A Benchmark for Evaluating Software Engineering Techniques for Improving Medical Processes

Stefan C. Christov
Dept. of Computer Science
University of Massachusetts
Amherst, MA 01003
christov@cs.umass.edu

George S. Avrunin
Dept. of Computer Science
University of Massachusetts
Amherst, MA 01003
avrunin@cs.umass.edu

Lori A. Clarke
Dept. of Computer Science
University of Massachusetts
Amherst, MA 01003
clarke@cs.umass.edu

Leon J. Osterweil
Dept. of Computer Science
University of Massachusetts
Amherst, MA 01003
ljo@cs.umass.edu

Elizabeth A. Henneman
School of Nursing
University of Massachusetts
Amherst, MA 01003
henneman@nursing.umass.edu

ABSTRACT

The software engineering and medical informatics communities have been developing a range of approaches for reasoning about medical processes. To facilitate the comparison of such approaches, it would be desirable to have a set of medical examples, or *benchmarks*, that are easily available, described in considerable detail, and characterized in terms of the real-world complexities they capture. This paper presents one such benchmark and discusses a list of desiderata that medical benchmarks can be evaluated against.

1. INTRODUCTION

Problems in health care have gained prominence in recent years. A 2009 US National Research Council report [14] argues that "these persistent problems do not reflect incompetence on the part of health care professionals—rather, they are a consequence of the inherent intellectual complexity of health care taken as a whole and a medical care environment that has not been adequately structured to help clinicians avoid mistakes or to systematically improve their decision making and practice." This report also notes that current health care information technology is rarely used to support data-driven process improvement. To address such concerns, the software engineering and medical informatics communities have been developing a range of approaches for evaluating medical processes (e.g., [3, 6, 15]). The complex nature of medical processes—for example, the possible involvement of a diverse set of medical professionals, the abundance of exceptional situations, and the concurrent execution of activities —makes these processes hard to model and analyze and the corresponding methodologies hard to evaluate.

Because of the complexity of medical processes, it is important to automate, at least partially, the reasoning about them. Researchers have thus used modeling notations with formal semantics to make the process models amenable to automated analysis. In this paper, we refer to a process model created in such notations with rigorous, formal semantics as a *process definition*.

To empirically evaluate process modeling notations and analysis techniques, researchers have applied them to different examples from the medical domain. For instance, the Little-JIL process definition language and the FLAVERS finite-state verification analysis technique have been applied to discover defects and suggest improvements in both blood transfusion and chemotherapy processes [3, 4]; the Asbru language and the KIV theorem prover have been used to reason about a jaundice protocol [15]; Message Sequence Charts translated to guarded Labeled Transition Systems have been combined with various kinds of static analyses to reason about a cancer therapy process [6].

Evaluating such process modeling notations and analysis techniques on medical examples is an important step towards a rigorous discipline of process improvememt. An important question arises: "What are the relevant strengths and weaknesses of each modeling notation or analysis technique?" and consequently "For what kinds of medical processes should one use a certain process modeling notation or a certain analysis technique instead of another notation or analysis technique?" Questions like these are difficult to answer when each modeling notation or analysis technique has been evaluated in the context of a different medical example and when many of these medical examples are not easily accessible, are ambiguously described, or are not carefully characterized in terms of the real-world complexities they are meant to capture. What seems to be needed is a set of medical examples, or *benchmarks*. Ideally, these medical benchmarks would be both easily accessible by the research community and rigorously described to reduce ambiguity and help ensure that different researchers are working on the same example. Preferably, these benchmarks should also be written in notations that are relatively easy to understand and would be characterized in terms of the complexities of the real-world medical processes that they capture. Such characterizations would guide potential users in their choice of benchmarks. A set of medical benchmarks would not only

SEHC '10 May 3-4, 2010, Cape Town, South Africa

facilitate the comparison of different process modeling notations and analysis techniques in terms of their applicability to health care, but it could also provide an infrastructure for controlled experimentation and reproducibility. In addition, a set of medical benchmarks could help lessen the burden on software engineering and informatics researchers of eliciting these processes from medical domain experts. Our experience indicates that eliciting an adequate description of a medical process can be very time-consuming for both the domain experts and the computer scientists.

