A Model-Integrated, Guideline-Driven, Clinical Decision-Support System

Janos L. Mathe, Akos Ledeczi, Andras Nadas, and Janos Sztipanovits, *Institute for Software Integrated Systems, Vanderbilt University*

Jason B. Martin, Liza M. Weavind, and Anne Miller, *Vanderbilt Medical Center*

Peter Miller and David J. Maron, *Vanderbilt HealthTech Laboratory*

Vanderbilt University and its Medical Center are applying model-integrated techniques to specify treatment guidelines as asynchronous processes and implement them in visual dashboards to assist healthcare teams.

Formalizing medical knowledge has been an active research area since the 1960s. Early work focused on creating systems that mapped signs, symptoms, and laboratory results to probabilistic estimates of different diagnoses.[1] These expert systems proved impractical for the everyday practice of medicine. Only with the development of electronic medical records (EMRs) have practitioners adopted knowledge-based systems.[2]

Today, medical knowledge-based systems focus on computerized physician order entry (CPOE) and clinical decision-support advisory systems.[3] CPOE systems depend on comprehensive EMRs to give physicians and nurses the means to create and execute orders for tests, procedures, and medications. CPOE and related systems are often called *physician workflow systems* because they're designed to fit the normative matrix of activities that flow from specific surrounding systems and medical practice standards.

Process management is another knowledge-based medical application area. Vanderbilt Medical Center (VMC) is pioneering the use of *process management dashboards* to inform medical staff of the status of required activities for patients with specific problems. Physicians create activity bundles for treating certain conditions; the dashboard shows the bundled activities and their status with red, yellow, and green indicators to remind hospital staff of which activities are completed and which remain to be done.

The key enabler of our work is model-integrated computing (MIC),[4] an approach and supporting tool suite for model-based software and systems engineering that Vanderbilt University has developed over the past two decades. This infrastructure offers new opportunities in creating clinical decision-support and process-management systems. MIC focuses on formally representing, composing, and manipulating integrated models of information processes and security/safety policies. The formal representation of treatment protocols promotes software reusability and maintainability in the overall management of complex medical processes by explicitly capturing a task's temporal structures and coordination (as opposed to hiding them in the code). MIC also provides tools for automated system generation directly from the models.

The open source MIC tool suite, including the Generic Modeling Environment (GME),[5,6] enables layered, multiple-view system modeling, model transformation, model analysis and validation, model execution, and system design evolution. The MIC tools establish a framework for creating clinical decision-support and process-management sys-

