3 agents know what to do in response to requests sent to them. They also know which pieces of knowledge in MET3's knowledge base are relevant to performing their tasks and access this knowledge through requests sent to MMA.

MET3's ontology is logically divided into five parts: Meta-data, Data, Interface, Support and Transport. The first four parts are implementation independent consequences of the design process, while Transport classes are due to the use of JADE in implementing MET3. In the following discussion of the ontology we use the terms "concept" and "class" interchangeably as is the case in Protégé.

Meta-data classes describe the clinical data stored in the ontology; e.g., domain and accepted values for a clinical attribute. Instances of Data classes are used for storing the actual clinical data, as MET3 currently uses Protégé as a database. Data classes use Metadata classes to describe stored values.

Interface classes describe MET3 user interfaces. By adopting ontology-based interface definitions we reduce the coding burden associated with developing multiple user interfaces. However, this simplification is achieved at the cost of standardizing the look and feel of the user

[1] Throughout the paper we refer to this product with its original and widely known name, Mirth.

interfaces, such as the way widgets are laid out. Consequently, one sacrifices certain degree of flexibility to achieve ease of extension and maintenance.

Support classes are used to provide information about the decision models and solvers used to evaluate a patient by MET3. They also describe operations to be performed on the raw patient data before they are input to the models used by DSA or TSA. We refer to these operations as transformations.

Support classes represent the essential information needed by MET3's agents to provide diagnosis (DSA), treatment (TSA), and evidence retrieval (EPA) functionality. The main class here is *Support Provider*. A support provider instance maintains the following information:

- The path to a decision model.
- An *Instance Creator*, which maintains a collection of transformations to be applied to the raw patient data to create a set of inputs that are used by the decision model. A decision model may only need a subset of the available patient data, or data that has been altered from its original form to what the model expects. Examples of such transformations are attribute selections and discretization.

In case of diagnosis and treatment planning it is important for MET3 to provide a flexible design that allows one to easily incorporate support for additional illnesses. Furthermore, these models may be generated using different technologies and methods. To this end specialized support provider classes called *Suggestion Provider(s)* are introduced. For example, there is an instance of suggestion provider corresponding to the asthma exacerbation predictor decision model used for diagnosis. Suggestion providers capture the following addition information:

- The location (or path) of the solver: As stated earlier, the solver is an external program that uses vendor specific programs and implements conventions such as input/output formats and parameters needed to generate a prediction using a particular decision model. Most times the solver uses external libraries, or OS provisions for executing external programs, to interact with the decision models. However, it could also directly implement this logic, as is the case for MET3's Abdominal Pain model.
- Parameters used by the model, if any (e.g. thresholds).
- Additional parameters needed by the solver; e.g., the declaration of variables used by the model.

Since DSA and TSA rely on specialized decision models to provide diagnostic and treatment suggestions, their performance depends on the quality of these models. In our previous research we focused on building diagnostic decision models by applying data mining techniques to data collected in retrospective chart studies. These models were usually represented in form of decision rules or decision trees. Such formats are widely used in clinical practice, thus the decision models could be verified by physicians before their practical use. We built rule-based models for diagnosis of abdominal pain [30] and scrotal pain [31], and a tree-based model for diagnosis of asthma exacerbations [32]. These models were also embedded in MET1 and MET2. MET1 with the diagnostic decision model for abdominal pain underwent a clinical trial in the emergency department where it demonstrated overall diagnostic accuracy comparable to the accuracy of emergency physicians [33].

Treatment decision models were introduced only in MET3 and unlike diagnosis decision models they are not learned from data. Instead they are created by converting appropriate CPGs (e.g. paediatric asthma guidelines published by the Canadian Association of Emergency Physicians [34]) into decision rules. These rules constitute the rule-based treatment decision models that are used by TSA to suggest proper treatment plans.

Diagnosis and treatment decision models are not embedded in DSA and TSA. Instead, the agents retrieve them when requested, making it relatively easy to maintain and update them. For example, when new patient data (with established gold standard diagnosis) becomes available, diagnosis models can be revised. This allows MET3 to benefit from new knowledge and experience during its operational life. Similarly, after modifying a CPG, the corresponding treatment model may be revised. Updating decision models may require updating the ontology; however, there is no need to change the agents and these updates can be disseminated without restarting the system.