This paper presents one such medical benchmark, based on a blood transfusion process. The components of this benchmark include: a blood transfusion process definition created in the Little-JIL language; a set of properties, or requirements, that a blood transfusion process must satisfy; and a set of bindings that establish the relationship between the blood transfusion properties and process definition.

The contributions of this paper are the blood transfusion medical benchmark and a list of desiderata against which medical benchmarks can be evaluated.

2. RELATED WORK

The medical community has created many descriptions of standard medical procedures (e.g., [17,18]) for purposes such as promoting evidence-based practice or training health care professionals. These medical process descriptions are usually written using natural language as a narrative, as a checklist, or as a high-level flow chart. They aim to provide important procedural details but have several drawbacks that make them difficult to use directly as benchmarks. These descriptions often contain poorly defined and inconsistently used terms and tend to focus on the normative workflow, omitting important details about exceptional scenarios.

The medical informatics community has created several languages that can be used to represent medical processes (e.g., EON [10], GLIF [11], Asbru [13]). These languages tend to have well-defined semantics, and thus medical process descriptions specified in these languages tend to be more precise and less ambiguous than the natural language descriptions mentioned above. Some of these languages also have support for integrating domain ontologies into the process description, which can further reduce ambiguity. In this respect, medical process descriptions created in such languages are a step closer to being usable as benchmarks.

Different languages have different strengths and weaknesses in terms of the semantic constructs they support, such as concurrency, exceptional flow, and real time constraints. The degree to which process descriptions capture such aspects is thus affected by the semantic richness of the language used. To compare several languages for specifying medical guidelines, Peleg et al. [12] specified portions of two medical guidelines in these languages. These guidelines and their corresponding descriptions focused on providing decision support to physicians about what tests to order and treatments to prescribe. The benchmark presented in this paper focuses mostly on the activities or how health-care providers should perform to deliver the selected treatments. We believe that both of these perspectives on medical processes are important and that benchmarks from each perspective are needed.

Researchers have used process descriptions created in some of the languages mentioned above to evaluate the applicability of several software engineering techniques to health care (e.g., [6, 15]). To the best of our knowledge, however, the medical process descriptions that were used for these evaluations are not publicly available to be used as benchmarks.

3. PROCESS DESCRIPTION DESIDERATA

This section discusses characteristics that should be taken into account when considering candidate medical processes as benchmarks. As the community develops a set of medical benchmarks and gains more experience with their use, we would expect this list to be augmented.

Detail and Precision. It is important that medical process descriptions be *detailed* enough to support meaningful reasoning about the process. For example, if one wishes to reason about the risks that can arise if a paper copy of a treatment plan becomes inconsistent with an electronic version, the process description needs to provide enough detail about the use of the paper and electronic versions to detect such inconsistencies. If a medical process description is to be utilized as a benchmark to compare the capabilities of different, and probably automated, analysis approaches, then it needs to be *precise* enough to avoid ambiguity and to ensure that the results from applying these analyses are based on process descriptions that are sufficiently complete and consistent. To further help reduce ambiguity, it is desirable to use a notation with well-defined semantics.

Artifacts. An *artifact* is an instance of a type of object (e.g., a blood tube or a treatment plan) that is consumed, produced, or used in a health care process. As mentioned in the previous paragraph, reasoning about certain features of a medical process may necessitate information about artifacts such as the versions of a treatment plan. Thus, it is often desirable that a medical process description incorporate an explicit mechanism for specifying the artifact types and instances that are involved in the process.

Agents. The involvement of a diverse group of medical professionals is typical for many situations in health care. For example, in a chemotherapy preparation and administration process, physicians, nurses, pharmacists, and support staff may all participate. For the purposes of this discussion, we define an *agent* to be an instance of a type of human, machine, or software system that is capable of performing some set of activities (e.g., triage nurse or robot that performs a surgery). To capture the complexity of real-world medical processes, process descriptions often need to provide information about the different agent types, and sometimes about particular agent instances (e.g., physician Phil or robot R2D2), involved in a process.

Medical professionals are continually making decisions influenced by both the patient's medical condition and by their own personal working style. As a result, such decisions cannot be specified a priori and it is often desirable that medical process descriptions provide support for letting agents make decisions. For human agents, this might involve a sense of "free choice", while for automated agents this might be represented by nondeterminism.