0740-7459/09/$25.00 © 2009 IEEE

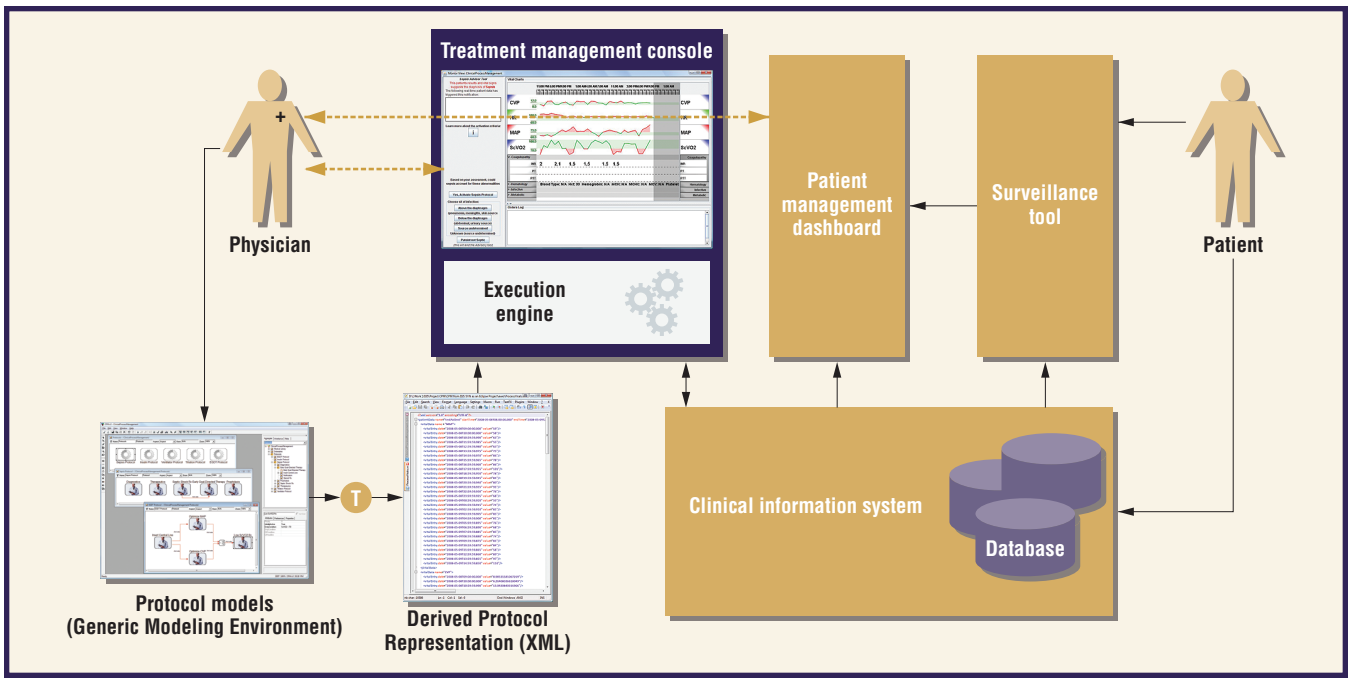


Figure 1. The Sepsis Treatment Enhanced through Electronic Protocolization (Steep) system architecture. Physicians design and update protocol models offline (left). The system then executes the treatment-management process (center) once the surveillance tool issues a sepsis alert (right).

tems. In this article, we describe its first application to the management of a serious illness.

Motivating Problem: Sepsis Management

To maximize our system's impact, we sought a clinical paradigm that was common, clinically important, and expensive with accepted, evidence-based treatment guidelines. Sepsis proved an ideal candidate. The sepsis syndrome results from a robust host reaction to infection and is characterized by a systemic inflammatory response, frequently with very low blood pressure and multiple organ failures. The disease process is very common. About 750,000 cases occur in the US annually,[7] and about 30 percent of septic patients die from the disease.[8] Severely septic patients consume many hospital resources, requiring on average 7–10 days in intensive care units (ICUs) and 3–5 weeks in a hospital. Sepsis-related expenditures are estimated to approach US$17 billion annually in the US alone.[9]

Sepsis treatment is a complex, extremely information-intensive process performed in ICUs and emergency departments. Given the large scope of this clinical problem, it's 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. The SSC has published a comprehensive set of treatment guidelines based on graded clinical evidence. The guidelines are widely considered to represent the state of the art in sepsis management,[10] but they will evolve over time. Also, they must be customized to individual patient needs, and their correct application has important quality and cost implications in sepsis care. The SSC guidelines are complex and require multiple time-sensitive interventions based on dynamic patient variables. Correct and timely implementation of the guidelines requires continuous assimilation and interpretation of numerous pieces of patient data.

We can categorize ICU information technology (IT) interventions generally (in order of increasing sophistication) as clinical reminders, clinical pathways, or real-time protocolized decision-support tools. Real-time tools continuously monitor specific patient variables; if they detect an unmet clinical need, they make treatment recommendations based on clinical guidelines. To our knowledge, the current effort is the most comprehensive attempt at managing sepsis through a sophisticated electronic detection and management tool.

System Architecture and Operation

The Sepsis Treatment Enhanced through Electronic Protocolization (Steep) system is a tool to manage sepsis treatment. Figure 1 shows the Steep system architecture with two distinct operational phases presented side by side:

- on the left, the design and occasional update of evidence-based treatment protocols (protocol models) and

Figure 2. The Steep treatment management console. The TMC includes two panels: on the left, the Steep advisory panel shows recommended actions; on the right, the monitoring panel displays patient health information.

- on the right, continuously running sepsis detection and treatment management via protocol execution.

A dedicated healthcare professional performs protocol design and maintenance offline. The GME tool enables the capture of treatment protocols. For Steep, we configured the GME to implement the Clinical Process Management Language (CPML), a visual domain-specific modeling language (DSML) designed for capturing treatment protocols. From the CPML protocol models, Steep automatically generates Derived Protocol Representation files that configure the system's execution engine.

The sepsis-detection and treatment-management process is integrated with Vanderbilt's existing clinical information system to access real-time patient data streams and to facilitate ordering medications or procedures, for example. The surveillance tool monitors specific lab and vital-sign abnormalities that are quite sensitive for diagnosing sepsis but lack specificity without clinical input and contextual interpretation. When a patient's results indicate the need for further review, the system alerts the healthcare team—first by a visual cue on the ICU patient management dashboard. If the dashboard alert isn't addressed in a timely manner, the system sends an electronic notification via a text page to appropriate team members. If the physician suspects that infection is causing the abnormal physiological parameters, he or she activates decision support.