Transport classes are used to implement agent communication as required by JADE. For agents to communicate properly they should share the same language, vocabulary as well as semantics for the content of messages. JADE framework provides base classes for defining an ontology, which allow for the implementation of this common vocabulary and defining the concepts in the domain which are used by the agents in exchanged messages. Additionally, the transport classes are used to semantically check the content of messages sent from one agent to the other. JADE's support for ontologies eases the burden of the programmer to deal with complex and structured content [35]. Moreover, Protégé provides a plug-in to generate Java classes corresponding to JADE ontology.

The elements used in the content of JADE messages can be semantically classified as [35]:

- Predicates, which are expressions that say something about the state of the world.
- Concepts, which are entities with complex structure, e.g. a patient or a visit
- Agent Actions, which are special concepts that indicate actions that can be performed by agents

Transport classes belong to one of the above categories. Every request in MET3 extends Agent Actions class. Every response in MET3 is an instance of the *Response* class which itself extends JADE's *Predicate* class. The content of this predicate is a concept which is the actual response to a request.

4 OSS Technologies Used

Similar to other modern software development efforts, to implement MET3 we relied on other third party systems and libraries to benefit from code reuse. In case of MET3 ICDSS we used the following OSS or free technologies:

a) A technology for implementing the agents and infrastructure of an MAS. Some of the potential candidates include Java Agent Services API, Trylian Agent Development Kit, Grasshopper, ZEUS, Cougaar and JADE. We chose JADE[2] [35], one of the most popular frameworks, which is being actively developed by industry and includes support for mobile platforms. Using JADE allowed us to benefit from pre-existing support for communication protocols including error handling. Additionally, the implementation of the agent behaviours was greatly simplified. We provide more details about this in section 6.

b) A technology to create, maintain and programmatically manipulate a domain's ontology and instances of the defined concepts. We chose Protégé for this purpose, as it has been used in a number of other projects. Protégé has been actively maintained and includes additional third party contributions in the form of add-ons.

c) A database system to maintain clinical evidence data used by the EPA. We chose MySQL, which is a very popular relational database. Other alternatives such as PostgreSQL can easily be used as drop-in replacements by the system.

[2] http://jade.tilab.com/

d) A technology to do full text search of the evidence database. Since the evidence data is stored in a relational database, we chose Hibernate Search[3] as it provides the required functionality by combining the full text search power of Apache Lucene[4] and object relational persistence.
e) A technology that allows MET3 to communicate with existing healthcare software. HL7 is a widely adopted standard in healthcare applications for exchanging data. MET3 is capable of intercepting as well as generating HL7 messages. We used Mirth[5] HL7 integration engine to route, filter, and transform messages between MET3 and the external HIS.
f) A technology that supports the deployment of MET3 agents on mobile devices with limited computing resources. For this we chose Sun Java Wireless Toolkit[6] (WTK) that allows developing applications for cell phones, personal digital assistants, and other small mobile devices.
g) A technology for generating decision models from retrospective data and executing these models. For this we used WEKA[7], which is a very well known and extensively used Java based system. WEKA provides a collection of machine learning algorithms that can be used for data mining tasks.

Although the selected OSS technologies present only one of many possible alternatives, we believe they represent a solution that ensures longevity, ease of maintenance, and future expansion of MET3 ICDSS. All these technologies have multi-year development history, are in widespread use, and in most cases are either directly developed or sponsored by industry.

5 MET3 at Work

To provide a better understanding of how MET3 works, in this section we provide a hypothetical, yet realistic, scenario of a clinical case in the emergency department setting and explain how specific MET3 agents act, how they cooperate, what information they use and what results they generate.

The scenario is shown in Figure 4. It covers all stages of the decision making process – data collection, diagnosis formulation and evaluation and treatment planning. Corresponding agent interaction diagrams are given in Figure 5. In all these diagrams it is assumed that the physician (Dr. Jane Brown) has already selected the patient (Peter Smith) from the list of patients currently handled by the system. This results in retrieving all available patient data from the Local Patient Database and making them available to EAA – a physician's gateway to MET3.

