Engineering of a Clinical Decision Support Framework for the Point of Care Use

Szymon Wilk, PhD[1], Wojtek Michalowski, PhD[1], Dympna O'Sullivan, PhD[1], Ken Farion, MD[2], Stan Matwin, PhD[1]
[1]University of Ottawa, Ottawa, Canada;
[2]Children's Hospital of Eastern Ontario, Ottawa, Canada

Abstract

Computerized decision support for use at the point of care has to be comprehensive. It means that clinical information stored in electronic health records needs to be integrated with various forms of clinical knowledge (elicited from experts, discovered from data or summarized in systematic reviews of clinical trials). In order to provide such comprehensive support we created the MET-A[3]Support framework for constructing clinical applications aimed at various medical conditions. We employed the multiagent system paradigm and the O-MaSE methodology to define an engineering process involving three main activities: requirements engineering, analysis and design. Then we applied the process to build MET-A[3]Support. The paper describes the engineering process and its results, including models representing selected elements of our framework.

Introduction

Most research in health informatics is fragmented and devoted to multiple distinct areas such as electronic health records (EHR), stand-alone clinical decision support systems (CDSS), computerized clinical practice guidelines and digital repositories of clinical evidence to name the few. Relatively little effort has been dedicated to integrative approaches encompassing the above areas so that comprehensive decision support is available at the point of care.

We consider comprehensive decision support as the provision of clinical information integrated with clinical knowledge. Clinical information includes patient data stored in EHR and clinical knowledge covers knowledge elicited from experts (presented usually as clinical guidelines), knowledge induced automatically from historical data using knowledge discovery techniques (presented in the form of decision models), and clinical evidence that summarizes results of randomized clinical trials. Patient data is integrated with guidelines and decision models to provide patient-specific suggestions. Moreover, patient data controls the search for evidence, so that retrieved results relate to a given patient.

In order to achieve such integration and to be useful in clinical practice, decision support has to be computerized, it has to be aligned with workflow, and it has to be available in the most appropriate form[1]. Building on our earlier research on the Mobile Emergency Triage (MET) system[2] that supports emergency triage of pediatric acute conditions, we moved towards creating a comprehensive decision support framework labeled MET-A[3]Support (**A**nytime **a**nd **A**nywhere support). This general framework can be used to build specific clinical applications offering support for various medical conditions.

The clinical applications developed for MET-A[3]Support are intended to be used by two groups of users – physicians and nurses. Both groups play varying roles in the patient management process, thus the functionality provided by the framework needs to be diversified. Considering that the functionality provided for a physician is a superset of that provided for a nurse, in the paper we focus on the first group of users only.

The creation of a comprehensive support framework required the consideration of several issues. Clinical information and knowledge is normally distributed among EHR, other hospital systems and external repositories (e.g., the Cochrane Library). Moreover, the main requirement – provision of comprehensive decision support – has to be further decomposed. Finally, the flow of information among different interacting entities has to be coordinated and available resources need to be shared (physically and logically). All these factors are often mentioned as premises for using the multiagent system (MAS) paradigm[3] for the engineering of clinical systems and we employed this paradigm while creating the MET-A[3]Support framework.

A MAS is a collection of relatively autonomous entities called "agents". Agents exchange and share information in order to achieve an overall goal that is too complex for any single agent to accomplish[4]. During these interactions they react to changes in their environment in a proactive, autonomous and intelligent manner.

The agents in MET-A^{3}Support are less sophisticated in terms of planning their actions and organization of their activities because the user (physician) acts as a controller defining the overall goal to be achieved and specific agents are employed to perform specific tasks moving physician towards this goal. Thus, agents in MET-A^{3}Support are reactive and user-controlled – their autonomous and proactive behavior is limited.

There are several methodologies for engineering MAS and they include Gaia, MAS-CommonKADS and O-MaSE[5, 6]. We decided to use O-MaSE because it has several important advantages. It is a flexible methodology that allows customizing of the engineering process, does not impose requirements for the intelligence of the agents, and finally it uses UML and AUML (Agent UML) notations in describing models, which improves their readability.