The execution engine starts running the treatment-management process by executing the protocol models. It also interacts with the treatment management console (TMC), a GUI that physicians use to assess the treated patient's health status, get decision support from evidence-based guidelines on the screen, and actuate their decisions. The TMC facilitates this interaction between the physician and the system through two panels: the monitoring panel and the advisory panel (see Figure 2). The monitoring panel presents a timeline for viewing categorized patient health information in context with the therapeutic actions provided to the patient. Displaying cause and effect relations involves linking patient data and treatments to show the effects of one on the other. We call this the *action-reaction concept*. The protocol models define this information (both displayed indicators and available treatment actions). In effect, they transform the generic GUI to a protocol-specific interface. Vital signs, including temperature, blood pressure, heart rate, and central venous pressure are health indicators that the EMR feeds to the system as a data stream. Laboratory test results, on the other hand, are updated on the screen when the information becomes available. The panel also shows the actions of the treatment that the patient received or is scheduled to receive. All displayed data is temporally aligned on the screen.

The advisory panel helps the physician make a formal diagnosis by using the built-in logic and available action controls. These include higher-level actions, such as selecting the sepsis severity level, as well as lower-level controls, such as ordering specific medications and procedures.

CPML Design

DSMLs require the specification of the language's abstract syntax, concrete syntax, semantic domain, and the mappings between the abstract and concrete syntax (syntactic mapping) and the abstract syntax and the semantic domain (semantic mapping).[5] The formal representations of these specifications are the language's metamodels. In MIC, the metalanguage for representing the abstract syntax of DSMLs and the syntactic mapping is based on UML class diagrams (with stereotypes) and the Object Constraint Language (OCL).[11] 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, semantic mappings are formally represented by using graph rewriting rules.[5,12]

The formal specification of CPML has proved to be difficult, first because healthcare organizations rarely phrase operational protocols, policies, and treatment guidelines in a mathematically sound, unambiguous manner. Second, healthcare prac-

titioners must consider the protocols that describe the medical processes constituting a treatment, their triggering conditions, and their coordination as guidelines—not rigid workflows that must be enacted the same way every time. This requirement is essential for the design of a model's execution semantics.

Because of these challenges, the CPML 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 multiple, predefined, possible outcomes or actions representing treatment steps. This approach followed the formalization efforts presented in the available medical literature,[12] and it was simple. However, it didn't express complex treatments efficiently, and it didn't scale well because the potential trajectories generated by the many concurrent and interacting treatment processes grew exponentially.

We therefore approached the problem from a new direction, grouping treatment steps under process concepts. Processes are concurrent, asynchronous, and interactive with each other via events. To capture the decision logic concisely, we organized processes in a hierarchical manner. Processes can listen to events happening around them and start running only if their triggering conditions are satisfied. Processes are coordinated with the help of events and related messages. The execution semantics of the selected process model corresponds to the Communicating Sequential Processes (CSP) model.[13] The CSP model lets us use hierarchies and define the segments of a complex protocol independently from each other (because processes can be composed in CSP). This semantics also proved to be more intuitive to the physicians, because it more closely resembles the mental process of medical decision-making.

A detailed description of CPML is beyond the scope of this article; however, Table 1 describes the language's major abstractions and their relations. Figure 3 shows segments of the metamodel.

Table 1

High-level concepts of the Clinical Process Management Language (concrete syntax)

Abstraction	Description
Medical Library	Top-level concept that serves as the placeholder for hierarchically categorizing general medical knowledge. Medical Library components serve as a knowledge base for the rest of the language. The three main information categories stored in a Medical Library are patient vitals, laboratory tests, and medications (see Figure 3a). Protocol and Orderables models use these components by reference.
Orderables	Top-level concept for building a hierarchical library for executable actions. Orderables provide the means for building bundles that are available for healthcare professionals. The actions include procedures, medications, and lab tests (see Figure 3b). Activity components in a Protocol refer directly to Orderables.
Protocol	Top-level concept for describing medical protocols (see Figure 3c).
Process	Coordinated group of activities used in Protocol models. Processes help decompose the treatment protocol and organize the treatment steps. Processes are concurrent and asynchronous, and they can interact with each other via Events.
Event	Component used in a Process. Events refer to state changes (such as activation, initiation, and completion events) of other components (such as a Protocol, Process, or Activity). They help create dependencies among models.
Activity	Lowest-level components of a Protocol. They are the representation of what actions must be performed at a given time as part of the treatment. Activities include ordering lab bundles, medication bundles, single medications, and procedures.
Step	Coordination primitive captured as a connection that specifies the execution order of Activities within a Process.
Synchronizing merge	Coordination primitive defining a synchronization point between activities where multiple Steps converge into a single Step. This means that if more than one path is taken, synchronization of the active paths must occur.

