2nd International Workshop on Model-Based Design of Trustworthy Health Information Systems, 2008

A Model-Integrated Approach to Implementing Individualized Patient Care Plans Based on Guideline-Driven Clinical Decision Support and Process Management - A Progress Report

Jason B. Martin, MD[3], Janos L. Mathe[1], Peter Miller[2], Akos Ledeczi, PhD[1], Liza Weavind, MD[3], Anne Miller, PhD[3], David J. Maron, MD[2,3], Andras Nadas[1], Janos Sztipanovits, PhD[1]

[1] Institute for Software Integrated Systems
[2] Vanderbilt HealthTech Laboratory
[3] Vanderbilt University Medical Center
Vanderbilt University

Abstract. Standardizing the care of patients with complex problems in hospital settings is a challenge for physicians, nurses and other medical professionals. In acute care settings such as intensive care units, the inherent problems of stabilizing and improving vital patient parameters is complicated by the division of responsibilities among different individuals and teams. The use of evidence-based guidelines for managing complex clinical problems has become the standard of practice. Computerized support for implementing such guidelines has tremendous potential. The use of model-based techniques for specifying and implementing guidelines as coordinated asynchronous processes is a promising new methodology for providing advanced clinical decision support. Combined with visual dashboards, which show the status of the implemented guidelines, a new approach to computer-supported care is possible. These techniques are being applied to the management of sepsis in acute care settings at Vanderbilt Medical Center.

Introduction

Formalization of medical knowledge has been an active area of research since the 1960s. Early efforts were focused on creating systems that mapped signs, symptoms and laboratory results to probabilistic estimates of different diagnoses [1][2][3]. These systems, embodied as expert systems, proved not to be practical for the everyday practice of medicine. Only with the development of the electronic medical record (EMR) have knowledge-based systems proven to be practical and been adopted by practitioners [4][5].

Medical knowledge-based systems today focus on computerized physician order entry (CPOE) and clinical decision support advisory systems [6][7][8]. CPOE systems depend on comprehensive EMRs to provide means to physicians and nurses to create and execute orders for tests, procedures and medications. A system such as WizOrder, developed at Vanderbilt

University Medical Center (VUMC) and sold commercially as Horizon Expert Orders by McKesson Corporation, contains multiple advisors that help physicians with issues such as identifying potential adverse drug interactions or determining which combination of medicines might be best for a particular patient [9][10]. CPOE and related systems are often termed 'physician workflow' systems because they are designed to fit the normative matrix of activities that flow from specific surrounding systems and the standardized practice of medicine.

Another area being actively explored is the use of computer-generated alerts. By utilizing rule engines through publish/subscribe models to actively monitor the patient's real time status, they are looking for specific problems which should trigger an alert [11].

The next area of application of knowledge-based systems is process management. VUMC is pioneering the use of process management 'dashboards' to inform medical staff of the status of required activities to be performed for patients with specific problems. This has been applied to the management of ventilator acquired pneumonia (VAP), a serious consequence of a patient's intubation and mechanical ventilation. The 'bundle' of activities required to minimize the development of VAP was created, and the status of these activities is shown using red, yellow and green indictors. These are made available to the hospital staff as reminders of what has been and what needs to be done.

The overall management of a complex medical process requires a formal representation of treatment protocols in order to be able to show the temporal structure and coordination of the tasks and the history of measurements that demonstrate status, trends and rates of change. The key insight in our work is *Model-Integrated Computing (MIC)* [12][13][14], an approach and its supporting tool suite for model-based software and systems engineering that has been developed over the last two decades at Vanderbilt. This infrastructure offers new opportunities in creating clinical decision support and process management systems. MIC focuses on formal representation, composition, and manipulation of integrated models of information processes and security/safety policies, and provides tools for automated system generation directly from the models. The open-source MIC tool suite [15][16][17] addresses layered, multiple-view system modeling, model transformation, model analysis and validation, execution, and design evolution. The application of MIC principles and tools casts the creation of clinical decision support and process management systems in the following framework:

1. *Design of modeling language for treatment protocols.* In MIC, modeling languages are formally defined by metamodels [15][17]. The MIC metaprogrammable tools for modeling, model management and model transformation are automatically customized by the metamodels.
2. *Modeling treatment protocols*. Using the modeling language defined in step 1, models of specific treatment protocols are created. These models are formal representation of guidelines that drive the management of clinical processes. The precise semantic foundation of the MIC modeling infrastructure and related tools enable validation and verification of the models against a range of safety, privacy and security related criteria defined as constraints or policies.
3. *Generation of process management systems*. Using the MIC model transformation infrastructure, the verified models are translated into configuration files that customize the generic run-time components (such as execution engine, Graphical User Interface and EMR Interface) of the process management system.

The components of this framework are consistent with the Model-Integrated Clinical Information System (MICIS) infrastructure [18], which is a generic tool suite for designing, testing and deploying clinical information systems. The goal of this paper is to show the use of the framework in creating a Sepsis Treatment Enhanced through Electronic Protocolization (STEEP) application. Sepsis management is a complex and extremely information intensive process performed in intensive care units and emergency departments. Application of guidelines that can evolve with accumulated experience and can be customized to the needs of individual patients has huge significance, which makes sepsis management an attractive application target for MICIS. Since the overall effort is complex, we restrict our discussion to the central issues in the model-integrated development approach: modeling language and model specification, model validation and verification and the automated system generation process.