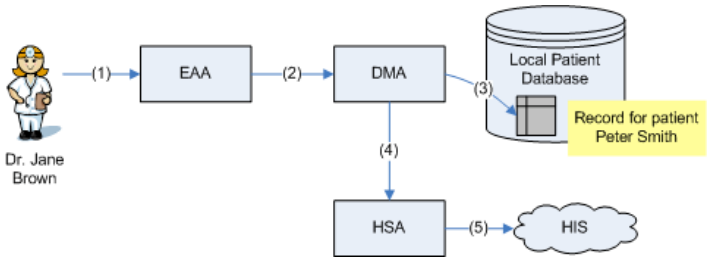
Peter Smith, a 4-years old boy, suffering from an asthma attack is brought by his parents to the emergency department. The emergency physician, Dr. Jane Brown is running MET3 on her tablet computer. She approaches the patient for a triage assessment; i.e., examines him for the first time, and asks his parents a few questions regarding his history and background. All collected information is entered into the system and stored. Dr. Brown follows the locally approved protocol for all asthmatic patients and prescribes bronchodilator treatment for Peter. After 30 minutes she approaches the patient again, conducts a repeated assessment and enters the data. This time Dr. Brown diagnoses the patient in

[3] http://search.hibernate.org/
[4] http://lucene.apache.org
[5] http://www.mirthproject.org
[6] http://java.sun.com/products/sjwtoolkit/
[7] http://www.cs.waikato.ac.nz/ml/weka/

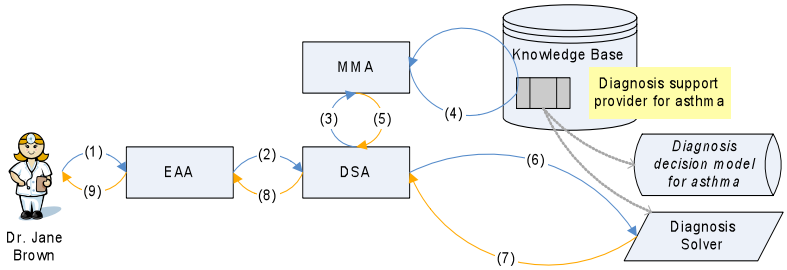
order to plan subsequent treatment. She suspects severe exacerbation and asks MET3 for a diagnosis to verify her opinion. The system indicates that moderate and severe exacerbations are possible; however, the former diagnosis is more likely. Dr. Brown revisits her opinion and confirms the moderate diagnosis. Then she asks MET3 for a treatment suggestion. The system proposes steroids and anticholinergics. Dr. Brown is not sure about providing steroids to the young patient, so she asks for evidence supporting such treatment. The system responds with reviews of clinical trials where steroids were given to patients with similar age and state. After checking a few reviews Dr. Brown decides to prescribe steroids for Peter.

Figure 4. A scenario of a clinical case

a) Data collection



b) Diagnosis formulation and evaluation



c) Treatment planning

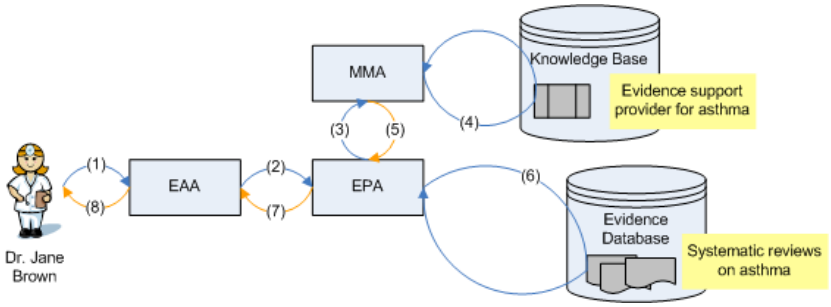


Figure 5. Agent interaction diagrams

In data collection stage (Figure 5a) Dr. Brown enters new data for Peter Smith (1) through the user interface provided by EAA. EAA sends a request to DMA (2). This request includes the updated information (e.g., results of a subsequent assessment). DMA updates the record of Peter Smith stored in the Local Patient Database according to the request (3) and also passes the updated data to HSA (4). Finally, HSA composes an appropriate HL7 message (observation report) using the updated data and passes it to HIS to ensure data consistency and recency.

In diagnosis formulation and evaluation stage (Figure 5b) Dr. Brown asks MET3 to provide a diagnosis for Peter (1). This causes EAA to send a request to DSA (2) which includes all of Peter Smith's data (EAA has already retrieved this information after selecting him from the patient list). DSA checks that Peter suffers from asthma and sends a request to MMA to return the corresponding diagnosis support provider (3). The diagnosis support provider for asthma points to the diagnosis decision model and its solver. MMA retrieves the diagnosis support provider from the Knowledge Base (4) and sends it back to DSA (5). DSA transforms Peter's data received from EAA according to transformations defined in the diagnosis support provider (e.g., discretizes them – see section 3.3 for more details). Then, it invokes the solver and passes it the transformed data and the pointer to diagnosis decision model for asthma (6). The solver returns elaborated diagnosis to DSA (7). DSA sends these to EAA (8), which finally reports them to the physician for verification.