The paper is organized as follows. First we briefly describe the O-MaSE methodology. Then we introduce the O-MaSE process for MET-A^{3}Support and present selected models constructed using this process. Finally, we conclude with a discussion.

O-MaSE Methology

O-MaSE (Organization-based Multi-agent System Engineering) is a process framework that allows for custom engineering processes for MAS[6]. It can be viewed as an abstraction of the object-oriented paradigm where agents are specialized objects[5]. Such a view is especially appealing as it allows handling entities of varying intelligence within the same framework.

O-MaSE assumes that an engineered MAS is an organization of agents. The organization has a specific purpose that defines its overall goal. The overall goal is decomposed into subgoals, the achievement of which requires the existence of specific roles. The roles are played by the agents and the assignment of roles to agents can be either static (during design time) or dynamic (during run time). In order to communicate agents use a common language describing the environment in which they act and represented in the form of a domain model. Finally, the behavior of the organization and the agents is regulated by policies.

Instead of proposing a single-size-fits-all engineering process, O-MaSE provides a set of tools and guidelines specifying how to combine tasks together depending on the requirements for the specific MAS. This allows constructing customized processes that are suited to the characteristic of a considered system.

The tasks are grouped into three categories that correspond to three main activities in an O-MaSE process: *requirements engineering*, *analysis*, and *design*. *Requirements engineering* is aimed at translating requirements into goals. *Analysis* is aimed at defining an organization model, possible roles and their interactions, and a domain model. Finally, *design* is aimed at defining agent classes, protocols used by agents to communicate and plans agents follow in order to accomplish specific goals. For simpler systems, where agents are reactive and the assignment of roles (and thus goals) to agents is static, a basic O-MaSE process is sufficient[6]. The basic process includes the following tasks: goals modeling and refining, domain modeling, agent classes modeling, protocols modeling and plans modeling. For complex MAS where the assignment of goals to agents is dynamic, the basic O-MaSE has to be expanded by additional tasks: organization modeling, roles modeling and policies modeling.

Engineering of MET-A^{3}Support

O-MaSE Process

The MET-A^{3}Support framework is composed of agents that have goals assigned during the design stage, so the basic O-MaSE process is sufficient. Because of limited space we are not able to describe here the domain modeling task that involved building an ontology defining important clinical concepts and relations between them (detailed information is available elsewhere[2]). Before developing the ontology we examined already available models of clinical information (e.g., HL7 RIM). Then we constructed the new model that employed some existing ideas (e.g., the concept of the patient encounter). Building the customized domain model allowed us to focus on aspects relevant to the MET-A^{3}Support and to avoid unnecessary complexity. All agents share the constructed model, thus it acts as a "common language" to facilitate information exchange.

The process starts with requirements engineering where requirements are transformed to a goal model (Figure 1) by the goals modeling and refining task. As we decided to exclude the domain modeling task from the description, there is no analysis activity and design follows immediately after requirements engineering. The first task in the design activity – agent classes modeling – uses the goal model to produce an agent class model (Figure 2). Although O-MaSE advocates defining agent classes in terms of played roles, they can be also defined directly in terms of achieved goals. The agent class model not only defines agent classes, but it also indicates external actors and identifies protocols corresponding to interactions between agents, and between agents and actors.

The agent class model is then used by the protocols modeling task to build protocol models. Protocol

models define the details of protocols identified in the agent class model in terms of messages exchanged between agents, or agents and external actors. The agent class model and the protocol models are further processed in the plans modeling task to create plan models (Figure 3) that describe algorithms used by agents to achieve a specific goal or a set of goals.

Requirements

Initial input information required by the O-MaSE process is a set of functional and non-functional requirements that have to be met by the engineered system. In the description below we focus on the former as they correspond to the provision of comprehensive decision support. Non-functional requirements addressing issues of security and performance have been included in the original engineering process, however, their description is beyond the scope of this paper.