Operational Semantics

The operational (behavioral) semantics specify a CPML model's behavior at runtime. CPML processes have five states: Deactivated, Active, Running (Enabled), Paused, and Terminated. An instantiated Process's state is determined on the basis of its InitiallyActive attribute. This attribute's default value is false, which initializes a Process in the Deactivated state. Processes in the Deactivated state do not perform any actions. If the InitiallyActive attribute is set to true or the Process receives an explicit activation message, then the Process moves to the Active state. Active processes monitor runtime events. If an Active process's EntryCondition attribute—a logical expression containing an event such as specific changes in one or more vital signs—becomes satisfied, it starts Running and its subprocesses get activated. Steep can suspend Running protocols and resume them later on demand.

The execution engine implements the protocol models' operational semantics (see Figure 1). It creates a concurrent state machine for every Protocol, Process, and Activity. It also provides the means for process synchronization by using implicit and explicit communication methods: condition evaluation and message exchange, respectively. Conditions typically include references to events (including time) and perform data evaluation.

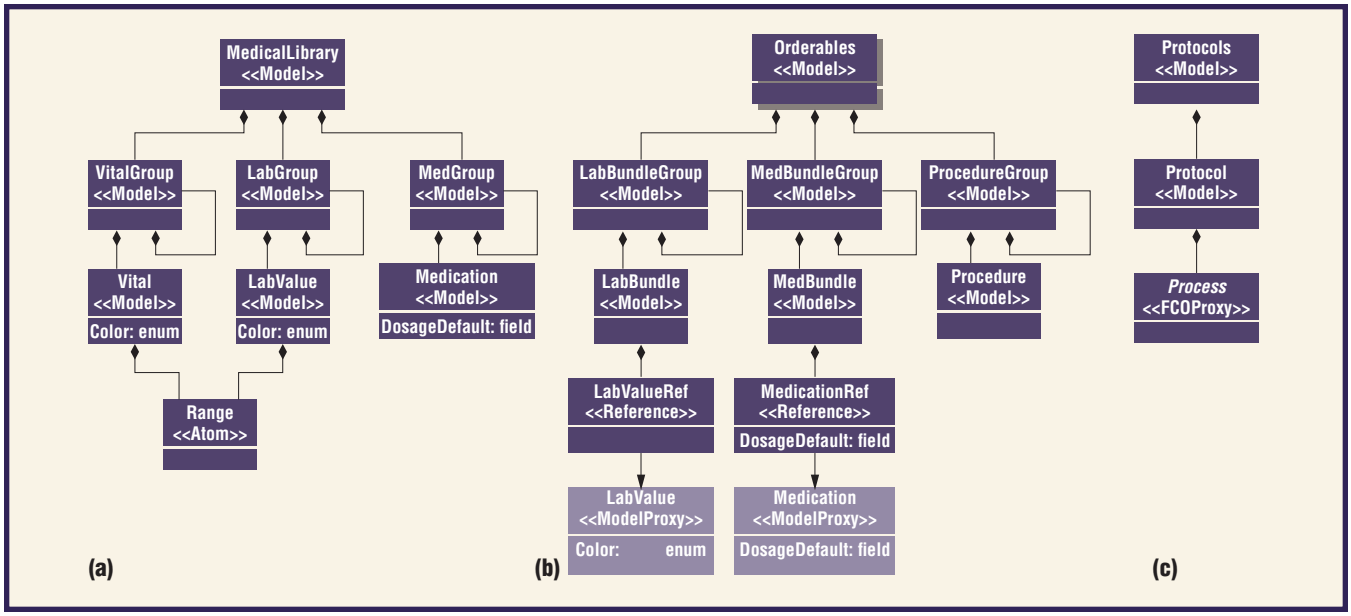


Figure 3. Segments of the Clinical Process Management Language in the Generic Modeling Environment: (a) the Medical Library, (b) Orderables, and (c) Protocols. CPML groups treatment steps into processes that can be organized hierarchically.