Resources. *Resources* are typically artifacts or agents for which there is contention. Health care processes often involve a variety of such resources (e.g., surgeons, beds, and X-ray machines). Resource availability can play a critical role in the quality, efficiency, and cost of health care. Thus, if a medical process description is to be used to reason about, or to evaluate tools that can be used to reason about, such issues, then that process description needs to contain ade-

quate representation of the resources used in the process. Since resource utilization often involves contention, process descriptions require additional information, such as access control policies, priorities, and resource capacities.

Aspects of Process Flow. To provide an adequate representation of real-world medical processes, process descriptions often need to specify complex process flow, including exception handling, concurrency, and real-time constraints.

Deviations from normal workflow occur in most medical processes. For example, a nurse performing blood transfusion can face a variety of non-normative events: the patient may be missing an ID band, be unconscious, or have a transfusion reaction. In all these situations, the agent (in this case a particular nurse) faces exceptional circumstances and is forced to deviate from the normal execution of the process to handle those circumstances appropriately. Our experience with modeling medical processes indicates that exceptional situations or deviations from the "happy path" are common in the medical domain, suggesting that special attention needs to be paid to them.

In many medical processes, different activities may happen simultaneously. For example, the activities of processing and analyzing a patient's test results (e.g., evaluation of laboratory and X-ray results) could happen in parallel. In addition, at different points in the process, information may need to be exchanged or coordinated before other activities can proceed. For example, surgery may not proceed until the lab and X-ray results have both been received.

Time-critical activities are prevalent in certain types of medical processes. During a surgery, for example, it may be necessary for tasks to be performed within a certain time frame.

4. THE BLOOD TRANSFUSION BENCHMARK

This section starts with a high-level description of the real-world in-patient blood transfusion process on which the benchmark is based. Then, the individual components of the benchmark as well as the tools used to produce and analyze them are described. The full benchmark is available at `http://laser.cs.umass.edu/btbenchmark/`.

The benchmark components were created in the course of several years of collaboration between medical professionals and computer scientists. The main goals of the project were to define and analyze high-risk medical processes to improve their safety and efficiency as well as to develop and improve software engineering techniques for defining and reasoning about processes. The benchmark components are based on a standard blood transfusion process. Standard process descriptions from the medical literature, however, tend to lack precise specification of exceptional scenarios. Thus, the knowledge of domain experts was used to describe common exceptional scenarios in the process definition.

4.1 Overview of the Process

In the blood transfusion process, a nurse receives a physician's order to transfuse one or more units of blood into a patient. To carry out the order, the nurse performs several subprocesses: 1) checking that the patient's blood type and screen are available and if they are not, obtaining a blood specimen so that the type and screen can be performed; 2) preparing documentation, and picking up the unit of blood from the blood bank; 3) performing the transfusion; and 4) performing follow-up documentation. The blood transfusion benchmark focuses mainly on the first and third of these subprocesses, since they are the most safety-critical.

If the type and screen have not been performed, the nurse needs to obtain a blood specimen from the patient so that the type and screen can be performed. Before the nurse can obtain the specimen, a physician's order to do so is needed. There is a possibility, however, that the computer system is temporarily down, in which case the physician needs to use a special downtime requisition form. Once the nurse has the physician order, the nurse needs to obtain the appropriate specimen labels and equipment for specimen collection, verify the patient's identity, confirm that the information on the patient's identification band matches the information of the specimen label, label the specimen, perform the blood draw, and send the blood specimen to the lab. This subprocess must be conducted in the order described with one exception—the blood label may be applied either prior to or immediately after obtaining the specimen for type and screen. An important safety property is that no other activities may occur between obtaining and labeling the specimen. Complications can arise at several points in this subprocess (e.g., the information on the patient's ID band does not match the information on the specimen label). All of these details are important for a high-risk process such as blood transfusion and, together with all of the above exceptional scenarios that may arise, make the blood transfusion process particularly challenging yet interesting to study.