MET-A^{3}Support (and derived clinical applications) has to aid the user in managing patient encounters and to support informed decision making. Supporting an encounter involves data collection and entry, evaluating the patient and prescribing appropriate treatment. Therefore, MET-A^{3}Support should provide structured data entry as well as sophisticated support involving such modalities as patient evaluation advice, treatment suggestions and provision of clinical evidence. The support modalities should be optional and the user should be able to make own decision without invoking them.

MET-A^{3}Support should also manage clinical information stored in records of currently handled patients and synchronize it with hospital information systems (HIS). We assume HIS include EHR and other systems (e.g., laboratory systems). Whenever a user modifies a record, the system should notify HIS to maintain data consistency. On the other hand, MET-A^{3}Support should properly respond to events reported by HIS (e.g., availability of new laboratory results) and modify its repository of records.

Models

Goal model. The goal model for MET-A^{3}Support is based on the functional requirements described above and it is presented as a goal tree in Figure 1. The top goal – *provide comprehensive decision support* – is decomposed into 3 subgoals corresponding to the main functional requirements specified for MET-A^{3}Support (managing patient data and synchronizing it with HIS, managing and supporting encounters and providing suggestions and evidence on request). These goals are refined into lower level subgoals that further explain the parent goals. The goal model supports two types of refinement – AND and OR. AND indicates a conjunction of subgoals (all child goals have to be satisfied in order to satisfy a parent goal), and OR indicates a disjunction of subgoals (a parent goal is satisfied if at least one of the child goals are satisfied). In the MET-A^{3}Support the *provide suggestions and evidence* goal is refined into subgoals with OR (the user does not have to use all available support modalities), and all the remaining goals are refined with AND.

The goals in the goal model may be parameterized to fully specify their purpose. For example, *suggest treatment* has one parameter of the *PatientRecord* type that indicates the record of an evaluated patient; used