All diagnosis solvers in MET3 are able to handle imperfect; e.g., incomplete, patient data. Instead of reporting a single suggestion, they report several suggestions with associated weights (more "likely" suggestions have greater weights). This reflects uncertainty that is inherent in diagnostic reasoning. Moreover, in each case (single or multiple diagnosis suggestions), the physician is able to overwrite the system's suggestion, which underlines the assistive role of MET3.

In the treatment planning stage (Figure 5c) Dr. Brown asks for a treatment suggestion and for evidence supporting the suggested treatment. Agent interactions for providing treatment suggestions are very similar to the interactions in diagnosis formulation and evaluation stage, so they are not repeated here. The major difference is that MMA retrieves the treatment support provider for asthma. The treatment support provider points to the proper treatment decision model and solver for asthma. Treatment solvers, similar to diagnostic solvers, are able to handle imperfect data. Again, the physician can always overwrite the suggested treatment with her own choices.

After MET3 provides treatment suggestions, Dr. Brown asks for evidence supporting steroids (1). In response, EAA sends a request to EPA (2). This request includes all of Peter Smith's data, as well as the potentially interesting treatment(s) (2). EPA identifies the complaint for Peter and sends a request to MMA to return the evidence support provider for asthma (3). The evidence support provider defines necessary data transformations that are needed to query the Evidence Database. These mostly involve mapping specific data values to search keywords. MMA retrieves the proper evidence support provider from the Knowledge Base (4) and returns it to EPA (5). EPA applies all defined data transformation to Peter's data and generates the search keywords. These keywords represent the selected treatment and patient characteristics. They are part of the set of keywords used to index the Evidence Database. Then EPA queries the Evidence Database to retrieve systematic reviews matching the search keywords (6). The retrieved reviews are the ones that describe patients similar to Peter who were given steroids. The reviews are sent back to EAA (7) and finally presented to Dr. Brown. If the query results in an empty answer set the physician is informed of the lack of supporting evidence in the evidence database.

6 Implementation of Agent Behaviours and Interactions

The implementation of agent behaviours (or plans) and inter-agent communications are some of the most important aspects of any MAS. Consider the example scenario and its corresponding agent interaction diagrams presented in the previous section. We need to implement both

individual agent behaviours as well as inter-agent communications. For instance in Figure 4.a when EAA sends a request to DSA to store Peter's data, we need to implement sending the request via a well defined message from EAA to DSA. On the receiving end, DSA should recognize this message and execute the code that satisfies this request. In practice there are predefined sequences of messages that are applicable in several situations and share the same communication patterns independent of the application domain. Such sequences of messages are known as interaction protocols [35]. All agent interactions in MET3 follow what is known as FIPA-Request interaction protocol. JADE provides a very well defined framework for implementing agent behaviours and common interaction protocols, which greatly simplifies implementation of an MAS. We cover these two topics in sections 6.1 and 6.2 respectively.

6.1 Implementing Agent Behaviours

Implementing the behaviour of an agent involves implementing the goals (and associated sub–goals) it is designed to achieve, as identified in the Agent class diagram. The overall behaviour of an agent can be divided into a set of independent (sub-) behaviours corresponding to these sub–goals. For instance, while the overall goal of the MMA is to retrieve various models from the knowledge base, it can be decomposed into retrieving GUI related models (done each time EAA displays a screen to the user), diagnosis models through diagnosis support provider (step 5 in Figure 4.b), treatment models through treatment support provider (step 5 in Figure 4.c), etc. Implementing each sub-goal itself may require a series of lower-level steps such as initializing variables, making calls to retrieve objects from the knowledge base, and setting up proper response message.

To implement individual sub–goals we define a subclass that extends/specializes the functionality available in JADE's *Behaviour* class or one of its subclasses. Two important subclasses of the *Behaviour* class are *SimpleBehaviour* and *CompositeBehaviour*. We will discuss specialized subclasses of these behaviour classes in the next section.

Once we determined the type of behaviour, we need to implement the actual action performed when the behaviour is executed. We also need to let an agent know when a particular behaviour is completed. To do this we only need to override the *action* and *done* methods in our specialized behaviour class.