The subprocess for performing the actual transfusion is also complex. If a patient is losing a lot of blood, the physician may order the administration of several units if blood. In that case, the nurse may need to manage several transfusions simultaneously. Before administering a unit of blood, the nurse needs to clinically assess the patient, obtain the equipment necessary for transfusion, and perform bedside checks. Each of these tasks has its own intricacies —the clinical assessment may reveal problems in patient history that need to be addressed before the transfusion can be started, necessary equipment may be unavailable, or there could be problems with verifying the patient's identity. In addition, the verification of the blood product information (part of the bedside checks) needs to be performed in cooperation by two different nurses as a safety measure.

Most of the activities in the blood transfusion process are performed by a nurse. Tasks performed by other agents, however, are also an important part of the process. Some examples are blood bank staff tasks for preparing the correct unit of blood and physician tasks for ordering the blood transfusion and deciding if the patient develops a transfusion reaction. The blood transfusion process requires a rich assortment of resources and artifacts (e.g., physician order, patient ID band, unit of blood product, etc.). The blood transfusion process is also interesting in terms of process flow. The normative flow is complex and has many important details itself, but elaborate deviations from the normative flow can also arise as a result of exceptional events. The process exhibits some concurrency as a nurse may need to perform multiple blood transfusions for the same patient simultaneously. Real time constraints are also inherent to the process (e.g., the nurse needs to act within a certain time if a patient develops a transfusion reaction).

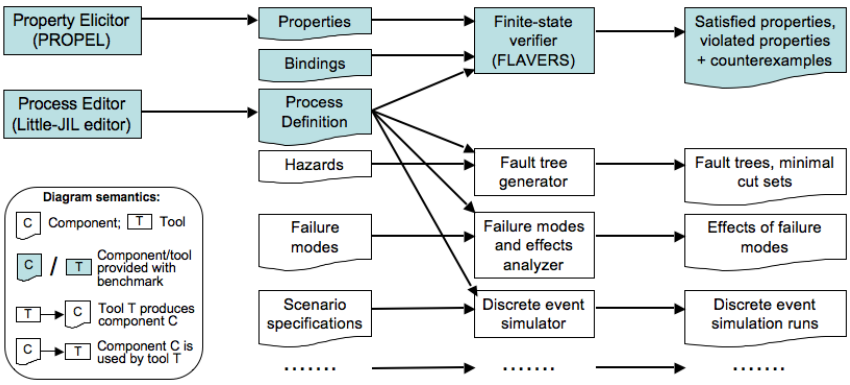

Figure 1: Benchmark architecture

4.2 Benchmark components

Figure 1 shows an architectural view of how the benchmark components relate to each other and indicates some of the tools and artifacts that could be used to populate these components. The second column in Figure 1 (i.e., "Properties", "Bindings", "Process Definition", etc.) corresponds to benchmark components; the first column shows the tools that we actually used to create some of these components; the third column shows other tools that could utilize the benchmark components for various kinds of analyses; and the fourth column shows potential results that might be generated from these tools. The shaded benchmark components and tools are those for which specific instances are being made available to the community for use. We have utilized the Little-JIL editor to create a definition of the blood transfusion process in Little-JIL [2] and the PROPEL property elicitation tool [5] to create a set of blood transfusion properties. To establish the correspondence between the properties and the process definition, we used the Little-JIL environment (not shown) to create a set of bindings between the events in the properties and the step names in the Little-JIL definition, since these components are usually created independently of each other. The Little-JIL definition, the properties, and the bindings are input to the FLAVERS finite-state verifier [8], which then checks if all possible traces through the process conform to the stated properties. The verification results consist of a confirmation that the blood transfusion process definition satisfies a given property or of a counterexample showing a process definition trace that violates the property. The shaded components and tools are described in more detail in the ensuing subsections.

The unshaded shapes in Figure 1 are components that can potentially be added to the benchmark and tools that can potentially operate on the existing or the newly added benchmark components. These components serve as examples of how the benchmark can continue to grow and the tools are ones that we currently have under development.