Sepsis Management Problem

In an effort to maximize the impact of our process management tool, we sought a universal clinical paradigm that was common, expensive (both in terms of hospital resources and financial expenditures), and has accepted evidence-based treatment guidelines. We found sepsis to be an ideal candidate for our intervention. The sepsis syndrome results from a robust host reaction to infection and is characterized by a systemic inflammatory response, frequently with hypotension and multiple organ failure. This disease process is very common, occurs with a worldwide distribution, and can impact patients of any sex, race, or age. About 750,000 cases occur in the United States annually [19], and about 30% of septic patients will die from the disease [20]. Severely septic patients consume many hospital resources, requiring on average 7-10 days in the intensive care unit and up to 3-5 weeks total hospital length of stay. In the United States, patients may accrue hospital charges of tens of thousands of dollars, and it is estimated that sepsis-related expenditures approach $17B in the United States annually [21].

Given the large scope of this clinical problem, it is not surprising that many treatment strategies have been proposed and investigated. The Surviving Sepsis Campaign (SSC), led by experts from numerous professional organizations, seeks to improve the diagnosis, management, and clinical outcomes in sepsis. The SSC has published a comprehensive set of treatment guidelines based on graded clinical evidence that are widely considered to represent the state of the art in sepsis management [22].

The SSC guidelines are complex and require multiple time-sensitive interventions based on dynamic patient variables. In clinical practice, the treatment guidelines are often grouped into "bundles" based on their ideal implementation time. For example, certain interventions are targeted for completion within 6 hours of diagnosis, including obtaining appropriate cultures, administering broad empirical antibiotics, and optimizing hemodynamics with early goal directed therapy in patients with septic shock. Other priorities, such as deep venous thrombosis and stress ulcer prophylaxis, are less time-sensitive but ideally completed within 24 hours. Therefore, correct and timely implementation of the guidelines requires continuous assimilation and interpretation of numerous pieces of patient data.

In the intensive care unit (ICU), the healthcare team must respond in a timely manner to the needs of many patients with a diverse array of clinical problems. Managing the massive flow of critical clinical information is challenging and may impede excellent care. For this reason, the ICU is an excellent test bed for information technology (IT) interventions. Such IT interventions can be categorized generally (in order of increasing sophistication) as clinical reminders, clinical pathways, or real-time protocolized decision support tools. Clinical reminders are “pop-ups,” or similar passive cues, activated on every patient and intended to remind a provider of a universal intervention. For example, clinical reminders are deployed frequently to ensure that physicians remember to order deep venous thrombosis prophylaxis, a measure required for most inpatients. Clinical pathways are lists of preferred interventions in patients with a specific disease. For example, in a patient with hyperglycemia and diabetic ketoacidosis, a physician may activate a treatment pathway and choose IV fluids, insulin, and other interventions from a list of therapies commonly applied during the treatment process of the disease. The most sophisticated IT interventions are real-time clinical advisory tools. Such tools continuously monitor specific patient variables, and based on clinical guidelines, provide treatment recommendations if an unmet clinical need is detected. There are few examples of such sophisticated IT interventions in ICU medicine currently. To our knowledge, the current effort is the most comprehensive attempt at managing sepsis though a sophisticated electronic detection and management tool.

We anticipate that the STEEP application will 1) decrease time to detection of patients with developing sepsis, 2) improve physician compliance with evidence-based standards as described in the SSC, and 3) result in improved clinical outcomes for patients (ICU and total inpatient length of stay, number of organ system failures and mortality rate).

The STEEP tool will monitor real-time patient data streams and, using specific laboratory and vital signs criteria, will identify patients with possible sepsis. The lab and vital sign abnormalities are quite sensitive for the diagnosis of sepsis, but lack specificity without clinical input and contextual interpretation. Therefore, these patients with “alert status” will be identified to the healthcare team for further review. The patients are identified first by a visual cue on the ICU dashboard; if this visual alert is not addressed in a timely manner, an electronic notification via text page will be sent to appropriate team members. When responding to the sepsis alert, physicians will be presented with an intuitive, visually rich, and educational explanation of why the sepsis alert was activated, and they will be offered the opportunity to activate decision support if there is a reasonable suspicion that the abnormal physiological parameters are due to infection. If the physician activates decision support, the tool will assess various patient parameters and provide customized decision support recommendations.

Overview of the System Architecture

On the highest level, MICIS-STEEP has two architectural views: the Modeling and Generation view and the Operation view.

Modeling and Generation Architecture

The Modeling and Generation architecture is shown in Figure 1. The STEEP application includes the model-based *Execution Engine* and two graphical user interfaces (GUIs), the *Sepsis Management GUI* and the *Supervisor GUI*. The *Execution Engine* runs the sepsis management process according to the specification in the *Derived Protocol Representation (XML)* file.

The *Protocol Models* containing the formally specified treatment protocols are designed by physicians using the Generic Modeling Environment (GME) [17]. The GME tool is a metaprogrammable graphical model builder; it can be customized to the designed protocol modeling language by defining its metamodel. The *Protocol Models* built with the help of the GME tools are transformed into the *Derived Protocol Representation (XML)* files used by the *Execution Engine* during operation.

Physicians use the *Sepsis Management GUI* to assess the treated patient's health status, to make decisions based on the evidence-based guidelines present on the screen and to actuate their decisions. Protocol Models are validated using Simulation. The *Simulation Supervisor* controls the simulation of a patient's treatment using the *Supervisor GUI*. The supervisor controls the environment which includes the patient's response to treatment and the behavior of the other simulated players, which include nurses administering drugs and laboratories delivering the lab results. *Sample data* for simulated execution of protocols are collected in spreadsheets and translated into XML files that are accessed by the *Execution Engine*.