6.2 Implementation of Inter-agent Protocols

Protocols in JADE are implemented using special kind of behaviours which are responsible for proper ordering of the message sequences for protocol they implement. The FIPA-Request protocol captures exchanges of messages between a requesting (initiator) agent and an answering (responder) agent. The source and destination of arrows in Figure 5 correspond to initiator and responder agents. JADE framework provides *Achieve Rational Effect* behaviour classes which are used to implement the FIPA-Request protocol. The *SimpleAchieveREInitiator* and *SimpleAchieveREResponder* behaviours can be used to implement the actions performed at the initiator and responder ends of the interaction, respectively. Similar to the case for the *Behaviour* class the programmer is expected to extend these classes and override specific methods to implement application specific requirements[8]. For instance, to implement step 3 in Figure 4.b we define a subclass of *SimpleAchieveREInitiator* which implements DSA's "request diagnosis support provider behaviour". To implement steps 4 and 5 carried by MMA we define

[8] For more details on these methods please see [35].

a subclass of *SimpleAchieveREResponder,* which implements the "retrieve diagnosis support provider" behaviour.

Corresponding to these simpler behaviours that implement the FIPA-Request protocol there are *AchieveREInitiator* and *AchieveREResponder* classes that are used to implement more complex interactions (behaviours) whereby the agent can register additional sub-behaviours as part of preparing the response to the request.

In MET3 all initiator agent behaviours are created by extending the *SimpleAchieveREInitiator* class. Following the terminology used by JADE, on the responding side we recognize two kinds of behaviours: simple and complex. A simple responder behaviour, such as the ones defined for MMA, can prepare the response to a request without the need to interact with other agents. A complex responder behaviour requires the responding agent to interact with other agents in order to prepare a response. In other words, in case of complex responses the responding agent acts as the initiator in a secondary interaction scenario. For instance, in Figure 4.c the EPA agent receives a request in step 2 from EAA to provide the evidence for the use of steroids. To be able to prepare a response, EPA requires additional information that it obtains by sending a request to MMA (step 3). The complex responder behaviours are created by extending *AchieveREResponder* class.

In summary, the general process of implementing an agent and its behaviours in MET3 is as follows:

- Identify the requests sent and received by the agent. The messages exchanged among agents are indicated in the Agent Class model but they require further refinement in terms of identifying the arguments passed along with the message and content returned as part of response. This is done by creating protocol models in O-MaSE process.
- For each request sent by the agent, develop an initiator behaviour by sub-classing the appropriate JADE class.
- For each request received (handled) by the agent develop a responder behaviour by sub-classing the appropriate JADE class.
- Develop a subclass of the *Agent* class, which will implement the agent itself. As part of the agent's start up process instances of the behaviours mentioned above are added to the agent. The details of selection and execution of the proper behaviours for a given request is handled by JADE's system code.

7 Testing MET3

For MET3 to operate in a hospital setting it needs to integrate with an existing HIS. Most HIS systems use HL7 messages to communicate with other healthcare software. An HL7 message path through such an integrated environment includes:

- A source node, which is part of HIS, and generates HL7 message for admitted patients.
- An intermediate node built around Mirth and used by MET3 to capture and route these messages.
- A receiving node; i.e. HSA, which is part of MET3 agent system.

Following the above message path we tested the ability of MET3 to handle the volume of messages typically generated in the Emergency Department (ED) of a hospital where our system can potentially be deployed. These tests were done in a lab setting. To do this we created an HL7 source node which simulates an ADT (admission-discharge-transfer) system generating ADT messages that would be created during the course of a patient's visit to the ED. For the purpose of this integration test we were only interested in patient admission messages. The

message generator was implemented as a prototype patient admission application that allows one to enter typical information collected at admission such as the patient's name, age and complaint. The application then generates an HL7 message containing this information and sends it to a preconfigured port on the computer running Mirth. We configured Mirth, MET3's underlying HL7 integration technology, so that it routes these messages to the specific port on the computer where HSA runs and listens for incoming messages.