4.2.1 Process definition

The definition of the blood transfusion process included in the benchmark was created in the Little-JIL process definition language [2] using the Little-JIL editor. Little-JIL's support for concurrency and synchronization, exception handling, resources and artifacts makes it suitable for modeling the complex aspects of medical processes, as discussed above, and its formally defined semantics make processes defined in it amenable to rigorous analysis. A Little-JIL process definition consists of three main specifications—a resource specification, an artifact specification, and a coordination specification. The resource specification defines the agents and resources (human and non-human) needed to perform process activities. The artifact specification defines the products of the process activities. The coordination specification brings these two together by defining which agents, using which resources, perform which activities on which artifacts at which times. The main building blocks of the Little-JIL coordination specification are steps. A step corresponds to an activity performed by a human or non-human agent and, in the graphical representation, is shown iconically by a black bar. A Little-JIL process definition is a hierarchical decomposition of steps where each step can be decomposed into substeps to an arbitrary level of detail.

Figure 2 illustrates some of the detail captured by the Little-JIL definition of the blood transfusion process. It shows the decomposition of the step *perform pre-infusion work*, part of the larger blood transfusion process definition. *Perform pre-infusion work* is a sequential step (indicated by the right arrow in the step bar), meaning that the agent(s)[1] need to perform the substeps in order from left to right.

Three of the substeps of *perform pre-infusion work* in Figure 2 can throw exceptions (indicated by the notes[2] under the step bars). When a Little-JIL step throws an exception, the exception propagates up the step hierarchy until a matching exception handler is found, and then executed. Thus, when the step *assess patient* throws *ProblemFoundInPatientHistory* exception, the matching exception handler *handle ProblemFoundInPatientHistory* is executed (exception handlers are connected to the "X" in the step bar of *perform pre-infusion work*). Similarly, when the other two substeps of *perform pre-infusion work* throw exceptions, their corresponding handlers are executed.

Exception handlers are themselves steps and can thus be hierarchically decomposed to an arbitrary level of detail and can throw exceptions. The handler *handle ProblemFoundInPatientHistory*, for example, can throw *ReceivedOrder-*

[1]Agent and artifact information is elided from these diagrams. The types of agents involved in the blood transfusion process are a nurse, a physician and blood bank staff.

[2]The notes are not part of the Little-JIL visual syntax but they are included in the diagrams here for clarity.

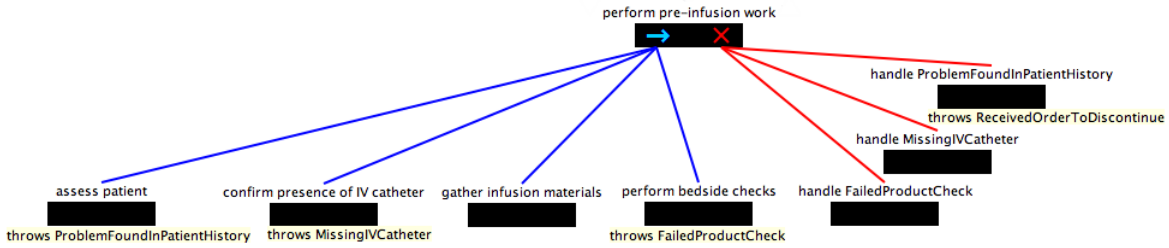

Figure 2: Elaboration of *perform pre-infusion work*

ToDiscontinueTransfusion exception, which corresponds to the situation when the physician has decided that blood transfusion cannot be completed. When the handling of an exception is completed, execution continues based on the resumption semantics indicated in the handler. In the blood transfusion process definition shown in Figure 2, after handling the exception *ProblemFoundInPatientHistory* is completed, the parent step needs to be restarted by the nurse if the physician decides that the problem is not significant enough to stop the transfusion (this communication between the physician and nurse is part of the exception handler).

All of the above steps are further decomposed in the full process definition included in the benchmark, except for *confirm presence of IV catheter* and *gather infusion materials*).

4.2.2 Properties

A *property* is a requirement or a goal that a system or process must satisfy. A property is usually independent of any particular implementation of a specific process or system and thus is required to hold across different processes, and indeed often across different organizations at which the processes are performed. The properties included in the benchmark are constraints on the event sequences that can occur during the execution of a blood transfusion process. For example, a property may state that one event cannot occur until after another one has occurred, or that some particular event must always occur after some other particular event.