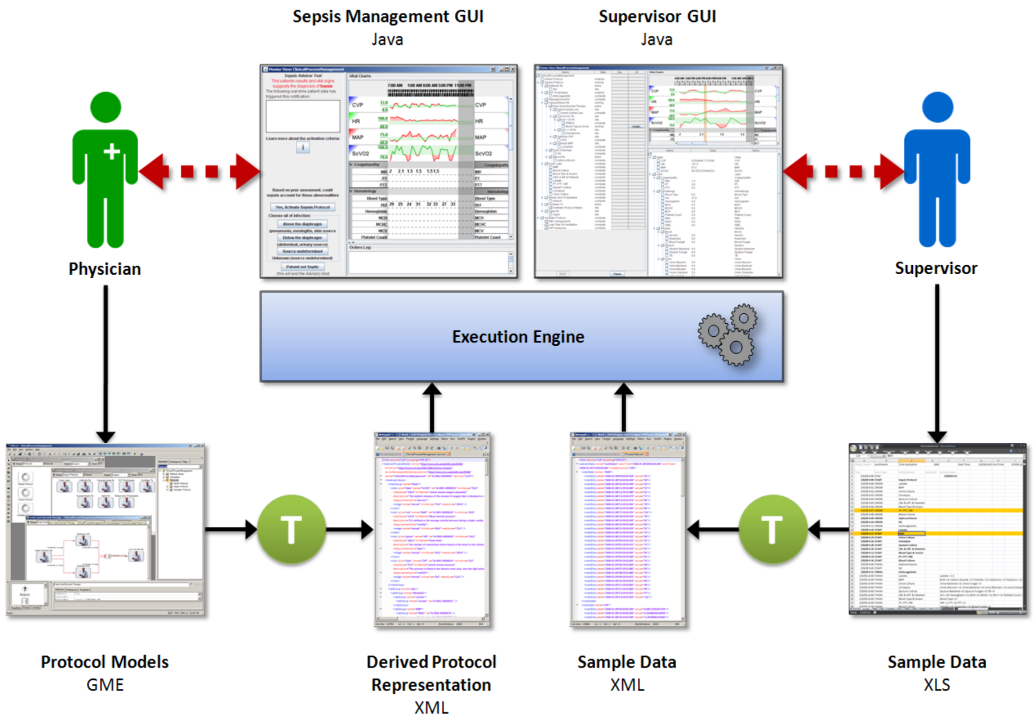


Figure 1 - Modeling and Generation Architecture

Operation Architecture

The Operation Architecture in Figure 2 shows the interactions among the *Physician*, STEEP, and the related components of the clinical information system.

The interaction between the Physician and the STEEP is facilitated by the Sepsis Management GUI by means of two panels: the Monitoring Panel and the Advisory Panel. The Monitoring Panel presents a timeline where categorized patient health information can be viewed in time in context with the actions of the therapy provided to the patient. Displaying cause and effect relations involves linking patient data and treatments so that the effect of one on the other can be seen; this is what we refer to as the action-reaction concept. The timeline runs from the past, when the treatment started, to the current time. Health indicators, fed to the system as a stream of data, include vital signs, such as temperature, blood pressure, heart rate and central venous pressure, etc. Laboratory test results, like the white blood cell count, are updated on the screen when the information becomes available. The panel also shows the actions of the treatment that were provided or are scheduled to be provided to the patient (e.g. the start of a normal saline (NS) treatment). All displayed data is temporally aligned in the same columns.