The Sepsis Protocol

CPML models capture the medical knowledge related to sepsis. Figure 4 presents an example that describes two components of the sepsis protocol. The main Protocols window contains the Sepsis Protocol model. The model's contents are shown in the Sepsis Protocol window, consisting of five Processes that are activated in the order specified by the activation arrows (from left to right) once the protocol starts executing. This activation mechanism has no direct control over the execution order of the processes; it just constrains the order by specifying when the components start to listen. The execution order isn't determined until runtime, when Steep can evaluate the entry conditions for processes.

The window labeled Diagnostics is the last window opened in the Figure 4 example. It shows the contents of a fairly simple process to initiate the ordering of laboratory test bundles, such as the one including the complete blood count (CBC) lab test. This process has no entry condition and is marked initially active, which means it will start executing immediately after the protocol starts. No dependencies exist among the provided actions (various laboratory tests), so their execution will be initiated simultaneously. During the protocol's execution, this generates a reminder on the TMC advisory panel (see Figure 2, left side) to order the listed laboratory tests.

Discussion

The use of evidence-based guidelines for managing complex clinical problems has become the standard of practice, but guidelines are protocols and not patient care plans. To be truly effective, protocols must be deployed as customized, individualized clinical care plans (protocol instances). Our approach inherently supports this idea by allowing protocol models to be tailored on a per-patient basis, if necessary, and treatment to be customized via the TMC at the bedside.

We had to develop a DSML because no widely accepted visual languages exist for capturing treatment protocols, and generic software modeling languages, such as UML, weren't designed for representing medical knowledge. The use of model-integrated techniques provides several benefits. The protocol models capture medical knowledge explicitly and avoid ambiguity. Medical professionals comprehend the models easily, eliminating the need for IT personnel to mediate between the medical and computer fields.

Furthermore, the protocol models enable knowledge transfer because they're based on the best practice available at the time. Medical students and residents using the tool thereby learn expert knowledge in actual practice. Moreover, the models can be updated on a regular basis as new findings emerge in the medical literature. Finally, the system facilitates the tracking of protocol execution, which helps not only increase compliance but also improve the protocols themselves by enabling the analysis of outcomes.

While our approach's medical benefits are clear, it also presents several advantages from a software development perspective. The software architecture is generic and expected to work just as well for other illnesses as it does for sepsis. In fact, we've already begun modeling congestive heart failure (CHF), a completely different problem. CHF is a chronic

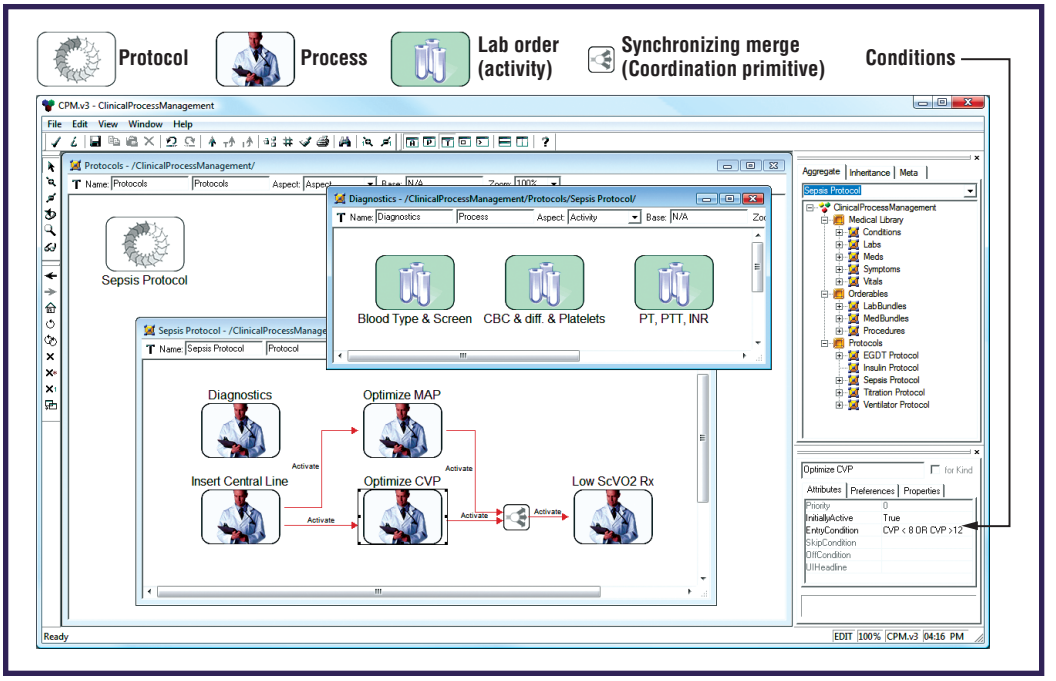


Figure 4. Sepsis management models expressed using CPML (partial view). The Sepsis Protocol model appears in the Protocols window. Opening the model reveals the five Processes in the Sepsis Protocol window. Opening the first Process shows the Diagnostics window for ordering laboratory test bundles.

condition with patients typically living at home, as opposed to acute sepsis, where treatment is administered in the ICU. We don't expect any software changes to the main components of the system as we attack different illnesses, just as there are no software changes when the protocols are updated according to new medical knowledge.