Eliciting precise and accurate properties is known to be difficult and error prone [5, 16]. To facilitate property elicitation and specification, we created a set of blood transfusion properties using the Propel tool [5]. Propel provides templates for most of the common finite-state verification patterns identified in [7].

Propel aims to bridge the gap between natural language, which is understandable but is also imprecise, and a mathematical formalism, which is precise but can be hard to understand. Each property specified with Propel has two final representations—a finite state automaton and a disciplined natural language paragraph expressing the different aspects of the property. We believe that the natural language representation of the properties in the benchmark, although verbose, makes them easier to understand, while the formal representation is unambiguous.

To give a sense of the kinds of properties included in the blood transfusion benchmark, we present one of the properties here[3]: *"Before infusing each unit of blood product into a patient, it must be checked that the medical record number (MRN) on that patient's ID band matches the medical record number on the tag affixed to the unit of blood product"*. This property has two main events, namely *infuse unit of blood product* and *check MRN on ID band and product tag match*.

Each property has three elements: an alphabet, a scope, and a behavior. The alphabet is the set of events of interest for a given property. The *scope* specifies over what parts of an execution of the process/system the property is required to hold. The property presented in Figure 3, for example, is required to hold throughout the entire execution of the blood transfusion process but, in general, a scope can specify that a property is required to hold before or after the occurrence of a given event or between the occurrences of two events. The scope can also indicate whether there are certain exceptional events after whose occurrence the property may no longer be required to hold. The *behavior* expresses the constraints between the property events that are required to hold within the specified scope. In addition to the events used to delimit the scope, the primary events participating in the property pattern, and any exceptional events, properties can also have *secondary events*. Secondary events are events that might be restricted or allowed to happen between primary events.

The DNL property statement in Figure 3 elaborates the high-level property statement given informally above. The finite-state machine in Figure 3 is an equivalent representation of the DNL representation of the property, but it is formal and is thus amenable to automated analysis.

4.2.3 Bindings

To check whether a process definition satisfies a given property, one needs to provide a mapping from the abstract events used to define the property to the actual events that take place in the course of execution of the process definition. We call such mapping a *binding*. Thus, to be able to verify the property mentioned in the previous paragraph, for example, the event *check medical record number on ID band and product tag match* is bound to the execution of the step *confirm product tag matches patient ID band*, which is part of the elaboration of *perform bedside checks* shown in Figure 2. The property event *infuse a unit of blood product* is bound to the execution of the step *infuse unit of blood product*, which is in a different part of the blood transfusion process definition (not shown) that is executed after the nurse has completed the step *perform pre-infusion work*. Property events are frequently bound to the executions of steps in a process definition, but they can also be bound to the throwing of exceptions or to the creation or utilization of artifacts during the execution of an activity.

Creating the bindings between the property events and

[3]The presented property is a simplified version of the property in the benchmark.

High-level property statement

Before infusing each unit of blood product into a patient, it must be checked that the medical record number on that patient's ID band matches the medical record number on the tag affixed to the unit of blood product.

Disciplined Natural Language (DNL) property statement

Scope:

1. From the start of any event sequence through the end of that sequence, the behavior must hold.

Behavior:

1. The events of primary interest in this behavior are ***infuse unit of blood product*** and ***check MRN on ID band and product tag match***.
2. There are no secondary events of interest.
3. ***infuse unit of blood product*** is not allowed to occur until after ***check MRN on ID band and product tag match*** occurs.
4. ***check MRN on ID band and product tag match*** is not required to occur.
5. Even if ***check MRN on ID band and product tag match*** does occur, ***infuse unit of blood product*** is not required to occur after ***check MRN on ID band and product tag match*** occurs.
6. After ***check MRN on ID band and product tag match*** occurs, but before the first subsequent ***infuse unit of blood*** occurs, ***check MRN on ID band and product tag match*** is allowed to occur again, zero or more times.
7. After ***check MRN on ID band and product tag match*** and the first subsequent ***infuse unit of blood product*** occur:
   - ***infuse unit of blood product*** is not allowed to occur again until after another ***check MRN on ID band and product tag match*** occurs;
   - ***check MRN on ID band and product tag match*** is allowed to occur again and, if it does, then the situation is the same as when the first ***check MRN on ID band and product tag match*** occurred, meaning that the restrictions described in parts 5, 6, and 7 would again apply.