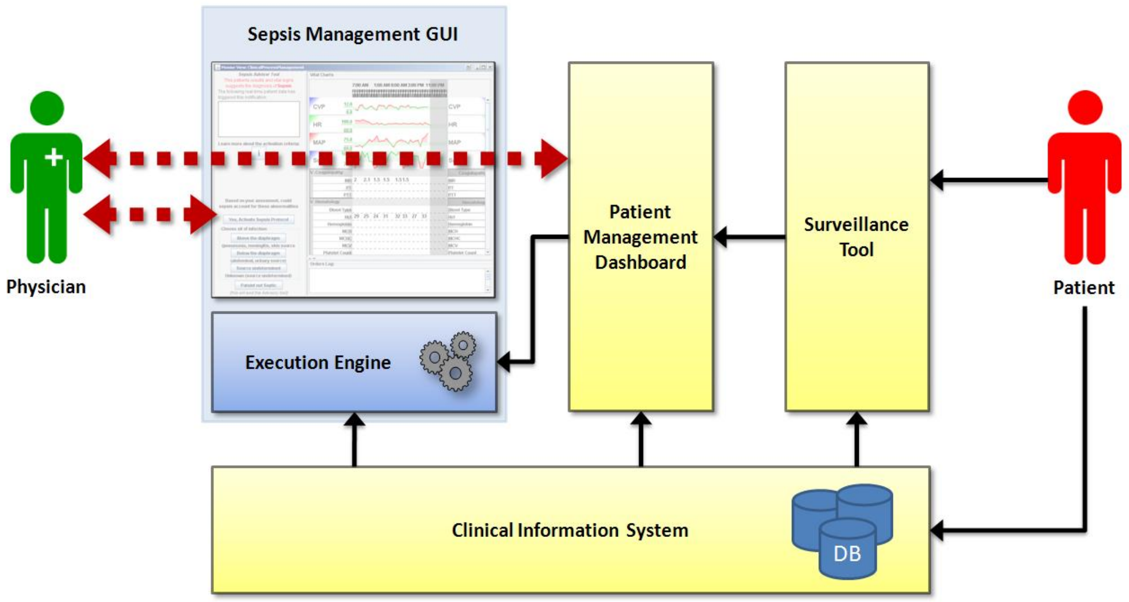


Figure 2 - Operation Architecture

Once physicians have assessed the patient information they must actively make a decision that the patient does or does not have sepsis. This decision is made in the Advisory Panel, which is the main view for performing the protocol-based treatment of the patient. Once the physician has assessed the patient data (visible on the Monitoring Panel) and any other information (e.g. patient history, admission notes, and physical exams), the physician then makes a formal diagnosis by using the built-in logic and the available action controls on the Advisory Panel. These actions include higher level control (e.g. selecting the sepsis severity level) as well as lower level controls (e.g. ordering of specific medications and procedures).

Design of the Clinical Process Management Language (CPML)

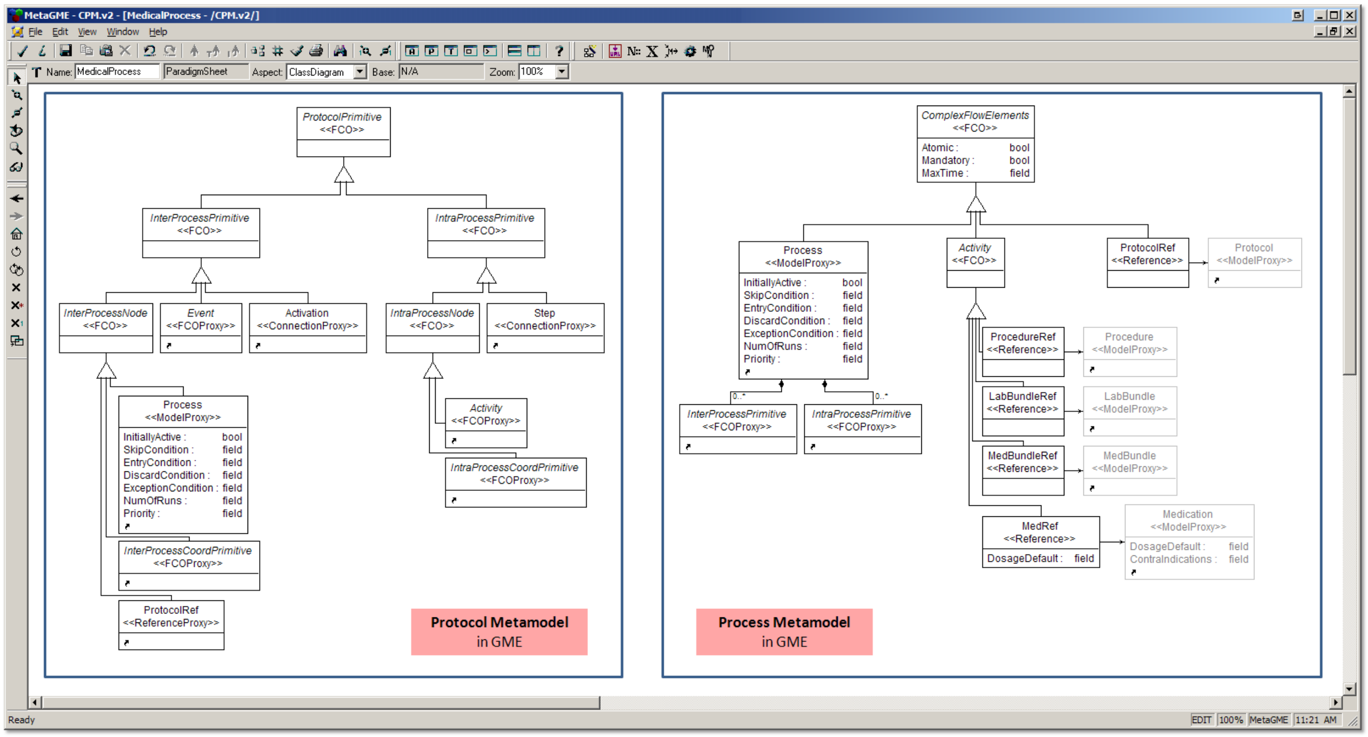
CPML is a domain specific modeling langue (DSML) designed for representing treatment protocols. Specification of DSMLs requires the specification of their abstract syntax, concrete syntax, semantic domain and the mapping between the abstract and concrete syntax (syntactic mapping) and the abstract syntax and the semantic domain (semantic mapping) [16]. The formal representations of these specifications are the meta-models and the language we use for describing meta-models is the meta-language. In MIC, the meta-language for representing the abstract syntax of DSML-s and the syntactic mapping is based on UML class diagrams (with stereotypes) and the Object Constraint Language (OCL) [23]. The abstract syntax defines the *concepts, relationships,* and *integrity constraints* available in the DSML. Thus, the abstract syntax determines all the (syntactically) correct "sentences" (domain models) that can be built. In MIC, the formal representation of the semantic mapping is done by using graph re-writing rules [15][24].

The precise specification of CPML has proved to be a hard problem due to the following issues. First, operational protocols, policies and treatment guidelines of healthcare organizations are rarely ever phrased in mathematically sound, unambiguous manner, which makes the design of a formal modeling language difficult. Second, the protocols that describe the medical processes constituting a treatment, their triggering conditions and their coordination methods need to be considered as guidelines, and not rigid workflows that must be enacted always the same way. This requirement is essential for the design of the execution semantics of models.