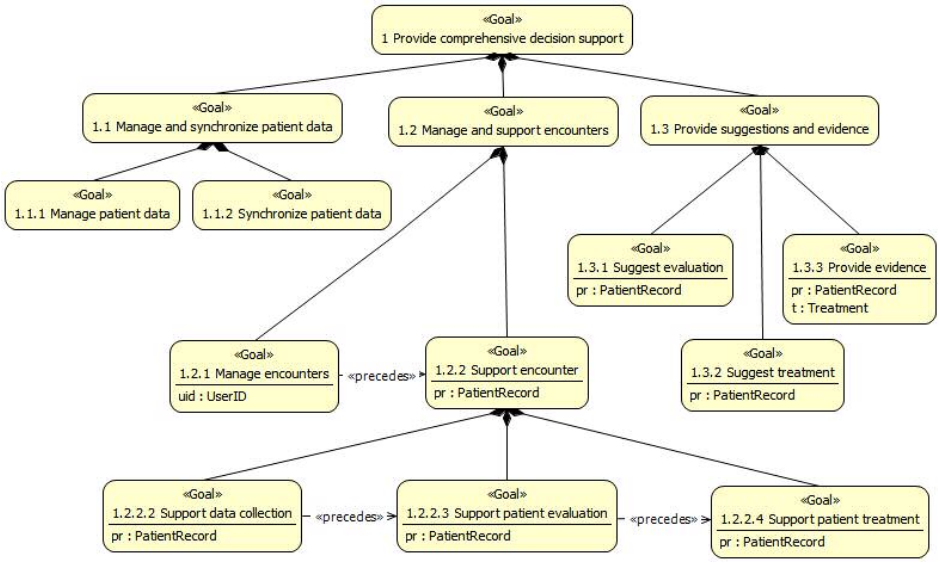

Figure 1. Goal model

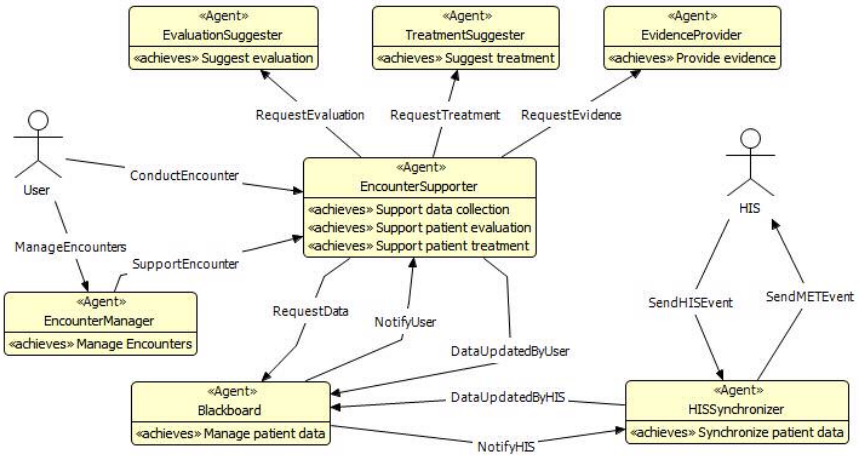

Figure 2. Agent class model

to establish a suggested treatment.

The goal model allows introducing a precedence relation between goals indicating that one goal has to be satisfied before another goal is pursued. For example, *support patient evaluation* precedes *support patient treatment* – a patient has to be evaluated before treatment is decided and prescribed.

Agent class model. The agent class model is presented in Figure 2. Each agent class is linked with at least one leaf goal (i.e., a goal with no subgoals) from the goal model and on the other hand each leaf goal is assigned to some agent class.

In the goal model we identified 9 leaf goals that are linked with 7 agent classes. The *EncounterSupporter* agent class is assigned 3 goals – all of them correspond to supporting an encounter, therefore, they are grouped together. There are also agent classes that are linked with single (and specific) goals. For example, *EvidenceProvider* is linked with a single goal – *provide evidence*. Achieving this goal is complicated, thus the agent class should not be overloaded with additional activities.

The agent class model also shows how agent classes interact. Specific interactions are represented as protocols – for each possible interaction the model indicates a separate protocol. For example, *Encounter-Supporter* interacts with *Blackboard* using two protocols: *RequestData* to retrieve required patient data and *DataUpdatedByUser* for data updates. Finally, the agent class model indicates two external actors that interact via protocols with the agent classes – *User* and *HIS.*

Protocol models. In the agent class model we identified 13 different protocols corresponding to interactions between agents and between agents and external actors. Each of these protocols requires its own model, so we need to create 13 protocol models. In MET-A^{3}Support all the protocols are very similar and rely on the request—reply pattern, so because of their simplicity and space limitations we do not include any examples.

Plan models. A plan model describes how a specific agent class achieves its linked goal or a set of goals. In the agent class model we defined 7 agent classes, so we need to have at least 7 plan models. Again, because of limited space we present only one model – Figure 4 presents the model of the *SupportTreatment* plan applied by *EncounterSupporter* to achieve the *support patient treatment* goal.

A plan model is represented as a finite state automaton. Each state, except start and end, may be associated with a set of actions that are executed in sequence after the state has been entered. Messages can be sent and received during state transitions. Moreover, it is possible to specify a guarding condition that determines whether the transition is enabled. For example, if in the *capture treatment* state the user wants to confirm the treatment, the transition to *confirm treatment* is enabled only if any treatment has been provided.

Conclusions

In this paper we described the engineering of the MET-A^{3}Support framework for providing comprehensive decision support using the MAS paradigm and the O-MaSE methodology. While O-MaSE proved to be a useful method for engineering MET-A^{3}Support, it is an example of a top-down approach with all associated limitations and benefits.

Due to space limitations we simplified the presentation of the engineering process – we focused on functional