Finite-state automaton (FSA) property representation

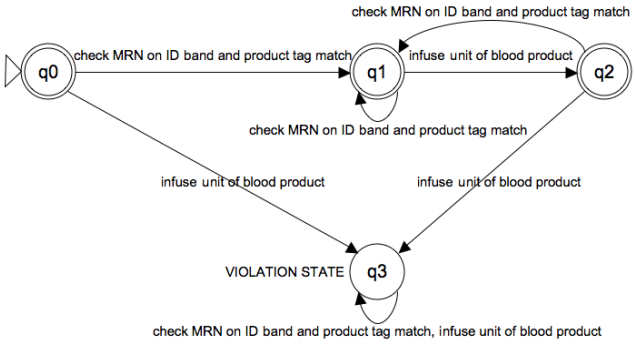

Figure 3: An example property.

the constructs from the process definition is a human intensive process that can be difficult and error-prone. Given a set of events from a property, one needs to choose what kind of process construct to bind the events to (e.g., steps, parameters, exceptions) and also find the appropriate instance of these process constructs (i.e., a particular step, parameter, exception) in a potentially large process definition. The provided tools somewhat facilitate the creation of bindings.

The need to bind abstract events of high-level properties to concrete entities in a process definition is not specific to the notations used here, but arises when verifying properties specified in one notation against process definitions (or other models) written in other formalisms. Analysis results and performance can differ significantly with minor variations in the bindings that do not initially appear significant. Thus, including a set of bindings in the benchmark seems essential if the properties and the process definition included in the benchmark are to be used as the basis for evaluating different analysis techniques.

4.3 Accompanying tools and components

In addition to the main benchmark components—the blood transfusion process definition, properties, and bindings—we also make available a set of tools. The toolset contains: the Little-JIL editor, Visual-JIL, which can be used to view the graphical representation of the process definition and to modify that definition; Propel, which can be used to view and modify the properties as well as to export the FSA representation to standard external formats (e.g., XML); and FLAVERS/Little-JIL, which can be used to create the bindings and to run the finite-state verification analyses.

We also provide the finite-state verification results obtained by applying FLAVERS/Little-JIL. FLAVERS/Little-JIL automatically translates a Little-JIL process definition into a model that represents all sequences of relevant events that could occur on process executions, where the relevant events include the primary, secondary, exceptional, and scope events of the property being verified. Then FLAVERS uses this model to algorithmically check whether the model satisfies a given property. For the blood transfusion example there are currently 23 properties in the benchmark. We were able to specify 21 of them in Propel and verify all of them using FLAVERS. The results are included in the benchmark website. The other 2 properties involved real time constraints and cannot be specified in Propel. We expect to add more properties to the benchmark as we elicit and formalize them.

5. DISCUSSION AND FUTURE WORK

Currently, the definition of the blood transfusion process included in the benchmark covers most of the aspects of medical processes discussed in section 3. The process definition provides a significant amount of detail for important parts of the blood transfusion process (i.e., checking for patient blood type and screen and performing the infusion of a unit of blood product) and its meaning is rigorous since it is defined in Little-JIL, a language with formally defined semantics. Exceptional scenarios, such as what happens when the patient's blood type and screen are unavailable, are also well represented in the process definition. The agents responsible for the execution of the steps in the process are specified, as well as the most important artifacts and resources used in the process. This process definition contains only a few examples of concurrent execution. The process definition does not do a good job of representing the real time constraints, since the Little-JIL language currently has weak support for real time constraints.

Another limitation of the benchmark is that the process definition is specified only in Little-JIL. Any choice of notation biases a benchmark toward information expressible in that notation. The ease of expressing complex behavior and the well-defined semantics of Little-JIL make it a good choice over many process-modeling languages. We also believe that a carefully elicited Little-JIL definition is preferable to a natural-language description, since it tends to be more precise and to contain more details than would normally be specified in a textual notation. Users of the process definition will need to familiarize themselves with Little-JIL's syntax and semantics to fully understand the definition of the blood transfusion process. We are currently working on a tool that automatically generates a natural language description from a Little-JIL process definition. Although use-

ful, this description is verbose and illustrates why naturally-created, textual descriptions tend not to contain all the necessary details. One could argue that a natural language description that is totally independent of any process definition notation would be preferable to one generated from a specific process definition. But we believe that for this description to be accurate and to capture all necessary information, it has to become very artificial, long, and detailed—similar to the automatically generated one—and we are not optimistic about a description that is independent of a formal notation being sufficiently unambiguous and precise.