Due to these challenges, the language development took several iterations. In our first attempt, the language explicitly represented treatment trajectories as a connected, directed, bipartite graph structure. The nodes were either *decision points* with predefined multiple possible outcomes or *actions* representing treatment steps. The advantages of this approach were that it followed the formalization efforts presented in the available medical literature [25] and that it was simple enough. However, this approach did not prove to be efficient for expressing complex treatments (like the one for described for sepsis) because it was not scaling well due to the exponentially large number of potential trajectories generated by the many concurrent and interacting treatment processes.

The following iteration of the CPML approached the problem from a new direction: treatment steps were grouped together under the concept of *processes*. Processes are concurrent, asynchronous and can interact with each other via *events*. In order to capture the decision logic concisely, processes can be organized in a hierarchical manner. Processes listen to events happening around them and will only start running if their triggering conditions are satisfied. Coordination of processes is done with the help of events (and related messages). The execution semantics of the selected process model corresponds to the well known Communicating Sequential Process (CSP) model [26]. The major advantages of the CSP approach is the possibility of using hierarchies and defining segments of a complex protocol independently from each other (processes compose in CSP). This semantics proved to be more intuitive to the physicians too, because it is closer to the way they tend to think of the different cases they deal with.

The CPML language is defined by the metamodel, which is the placeholder for the definitions of the various concepts we use to define a protocol. These concepts include the aforementioned abstractions: the process and the protocol, to which the precise models (defined in GME) are shown in Figure 3. The two highlighted boxes show the building blocks of a protocol and the building blocks of a process.



MetaGME - CPM.v2 - [MedicalProcess - /CPM.v2/]
File Edit View Window Help
Name: MedicalProcess
ParadigmSheet
Aspect: ClassDiagram
Base: N/A
Zoom: 100%
ProtocolPrimitive <<FCO>>
InterProcessPrimitive <<FCO>>
IntraProcessPrimitive <<FCO>>
InterProcessNode <<FCO>>
Event <<FCOProxy>>
Activation <<ConnectionProxy>>
IntraProcessNode <<FCO>>
Step <<ConnectionProxy>>
Process <<ModelProxy>>
InitiallyActive : bool
SkipCondition : field
EntryCondition : field
DiscardCondition : field
ExceptionCondition : field
NumOfRuns : field
Priority : field
Activity <<FCOProxy>>
IntraProcessCoordPrimitive <<FCOProxy>>
InterProcessCoordPrimitive <<FCOProxy>>
ProtocolRef <<ReferenceProxy>>
Protocol Metamodel in GME
ComplexFlowElements <<FCO>>
Atomic : bool
Mandatory : bool
MaxTime : field
Activity <<FCO>>
ProtocolRef <<Reference>>
Protocol <<ModelProxy>>
ProcedureRef <<Reference>>
Procedure <<ModelProxy>>
LabBundleRef <<Reference>>
LabBundle <<ModelProxy>>
MedBundleRef <<Reference>>
MedBundle <<ModelProxy>>
MedRef <<Reference>>
DosageDefault : field
Medication <<ModelProxy>>
DosageDefault : field
ContraIndications : field
InterProcessPrimitive <<FCOProxy>>
IntraProcessPrimitive <<FCOProxy>>
0..*
0..*
Process Metamodel in GME
Ready
EDIT 100% MetaGME 11:21 AM

Figure 3 - Segments of the CPML Metamodel (partial view)

To explain the metamodel in detail is not the focus of this paper however some of the major concepts, without which the protocol modeling language would not be complete, are listed and described in Table 1.

Table 1 - High-level concepts of CPML

Abstraction	Description
Protocol	Top level concept, in which medical protocols can be described.
Medical Library	Top level concept, which serves as the placeholder for hierarchically categorizing general medical knowledge on the three main information categories that are referred to in our protocols: Patient Vitals, Patient Labs and possible Medications.

Orderables	Top level concept for building a hierarchy of bundles from the elements defined in the *Medical Library*. *Orderables* are the items that *Activities* of a *Protocol* refer to and are specific to healthcare organizations.
Process	Coordinated group of activities used in *Protocols*. *Processes* help to decompose the treatment protocol and to categorize the treatment steps. They are concurrent, asynchronous and can interact with each other via Events.
Activity	*Activities* are the lowest level components of a *Protocol*. They are the items that a physician can order, including Lab Bundles, Medication Bundles, single Medications and Procedures.
Event	Component used in *Processes*. *Events* refer to the activation, starting and completion events of other components, such as Protocols, Processes and Activities. They help to create dependencies among the mentioned runnable components.
Step	Example of a coordination primitive. It is the connecting element with which the execution order of *Activities* within a *Process* can be specified.
Synchronizing Merge	Coordination primitive used together with the *Step* connection. It defines a synchronization point in between activities where multiple paths of the *Step* connection converge into one single one. This means that if more than one path is taken, synchronization of the active paths needs to take place.

The example model in Figure 4 describes two components of a complete sepsis protocol that were represented using CPML. The first open window in the figure shows the contents of a fairly simple process, called “Order Labs”, which initiates the ordering of laboratory tests (such as blood culture, etc.). It is a process with no entry condition and marked initially active, which means that it will start executing immediately after the protocol starts. Since there are no dependencies among the provided actions, (various laboratory tests), their execution will be initiated simultaneously. During the execution of the protocol this translates to the following behavior: after activating the sepsis protocol the physician receives reminders on the Sepsis Management GUI that the ordering of the listed laboratory tests is advised.