To simulate a typical day in ED, we looked at available statistics for the ED of Children's Hospital of Eastern Ontario (CHEO) in Ottawa, ON. CHEO is a tertiary-care paediatric teaching hospital affiliated with the University of Ottawa serving patients up to 18 years of age. Based on available statistics we computed the average number of admissions in a day to be about 145 patients. We admitted 125 simulated patients, a relatively heavy load over a 12 hours period. Amongst these there were 15 patients with abdominal pain and 25 patients with asthma exacerbation, which would be further processed by MET3. We verified that corresponding patient visit records were created by DMA. We also simulated 10 concurrent sessions of patient assessment (meaning that 10 patients were assessed using MET3 at the same time). These tests showed the successful HL7 message routing and processing, as well as the use of various functionalities of the MET3 ICDSS. It is worth noting that the only difference between these tests and a deployment test would be a change in the source node that generates HL7 messages.

Additionally, we have also used Mirth in a different system employed during a clinical trial at CHEO. In this case the system was receiving real ADT messages from CHEO's Admission and Triage information systems. In the first few months of deployment Mirth had successfully processed over 100,000 HL7 messages, routing only messages that were relevant to our trial. This results further show that Mirth is a robust solution for the purpose of integrating a CDSS with existing HIS.

8 Implications of the Use of OSS for ICDSS Implementation

The proponents of the use of OSS point out the reduced cost, increased reliability, stability, auditability and flexibility, as well as existing support networks as beneficial implications of the use of these technologies. However, different projects may see one or more of these benefits realized in practice. In case of MET3 we observed:

- Reduction in development cost, as we did not pay for any of the libraries, components and systems used in developing MET3. Additional cost saving was realized through the speed by which MET3 was developed (7 person-months). This time included researching and experimenting with the relevant OSS, learning some of the libraries, and setting up software repositories and build processes among other things.
- Ability to correct or adapt existing source code as needed, which improved the stability and reliability of MET3. For instance, in a few occasions we needed to modify the behaviour of some of the GUI widgets. We also had to modify the behaviour of the Java bean generator plug-in, which came with Protégé.
- Benefiting from existing community support in doing the above modifications and finding answers to occasional questions about various technologies as the issues and their resolution were discussed by others who had experience with these software systems and libraries.
- Having additional flexibility, as for each OSS used in MET3 we had a choice among at least a few alternatives.

We did not have to audit the software we used for compliance to specific security requirements but it is conceivable for a healthcare provider to request such an audit.

While having multiple technologies to choose from gives certain amount of flexibility and assurance in developing software, it also entails additional effort to identify the most appropriate one. In an application such as a CDSS, which is expected to serve the end users for a long period of time, continuous support for the underlying technology is as important as having the desired features. For instance, during development of MET3 we needed to identify a library to create the mobile version of EPA. Initially we decided to use Thinlet, a technology used in one of our earlier projects that involved mobile devices such as PDAs. However, we soon found a better technology, namely WTK by Sun Microsystems. In fact one of the common reasons for the termination of an open source project is the appearance of alternative projects that provide a better solution. In some cases these alternative projects are extensions of the original technology or integrate it in a newer technology, which typically stay backward compatible. For instance, recently WTK has been integrated into Java ME SDK 3.0. However, if one does not want to maintain and adapt the code of a terminated OSS project, then upgrading to a newer technology may imply expenditure of additional funds. The difference with the commercial software products is that these upgrades are not forced by a vendor, rather are initiated by the end user.

Additional considerations when using OSS includes:

- Documentation is an important factor in deciding which open source technology to use. Better established projects tend to come with better documentation. OSS projects tend to rely on user community help, however this does not replace the need for proper documentation, which may be lacking in some cases.
- While the use of OSS software is becoming more popular, generally speaking it may be easier to fined trained and competent staff for some commercial software systems compared to their OSS counterparts.
- Liability of software provider is another consideration when using OSS or commercial software and may result in a decision in favour or against using such software.
- Some open source software have a tendency to generate numerous versions. Furthermore, the potential for generating additional versions through the process of forking the source code, and consequently creating separate evolution paths, may make managing the projects that rely on them more difficult [35].

In our experience when choosing the appropriate OSS for implementing a system one should look for software which is in widespread use, complies with well established standards, and is backed by industry.

9 Conclusion

In this paper we argued the need for creating an ICDSS. To demonstrate the viability of our proposal, we set out to implement MET3 ICDSS using the multi-agent architecture. MET3 integrates some of the most common tasks in a hospital setting: clinical data collection, diagnosis formulation, and treatment planning and evidence retrieval in support of the treatment. To provide further flexibility and convenience of use we designed and implemented the system so it can be used on a conventional desktop computer, tablet computer or modern mobile devices.