Treatment protocols, even if they serve only as guidelines in 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 MIC approach is that modeling languages are formally sound and provide a foundation for disciplined validation and verification processes.

Validation

Protocol validation tests whether the generated decision-support guidance corresponds to clinicians' expectations. The first step is to model walkthroughs with clinicians. The modeling language's expressiveness is helpful in this process and fully confirms the importance of using DSMLs highly customized to the clinical environment. Physicians actively participated in CPML's iterative development over several months. In our experience with many different domains, domain expert involvement in DSML development is an absolute necessity.

The second validation step is simulation-based studies. The Steep system architecture supports the generation of simulated execution through a supervisor console. The console helps the supervisor control the environment, including the simulated patient's response to treatment and the behaviors of other simulated players, such as physicians ordering drugs and procedures, nurses administering drugs, and laboratories delivering lab results. Sample data for simulated execution of protocols are stored in XML files that the execution engine accesses and the TMC displays just as they would with real data.

The simulation must be conducted in a realistic environment, where ICU personnel can face treatment management situations similar to real life and can interact with the system to make decisions. The validation process must be closely monitored and the results precisely evaluated. VMC provides the infrastructure for this evaluation at the Simulation Center of the Center for Experiential Learning and Assessment (www.mc.vanderbilt.edu/medschool/otlm/cela/stp/index.html). The Simulation Center not only helps validate the protocol models but also provides valuable training to the medical personnel before they use the system in the ICU with actual patients.

Verification

Another benefit of using DSMLs is that the domain models can be formally verified against established criteria. This is a significant step forward. In traditional approaches, where the system is manually coded, the model is not explicit and can't be independently verified. Our models support verification on three levels.

The first line of defense is static model verification, which the GME provides. Metamodels

Protocols are instantiated into a multithreaded program that interacts with personnel, data, and events.

include well-formedness rules that separate syntactically correct models from incorrect ones. The constraints are expressed using OCL. During modeling, GME enforces these well-formedness rules. In CPML, the constraints include clinical limits for parameters as well as more sophisticated rules that would be difficult to check without automated verification.

Next is verification of dynamic properties at design time. The execution engine transforms models into behaviors at runtime. In fact, protocols are instantiated into a complex, multithreaded program that interacts with ICU personnel, patient data, and events. Using 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, and deadlock. We've developed a model translator to map the protocol models into an intermediate executable model using Mathworks Stateflow (www.mathworks.com/products/stateflow). The Stateflow models can drive a number of verification tools, such as model checkers, simulators, and reachability analysis tools. We plan to use these tools in implementing a dynamic verification strategy.

Finally, 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 TMC and to the invocation of actions such as initiating treatment processes and ordering medications. In the current implementation, we rely on general ICU access-control policies, but we intend to make this customizable in later phases. Decisions present in the protocol let healthcare professionals order various actions during treatment that must be not only logged but also matched against a set of legal regulations and the hospital's own policies. Systems interfaced to the execution engine perform several of these checks—for example, the order management system checks all medication-related actions against a large suite of rules.

The Sepsis project started in 2007 as a collaborative effort between the Vanderbilt School of Engineering and Vanderbilt Medical Center to apply advanced MIC techniques to the management of complex clinical processes. The team has completed the beta version of the generic software infrastructure and the sepsis treatment protocol models resulting in the Steep toolset. We are performing a carefully coordinated, multiphase experiment to evaluate the approach in terms of usability and effectiveness. Phase one of the clinical tests has already started in two ICUs at Vanderbilt to establish the baseline for the comparative study. We're gathering data on patient outcomes using the surveillance tool only. The entire Steep toolset will be introduced later this year. We anticipate the application will decrease the time it takes to detect patients with developing sepsis as well as improvements both in physician compliance with evidence-based standards and clinical outcomes for patients.

Once the approach is validated for sepsis, we will apply the technology and corresponding tools to the treatment of other serious illnesses.