The second example, in the second open window in Figure 4, describes the coordination among the components of the “Early Goal-Directed Therapy” process (EGDT) [25]. The EGDT process contains subprocesses that get activated in the order specified by the activation arrows (from left to right) once EGDT starts executing. This activation mechanism has no control over the execution order of the processes, it just specifies when the components start to listen. Only in runtime will the execution order be determined, when the entry conditions for process can get evaluated (completely independently from the activations).

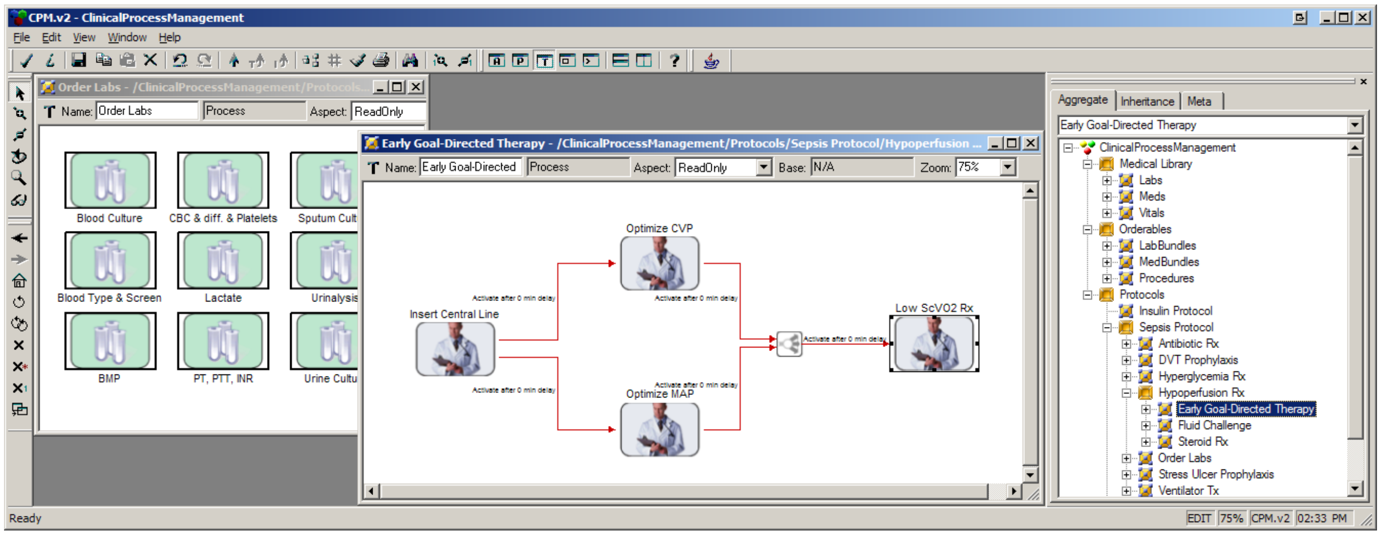


Figure 4 - Sepsis management models expressed using CPML (partial view)

Model Validation and Verification

With the help of the approach presented in the previous section we were able to represent treatment processes, operational policies and guidelines of Health Care Organizations as a set of well-formed and explicitly defined protocols. Also, as mentioned, by defining abstractions in CPML that are based on the basic concepts that healthcare professionals use in daily treatment we achieved that the formally defined protocols themselves can now be interpreted by the healthcare professionals. Treatment protocols, even if they serve as guidelines in the patient management, are safety critical and their validation and verification is an essential part of the protocol specification process. One of the key advantages of the MICIS approach is that the modeling languages are formally sound and provide foundation for disciplined validation and verification processes.

Validation

The role of protocol validation is to test if the generated decision support guidance corresponds to the clinicians' expectations. The first step of the validation is to model walk-through-s with clinicians. Expressiveness of the modeling language is an extremely important help in this process and fully confirms the importance of using DSML-s highly customized to the clinical environment. The second step of the validation is simulation based studies. As described earlier, the MICIS-STEEP architecture supports the generation of simulated execution. The simulation function includes Sample Data import, interface to the Execution Engine and a Supervisor GUI through which the simulated execution of the protocol can be controlled.

An essential part of the validation process is that it is needed to be conducted in a realistic environment, where ICU personnel can face near real-life situations of sepsis management and can interact with the STEEP system to make decisions. The validation process needs to be closely monitored and the results precisely evaluated. The infrastructure for this evaluation is provided by the Simulation Center of the Center for Experiential Learning and Assessment at Vanderbilt University Medical Center [27] (see Figure 5).

Figure 5 - Simulation Center at the Center for Experiential Learning and Assessment

Verification

Another benefit of using DSMLs is that the formal verification of the domain models against established criteria becomes possible. This is a significant step ahead, because in traditional approaches where the system is hardcoded (using programming languages, like Java, C, etc.), the model is not explicit and cannot be independently verified. Verification using our models can be performed on the following three levels:

1. *Static model verification*. This is the first line of defense, which is provided by the MICIS modeling tool, GME. As we described before, the metamodels of protocol modeling languages (such as CPML) include wellformedness rules that separate syntactically correct models (that can be translated into executable protocols) from incorrect models. The constraints are expressed using the Object Constraint language (OCL). In modeling time, GME (with the help of its constraint checker) enforces the wellformedness rules defined as constraints. These constraints include clinical limits for parameters as well as more sophisticated constraints that would be extremely hard to check without automated verification. An example for an OCL constraint is presented below:

   Treatment processes should not contain references to medical procedures that have not been defined yet or have been deleted from the available medical procedures library.