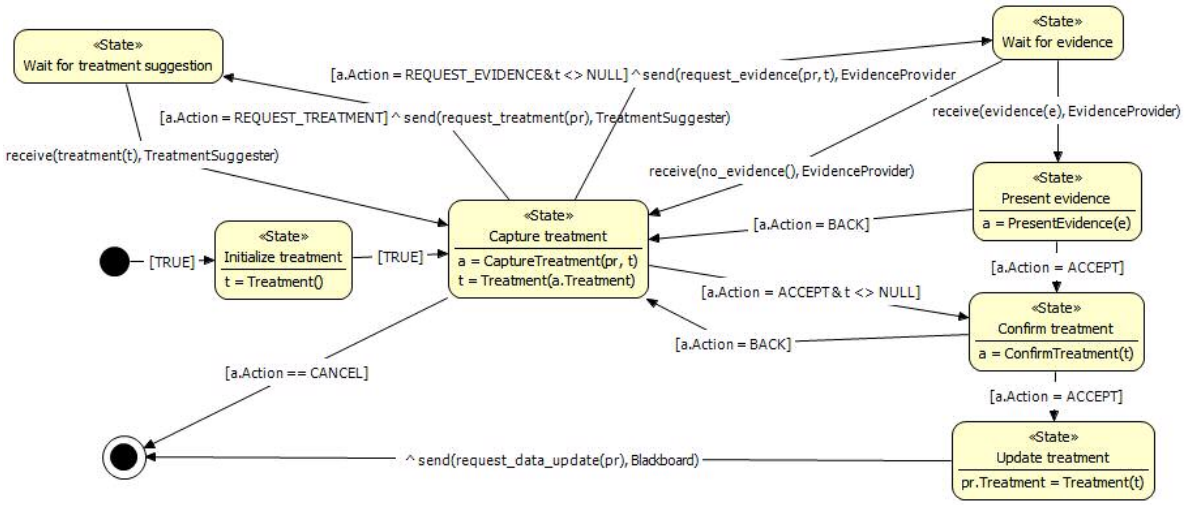

Figure 3. *SupportTreatment* plan model

requirements for MET-A^{3}Support and excluded the domain model. Nevertheless, we were able to demonstrate advantages of using O-MaSE for creating the comprehensive decision support framework. On the one hand, this methodology allowed us to customize the engineering process so that specific requirements were met. On the other hand, it imposed a well defined structure controlled by O-MaSE guidelines, thus the integrity and correctness of the process was preserved.

A novelty of the approach discussed here was to take the principles of O-MaSE methodology, adapt them to the comprehensive decision support requirements, and to create a support framework as defined by these principles. In that sense we showed how a customized structured process facilitated engineering of the MAS that involved agents of different levels of intelligence, autonomy and task complexity.

The first clinical application created for MET-A^{3}Support will be MET-A^{3}Support-Asthma for supporting emergency management of pediatric asthma exacerbations. We are currently working on system's prototype that is being developed using JADE (Java Agent Development Framework). We are also using Protégé to maintain and store the domain ontological model. We have already completed the first version of the *EvidenceProvider* agent and now we are focused on *EvaluationSuggester*. In comparison with more traditional methods of development (e.g., those used in creating the first generation of the MET system), the current approach allows for extensive modularity on all levels of the development process. New agents may be easily updated and added to a running application, what significantly eases testing and deployment.

Acknowledgment

Research described in this paper was supported by the NSERC-CIHR grant.

References

1. Kawamoto K, Houlihan CA, Balas EA, Lobach DF. Improving clinical practice using clinical decision support systems: a systematic review of trials to identify features critical to success. BMJ 2005;330(7494):765-772.
2. Michalowski W, Slowinski R, Wilk S, Farion K, Pike J, Rubin S. Design and development of a mobile system for supporting emergency triage. Methods Inf Med 2005;44(1):14-24.
3. Moreno A. On the evolution of applying agent technology to healthcare. IEEE Intel Syst 2006;21(6):8-10.
4. Weiss G. A modern approach to distributed artificial intelligence: MIT Press; 1999.
5. DeLoach SA, Wood MF, Sparkman CH. Multiagent systems engineering. Int J Softw Eng Know 2001;11(3):231-258.
6. Garcia-Ojeda JC, DeLoach SA, Robby, Oyenan WH, Valenzuela J. O-MaSE: a customizable approach to developing multiagent development processes. In: Luck M, editor. Agent-oriented software engineering VIII: the 8th International Workshop on Agent Oriented Software Engineering (AOSE 2007). Berlin: Springer-Verlag; 2008. p. 1-15.