Acknowledgments

This work was supported in part by TRUST (Team for Research in Ubiquitous Secure Technology, US National Science Foundation (NSF) award CCF-0424422), the Vanderbilt HealthTech Laboratory, and the Vanderbilt Informatics Center.

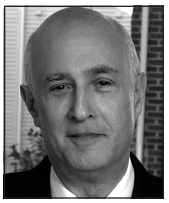
References

1. R.A. Miller, "Medical Diagnostic Decision Support Systems—Past, Present, and Future: A Threaded Bibliography and Brief Commentary," *J. Am. Medical Informatics Assoc.*, vol. 1, 1994, pp. 8–27.
2. W.W. Stead and W.E. Hammond, "Computer-Based Medical Records: The Centerpiece of TMR," *M.D. Computing: Computers in Medical Practice*, vol. 8, 1988, pp. 48–62.
3. D.L. Hunt et al., "Effects of Computer-Based Clinical Decision Support Systems on Physician Performance and Patient Outcomes: A Systematic Review," *J. Am. Medical Assoc.*, vol. 280, 1998, pp. 1339–1346.
4. J. Sztipanovits and G. Karsai, "Model-Integrated Computing," *Computer*, Apr. 1997, pp. 110–112.
5. G. Karsai et al., "Model-Integrated Development of Embedded Software," *Proc. IEEE*, vol. 91, no. 1, 2003, pp. 145–164.
6. A. Ledeczi et al., "Composing Domain-Specific Design Environments," *Computer*, Nov. 2001, pp. 44–51.
7. G.S. Martin et al., "The Epidemiology of Sepsis in the United States from 1979 through 2000," *New England J. Medicine*, vol. 348, 2003, pp. 1546–1554.
8. G.R. Bernard et al., "Efficacy and Safety of Recombinant Human Activated Protein C for Severe Sepsis," *New England J. Medicine*, vol. 344, 2001, pp. 699–709.
9. D.C. Angus et al., "Epidemiology of Severe Sepsis in the United States: Analysis of Incidence, Outcome, and Associated Costs of Care," *Critical Care Medicine*, vol. 29, 2001, pp. 1303–1310.
10. R.P. Dellinger et al., "Surviving Sepsis Campaign: International Guidelines for Management of Severe Sepsis and Septic Shock: 2008," *Critical Care Medicine*, vol. 36, 2008, pp. 296–327.
11. *Object Constraint Language, OMG Available Specification*, v. 2.0, Object Management Group, May 2006; www.omg.org/docs/formal/06-05-01.pdf.
12. K. Chen, J. Sztipanovits, and S. Neema, "Toward a Semantic Anchoring Infrastructure for Domain-Specific Modeling Languages," *Proc. 5th ACM Int'l Conf. Embedded Software* (Emsoft 05), ACM Press, 2005, pp. 35–43.
13. S.D. Brookes, C.A. Hoare, and A.W. Roscoe, "A Theory of Communicating Sequential Processes," *J. ACM*, vol. 31, no. 3, 1984, pp. 560–599.

About the Authors

Janos L. Mathe is a PhD student in Vanderbilt University's Department of Electrical Engineering and Computer Science. His research interests include applying model-integrated computing techniques to security and privacy requirements in healthcare settings. Mathe has an MSc in computer science from the Technical University of Budapest. Contact him at janos.l.mathe@vanderbilt.edu.

Jason B. Martin is a senior clinical fellow in the Vanderbilt Medical Center's Division of Allergy, Pulmonary, and Critical Care. His research interests focus on developing the knowledge and skill necessary to be an effective pulmonologist and critical care physician. He also has an interest in clinical research and technology interventions for optimizing the treatment of patients with systemic infections, or sepsis. Martin has an MD with research honors degree from the University of South Alabama College of Medicine. He completed an internship and residency in internal medicine at the Vanderbilt University Medical Center. Contact him at jason.martin@vanderbilt.edu.



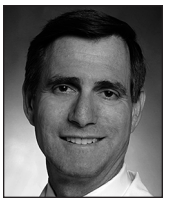
Peter Miller is the director of the Vanderbilt HealthTech Laboratory, which has the mission of transforming healthcare processes through the discovery and demonstration of disruptive informatics-enabled technologies. His research interests focus on biomedical informatics. Miller has an MS/EE from the Massachusetts Institute of Technology. Contact him at peter.miller@vanderbilt.edu.

Akos Ledeczi is a research associate professor in Vanderbilt University's Department of Electrical Engineering and Computer Science and a senior research scientist at its Institute for Software Integrated Systems. His research interests include model-integrated system development and wireless sensor networks. Lédeczi has a PhD in electrical engineering from Vanderbilt University. Contact him at akos.ledeczi@vanderbilt.edu.