Formally:

```
let RefSet = self.referenceParts() in

let NotEmptyRefSet = RefSet->notEmpty() in

if NotEmptyRefSet then RefSet->forAll( not
refersTo().isNull() ) else true endif
```

Development of OCL constraints requires expertise in clinical constraints on protocols and in metamodeling. However, this is a crucial part of the development process and greatly contributes to the safety of protocol specifications.

2. *Verification of dynamic properties at design time*. Models are transformed into behaviors by the Execution Engine. In fact, protocols are instantiated into a complex, multi-threaded program that interacts with ICU personnel, patient data and events. Using a well defined, clean execution semantics (such as CSP) is crucial for verifiability of the models against a set of predefined behavioral properties such as *determinacy*, *livelock, deadlock* and others. At this point, we have developed a model translator to map the protocol models into an intermediate executable model – Stateflow [28]. The Stateflow models can drive a number of verification tools (model checkers, simulators, reachability analysis tools) that we plan to use in implementing our dynamic verification strategy. This is a planned activity in the next phase of our research.
3. *Run-time checking*. Critical actions that are performed during the treatment need to be checked at runtime. Security and privacy policies determine access rights to data published through the STEEP GUI and to the invocation of actions (initiating treatment processes, ordering medication, etc.). In the current implementation, STEEP access control is part of the general access control policies of the ICU, but we intend to make this customizable in later phases. Decisions present in the protocol allow various actions to be ordered by the healthcare professionals during treatment that are not only need to be logged, but they have to be matched against a set of legal regulations and the hospital's own policies. A number of these checks are performed by systems interfaced to STEEP, such as the WizOrder order management system that checks all medication related actions against a large suite of rules.

Continuation of this research in the area of validation and verification will allow the model-integrated approach to provide safe customization of individualized guideline-driven clinical decision support and process management systems. This is the primary area of our current research efforts and the motivation for the further development of the MICIS infrastructure.

Results and Ongoing Work

The project started in 2007, as a collaborative effort between the Vanderbilt School of Engineering and Vanderbilt University Medical Center to apply advanced model-based computing techniques to the management of complex clinical management processes which occur in acute care settings in hospitals. The initial results of that effort – MICIS – were reported at the MOTHIS 2007 Conference [29]. STEEP has been implemented as part of the MICIS framework. The team has completed much of the work on the first prototype of STEEP and is preparing to launch a clinical test of the system in late 2008. The completed work includes the completion of the modeling language, the runtime environment and the gathering of the testing data samples.

At present, the followings are our primary focus:

1. In order to be able to seamlessly integrate the STEEP application into the existing information systems architecture currently in place at the VUMC we plan to implement a *Surveillance Tool* (see Figure 2, also briefly mentioned in the Introduction). This tool will initiate the sepsis protocol by constantly monitoring the patients' health status and issuing alerts for the healthcare professionals if a predefined set of criteria is met.
2. Another challenge we are currently addressing is the development of a generic GUI that can be customized from protocol models. While we have solved this problem in case of the Simulation Panel the rest of the GUI elements (the Monitoring and the Advisory Panels) are still specific to STEEP and their configurability is limited. The solution will require the specification of a new aspect in CPML, which will allow the customization of the GUI elements. Otherwise all components of Figure 1 are completed and functional.
3. We are in the process of designing and performing a carefully coordinated, multi-phase experiment to evaluate the presented approach in terms of usability and effectiveness. The discussion of the evaluation plan is not in the scope of this paper.

Acknowledgments

This work was supported in part by TRUST (Team for Research in Ubiquitous Secure Technology), which receives support from the National Science Foundation (NSF award number CCF-0424422) and the following organizations: AFOSR (#FA9550-06-1-0244), BT, Cisco, ESCHER, HP, IBM, iCAST, Intel, Microsoft, ORNL, Pirelli, Qualcomm, Sun, Symantec, Telecom Italia, and United Technologies.

Support also came from the Vanderbilt HealthTech Laboratory, a research facility of Vanderbilt Medical Center, the Division of Allergy, Pulmonary and Critical Care Medicine in the Department of Medicine of Vanderbilt Medical Center, the Emergency Department of Vanderbilt University Hospital and the Informatics Center of Vanderbilt Medical Center.

References

[1] Bleich HL. Computer evaluation of acid-base disorders. J Clin Invest. 1969; 48:1689-96.

[2] Kassirer JP. A report card on computer-assisted diagnosis: the grade, C. N Engl J Med. 1994; 330:1824-5.

[3] Miller RA. Medical diagnostic decision support systems—past, present, and future: a threaded bibliography and brief commentary. J Am Med Inform Assoc. 1994; 1:8–27.

[4] Stead WW, Hammond WE. Computer-based medical records: The centerpiece of TMR. MD Comput. 1988; 8:48-62.

[5] Wilcox A, Hripcsak G. The role of domain knowledge in automating medical text report classification J Am Med Inform Assoc 2003; 10:330-338.

[6] Hunt DL, Haynes RB, Hanna SE, Smith K. Effects of computer-based clinical decision support systems on physician performance and patient outcomes: a systematic review. JAMA. 1998; 280:1339–46.