We advocated the use of OSS as a desirable alternative to proprietary closed source systems. Our experience shows that one can create an ICDSS with features mentioned above solely relying on open source and free software technologies. In other words the current state of the art allows one to create complex ICDSS with a fairly low budget. While our overall experience with the use of OSS in developing CDSS has been positive, one has to acknowledge the need for

additional initial investigation for selecting the best technology which ensures continuous and long term use of the system. Other factors such as documentation, liability, staff expertise and project forking may as well influence this decision.

MET3 was tested in a lab setting with EAA running on desktops and tablets as well as Nokia mobile phones. We simulated a heavy usage load based on admission statistics of a CHEO's ED department. Additionally, we successfully tested MET3's underlying HL7 integration technology as part of a clinical trial at CHEO.

Acknowledgments

The research presented here was supported by NSERC-CIHR program. The authors would like to thank the MET3 programming team of Tomasz Buchert, Bartosz Kukawka, and Tomasz Maciejewski.

References

[1] Patel, V.L., Kushniruk, A.W., Yang, S., Yale J.F., Impact of a computer-based patient record system on data collection, knowledge organization, and reasoning. *J Am Med Inform Assoc*, 7(6):569-85, 2000.

[2] Patel, V., Arocha, J.F., Zhang, J., Thinking and reasoning in medicine. In: Holyoak, K., Morrison, R, editors. *The Cambridge handbook of thinking and reasoning*. Cambridge: Cambridge University Press; 2005, p. 727-50.

[3] Field, M.J., Lohr, K.N., editors. *Guidelines for clinical practice: from development to use*. Washington, D.C.: National Academy Press; 1992.

[4] The Cochrane library. http://www.thecochranelibrary.com/

[5] Sackett, D.L., Rosenberg, W., Gray, J.M/, Haynes, R., Richardson, W., Evidence based medicine: what it is and what it isn't. *Br Med J*, 312:71-72, 1996.

[6] Berlin, A., Sorani, M., Sim, I., A taxonomic description of computer-based clinical decision support systems. *J Biomed Inform*,39(6):656-67, 2006.

[7] Sittig, D.F., Wright, A., Osheroff, J.A., Middleton, B., Teich, J.M., Ash, J.S., et al. Grand challenges in clinical decision support. *J Biomed Inform*, 41(2):387-92, 2008.

[8] Wilk, Sz., Michalowski, W., O'Sullivan, D., Farion, K., Matwin, S.. Engineering of a Clinical Decision Support Framework for the Point of Care Use. *AMIA Annu Symp Proc*, 6:814-818 , 2008.

[9] Farion , K., Michalowski, W., Wilk, Sz., O'Sullivan, D., Rubin, S., and Weiss, D., Clinical Decision Support System for Point of Care Use: Ontology Driven Design and Software Implementation. *Methods of Information in Medicine* 48(4): 381-390, 2009.

[10] Michalowski, W., Slowinski R., Wilk S., Farion K., Pike J., and Rubin S., Design and development of a mobile system for supporting emergency triage. *Methods Inf Med*. 44 (1): 14-24, 2005.

[11] Moreno A, Valls A, Riaño D. PalliaSys: agent-based proactive monitoring of palliative patients. *In proceedings of IWPAAMS*, Oct 20-21, León, Spain, 2005.

[12] Su, C., Wua, C. JADE implemented mobile multi-agent based, distributed information platform for pervasive health care monitoring. *Applied Soft Computing* (In press).

[13] Moreno, A., Isern, D., Sanchez D., Provision of agent-based health care services. *Journal of AI Communications.* 16(3): 167-178,, 2003.

[14] Hudson, D.L., and Cohen, M. E., Use of intelligent agents in the diagnosis of cardiac disorders. *Computers in Cardiology*, 633 – 636, 2002.

[15] Godo, L., Puyol-Gruart, J., Sabater, J., Torra, V., Barrufet, P., and Fàbregas, X., A multi-agent system approach for monitoring the prescription of restricted use antibiotics. *Journal of Artificial Intelligence in Medicine*, 27(3): 259-282, 2003.

[16] Hashmi, Z.I., Abidi, S. S. R., and Cheah, Y. An intelligent agent-based knowledge broker for enterprise-wide healthcare knowledge procurement, *In Proceedings of 15th IEEE Symposium on Computer-Based Medical Systems*. Maribor, Slovenia, 173, 2002.