Liza M. Weavind is an associate professor of anesthesiology and critical care, director of the Critical Care Fellowship Program, and medical director of the Surgical Intensive Care Unit at Vanderbilt University. Her academic interests focus on educational models for critical-care practitioners and trainees, patient safety initiatives, and the use of telemedicine and technology to improve and standardize ICU care. Weavind received her MB, BCh degree from the University of Witwatersrand Medical School, then completed postgraduate training in internal medicine, obstetrics, and general and cardiothoracic surgery at Johannesburg General Hospital. She completed the anesthesiology residency and critical-care fellowship programs at the University of Texas Medical School. Contact her at liza.weavind@vanderbilt.edu.

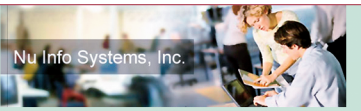
Andras Nadas is a research engineer at Vanderbilt University's Institute for Software Integrated Systems. His research interests are in embedded systems, including wireless sensor networks, configurable model-driven enterprise services, and configurable runtime engines. Nadas has a master's in computer science from the Budapest University of Technology and Economics. Contact him at andras.nadas@vanderbilt.edu.

Anne Miller is an assistant professor in Vanderbilt University's School of Nursing. Her research interests include the role of clinical information displays in facilitating safe, efficient, and effective decision-making and care coordination in interdisciplinary healthcare settings. Miller received her PhD in psychology from the University of Queensland. She completed a postdoctoral fellowship focusing on care coordination in ICUs at the Vanderbilt University Medical Center. Contact her at anne.miller@vanderbilt.edu.



David J. Maron is an associate professor of medicine in Vanderbilt University's Division of Cardiovascular Medicine. He also serves as medical director of the Vanderbilt Center for Health Promotion, director of emergency cardiology, and clinical advisor to the Vanderbilt HealthTech Laboratory. His research interest is the application of optimal medical therapy in the management of coronary heart disease. Maron received his MD from the University of Southern California's Keck School of Medicine. He completed internal medicine training at the University of California, Los Angeles. He was a Robert Wood Johnson Clinical Scholar at Stanford University, where he also completed his cardiology training. Contact him at david.maron@vanderbilt.edu.

Janos Sztipanovits is the E. Bronson Ingram Distinguished Professor of Engineering at Vanderbilt University and founding director of the Institute for Software Integrated Systems. His research interests are at the intersection of systems and computer science and engineering, including model-integrated computing for distributed embedded software, structurally adaptive systems, autonomous systems, design space exploration, and systems-security codesign technology. Sztipanovits has a doctorate in electrical engineering from the Hungarian National Academy of Sciences. He's a Fellow of the IEEE. Contact him at janos.sztipanovits@vanderbilt.edu.



Nu Info Systems, Inc.

NU INFO SYSTEMS, INC., a software consulting firm, headquatered in West Palm Beach Florida, has multiple ongoing opportunities in each technology set for experienced software professionals.

CANDIDATES MUST HAVE:

- Bachelor in Comp Science/Engg., Math or Business & 2 years experience in stated technology group or five years experience in job in required skills.
- Some positions require Master's Degree or equivalent and two years of experience.

TECHNOLOGIES WANTED:

- JAVA, J2EE, JAVA Server, JSP, XML
- VC++, C++, C, MFC, COM/DCOM, OOD
- Systems Administrators: Unix/Solaris/AIX/Windows NT
- Win Runner, Load Runner, TSL, Rational Suite
- VB.NET, ASP.NET, VB 6.0, MS SQL Server, Oracle
- GIC, ARC INFO, ARC GIS, ARC MAP, ARC IMS
- Siebel, Siebel Actuate, Siebel Analytics, EIM & EAI
- C++, Unix, PERL, CORBA, Oracle, Multithreading, SS7 Protocol, TCP & IP, CAIN
- COBOL, CICS, DB2, JCL, MVS
- Database Administrators: Oracle, SQL Server, DB2,
- Power Builder, PFC, Oracle, Sybase SQL Server
- Management Analyst - UML, JAD Sessions, Rational Rose.

QUALIFIED CANDIDATES:

Please **EMAIL** CV to: hr@nuis.com *or*
FAX to: **561-828-6383** *or*
MAIL to: Attn: HR Dpartment
8461, Lake Worth Road, Suite 225
Wellington, FL 33467

NU INFO SYSTEMS, INC. www.Nuis.com

www.nuis.com