[7] Bates DW, Leape LL, Cullen DJ, et al. Effect of computerized physician order entry and a team intervention on prevention of serious medication errors. JAMA. 1998; 280:1311–6.

[8] Sittig DF, Stead WW. Computer-based physician order entry: The state of the art. J Am Med Inform Assoc . 1994; 1:108–23.

[9] Bates DW, Leape LL, Cullen DJ, Laird N, Petersen LA, Teich JM, et al. Effect of computerized physician order entry and a team intervention on prevention of serious medication errors. Jama 1998; 280(15):1311-6.

[10] Oliven A, Michalake I, Zalman D, Dorman E, Yeshurun D, Odeh M. Prevention of prescription errors by computerized, on-line surveillance of drug order entry. Int J Med Inform 2005; 74(5):377-86.

[11] Heather E. Finlay-Morreale, Clifton Louie, and Pearl Toy. Computer-generated Automatic Alerts of Respiratory Distress after Blood Transfusion. J. Am. Med. Inform. Assoc. 2008; 15(3):383-385.

[12] Sztipanovits, J., and Karsai, G., "Knowledge-Based Techniques in Instrumentation," *IEEE/EMBS Magazine,* , Vol. 7, No. 2, 1988, pp. 13-17

[13] Abbott, B., Bapty, T., Biegl, C., Karsai, G., and Sztipanovits, J., "Model-Based Approach for Software Synthesis," *IEEE Software,* May 1993, pp. 42-53

[14] Sztipanovits, J., Karsai, G.: "Model-Integrated Computing", IEEE Computer,V.30. pp. 110-112, April, 1997.

[15] Karsai, G.; Sztipanovits, J.; Ledeczi, A.; Bapty, T.: "Model-integrated development of embedded software", *Proceedings of the IEEE*, Volume: 91, Issue: 1, Jan. 2003 Pages:145 – 164.

[16] Karsai G., Nordstrom G., Ledeczi A., Sztipanovits J.: Towards Two-Level Formal Modeling of Computer-Based Systems, Journal of Universal Computer Science, Vol. 6, No. 11, pp. 1131-1144, November, 2000.

[17] Ledeczi, A.; Bakay, A.; Maroti, M.; Volgyesi, P.; Nordstrom, G.; Sprinkle, J.; Karsai, G.: Composing domain-specific design environments, IEEE Computer, Nov. 2001, Page(s): 44 –51

[18] Mathe, J.; Werner, J.; Lee, Y.; Malin, B.; Ledeczi, A. Model-Based Design of Clinical Information Systems. Methods of Information in Medicine. "Forthcoming".

[19] Martin GS, Mannino DM, Eaton S, Moss M. The epidemiology of sepsis in the United States from 1979 through 2000. New England Journal of Medicine 348: 1546-54, 2003.

[20] Bernard GR, Vincent JL, Laterre P, Larosa SP, Dhainaut JF, Lopez-Rodriguez A, Steingrub JS, Garber GE, Helterbrand JD, Ely EW, Fisher CJ. Efficacy and safety

of recombinant human activated protein C for severe sepsis. New England Journal of Medicine 344: 699-709, 2001.

[21] Angus DC, Linde-Zwirble WT, Lidicker J, Clermont G, Carcillo J, Pinsky MR. Epidemiology of severe sepsis in the United States: Analysis of incidence, outcome, and associated costs of care. Critical Care Medicine 29: 1303-10, 2001.

[22] Dellinger RP, Levy MM, Carlet JM, Bion J, Parker MM, Jaeschke R, Reinhart K, Angus DC, Brun-Buisson C, Beale R, Calandra T, Dhainaut JF, Gerlach H, Harvey M, Marini JJ, Marshall J, Ranieri M, Ramsay G, Sevransky J, Thompson BT, Townsend S, Vender JS, Zimmerman JL, Vincent JL. Surviving Sepsis Campaign: International guidelines for management of severe sepsis and septic shock: 2008. Critical Care Medicine 36: 296-327, 2008.

[23] Object Constraint Language, OMG Available Specification, Version 2.0, http://www.omg.org/technology/documents/formal/ocl.htm

[24] Chen, K., Sztipanovits, J., and Neema, S. 2005. Toward a semantic anchoring infrastructure for domain-specific modeling languages. In Proceedings of the 5th ACM international Conference on Embedded Software (Jersey City, NJ, USA, September 18 - 22, 2005). EMSOFT '05. ACM, New York, NY, 35-43. DOI= http://doi.acm.org/10.1145/1086228.1086236

[25] Rivers E, Nguyen B, Havstad S, Ressler J, Muzzin A, Knoblich B, Peterson E, Tomlanovich M; Early Goal-Directed Therapy Collaborative Group, "Early goal-directed therapy in the treatment of severe sepsis and septic shock", The New England Journal of Medicine, Volume 345:1368-1377, November 8, 2001, Number 19

[26] Brookes, S. D., Hoare, C. A., and Roscoe, A. W. 1984. A Theory of Communicating Sequential Processes. J. ACM 31, 3 (Jun. 1984), 560-599. DOI= http://doi.acm.org/10.1145/828.833

[27] CELA-SPT website: http://www.mc.vanderbilt.edu/medschool/otlm/cela/stp/index.html

[28] Stateflow, The MathWorks, http://www.mathworks.com/products/stateflow/

[29] Miller PB, Martin JB. Model-driven Generation of Individualized Clinical Care Plans to Support Protocol-Driven Clinical Process Management, MOTHIS 2007 Conference Presentation.