[17] Bloodsworth, P. and Greenwood, S. 2005. COSMOA an ontology-centric multi-agent system for co-ordinating medical responses to large-scale disasters. *AI Commun.* 18(3):229-240, 2005.
[18] Cruz-Correia, R., Vieira-Marques, P., Costa, P., Ferreira, A., Oliveira-Palhares, E., Araújo, F., and Costa-Pereira, A. Integration of hospital data using agent technologies - A case study. *AI Commun.* 18(3):191-200, 2005.
[19] Hogarth, M.A. and Turner, S., A Study of Clinically Related Open Source Software Projects. *Proceedings of AMIA Symposium*, 330-334, 2005.
[20] Wright, A, Sittig, D.F., A four-phase model of the evolution of clinical decision support architectures. *Int J Med Inform*. 77(10):641-9, 2008.
[21] Chu, S.C, From component-based to service oriented software architecture for healthcare. In Proceedings of 7th International Workshop on Enterprise Networking and Computing in Healthcare Industry,. *HEALTHCOM*, 96-100, 2005.
[22] Weiss G. *A modern approach to distributed artificial intelligence*. MIT Press; 1999.
[23] Payne, T.R., Web services from an agent perspective. *IEEE Intelligent Systems*. 23(2):12-4, 2008.
[24] Garcia-Ojeda, J.C., DeLoach, S.A., Robby, Oyenan, W.H., Valenzuela, J., O-MaSE: a customizable approach to developing multi-agent development processes. In: Luck M, editor. Agent-oriented software engineering VIII*: the 8th International Workshop on Agent Oriented Software Engineering (AOSE*) , Berlin: Springer-Verlag.1-15, 2008.
[25] Woodridge, M, Jennings, N. R., and Kinny D., The Gaia Methodology for Agent-Oriented Analysis and Design. *In Journal of Autonomous Agents and Multi-Agent Systems*. 3(3):285-312. 2000.
[26] Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J., TROPOS: An Agent-Oriented Software Development Methodology. *In Journal of Autonomous Agents and Multi-Agent Systems*. Kluwer Academic Publishers, May 2004.
[27] Iglesias, C.A., Garijo, M., Centeno-González, J., Velasco, J.R., Analysis and Design of Multiagent Systems Using MAS-Common KADS, *Proceedings of the 4th International Workshop on Intelligent Agents IV, Agent Theories, Architectures, and Languages, Lecture Notes In Computer Science*. Vol. 1365 Springer-Verlag London, UK, 313 – 327, 1997.
[28] Foster, I., Jennings, N. R. and Kesselman C., Brain meets Brawn: why Grid and agents need each other. *In Proceedings 3rd International Conference on Autonomous Agents and Multi-Agent System*s, New York, US, 2004.
[29] Kawamoto, K. , Lobach, D.F., Proposal for fulfilling strategic objectives of the U.S. roadmap for national action on decision support through a service-oriented architecture leveraging HL7 services. *J Am Med Inform Assoc*, 14(2): 146-155, 2007.
[30] Wilk, Sz., Slowinski, R., Michalowski, W., Greco, S., Supporting triage of children with abdominal pain in the emergency room. *Eur J Oper Res*, 160(3): 696-709, 2005.
[31] Michalowski, W., Wilk, Sz., Farion, K., Pike,. J., Rubin, S., Slowinski, R., Development of a decision algorithm to support emergency triage of scrotal pain and its implementation in the MET system. *INFOR* 43(4): 287-301, 2005.
[32] Farion, K., Michalowski, W., Wilk, Sz., O'Sullivan, D., Matwin, S., A tree-based decision model to support prediction of the severity of asthma exacerbations in children. *J Med Syst*, 2009 (in press).
[33] Farion, K., Michalowski, W., Rubin, S., Wilk, Sz., Corell, R., Gaboury, I., Prospective evaluation of the MET-AP system providing triage plans for acute pediatric abdominal pain. *Int J Med Inform,* 77(3): 208-218, 2008.
[34] Canadian Association of Emergency Physicians: Guidelines for Emergency Management of Paediatric Asthma. Available at http://www.caep.ca/
[35] Bellifemine, F.L., Caire, G, Greenwood, D, *Developing Multi-Agent Systems with JADE*. Wiley, 2004.
[36] Krishnamurthy, S. A Managerial Overview of Open Source Software, *Business Horizons*, 46(5), 47-56, September-October 2003.