The set of blood transfusion properties included in the benchmark covers a wide range of requirements, ranging from policy and regulatory requirements to purely clinical requirements. Moreover, they are high-level requirements that should be applicable, with minor variations, to different hospitals. A potential limitation of the set of properties comes from the fact that PROPEL supports only event-based properties. Event-based properties can encode constraints on the order of occurrence of certain process events but are weak in encoding constraints that involve state information. The specification patterns that PROPEL is based on also do not support real time constraints. There has been some work on extending these specification patterns with support for real-time constraints [9] and such extensions to PROPEL will be considered in the future. Most of the properties that we elicited from the domain experts, however, turned out to be event-based properties with no real time constraints and thus we were able to create formal FSA representations and a detailed DNL descriptions for these properties. For the few properties that involved real time constraints, we did not create an FSA or a DNL description, but we provide the high-level natural language statements.

The bindings included in the benchmark are tied to the specific process definition and the semantics of the Little-JIL language and to specific properties. Such bindings are needed if the properties and process definition are developed at all independently from each other. The mapping between property events and constructs from a process definition are not always trivial and the bindings included in the benchmark provide several such examples.

There has been evidence in the research literature that even when easily accessible, unambiguously described, and carefully characterized benchmarks are available, comparisons of different methodologies are still difficult to make [1], and thus we acknowledge that a set of medical benchmarks, like the one described in this paper, will not solve all problems in comparing different methodologies. We believe, however, that creating such benchmarks is an important step and will constitute an improvement of the community's ability to evaluate the relative strengths and weaknesses of different software engineering and medical informatics methodologies when applied to health care.

6. ACKNOWLEDGMENTS

This material is based upon work supported by the NSF under awards CCF-0427071, CCF-0820198. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the NSF.

The authors gratefully acknowledge the contributions of Rachel Cobleigh, Huong Phan, Dr. Philip L. Henneman, Bin Chen, Heather Conboy, and Alexander Wise.

7. REFERENCES

[1] G. S. Avrunin, J. C. Corbett, and M. B. Dwyer. Benchmarking finite-state verifiers. *Softw. Tools for Technology Transfer*, 2(4):317–320, 2000.

[2] A. G. Cass, B. S. Lerner, J. Stanley M. Sutton, E. K. McCall, et al. Little-JIL/Juliette: a process definition language and interpreter. In *Proc. 22nd Intl. Conf. Softw. Eng.*, 754–757, 2000.

[3] B. Chen, G. S. Avrunin, E. A. Henneman, L. A. Clarke, et al. Analyzing medical processes. In *Proc. 30th Intl. Conf. Softw. Eng.*, 623–632, 2008.

[4] S. Christov, B. Chen, G. S. Avrunin, L. A. Clarke, et al. Rigorously defining and analyzing medical processes: An experience report. *MoDELS 2007 Wkshps, Springer, LNCS 5002* , 118–131, 2008.

[5] R. L. Cobleigh, G. S. Avrunin, and L. A. Clarke. User guidance for creating precise and accessible property specifications. In *Proc. 14th ACM SIGSOFT Intl. Symp. Found. Softw. Eng.*, 208–218, 2006.

[6] C. Damas, B. Lambeau, F. Roucoux, and A. van Lamsweerde. Analyzing critical process models through behavior model synthesis. In *Proc. 2009 31st Intl. Conf. Softw. Eng.*, 441–451, 2009.

[7] M. B. Dwyer, G. S. Avrunin, and J. C. Corbett. Patterns in property specifications for finite-state verification. In *Proc. 21st Intl. Conf. Softw. Eng.*, 411–420, 1999.

[8] M. B. Dwyer, L. A. Clarke, J. M. Cobleigh, and G. Naumovich. Flow analysis for verifying properties of concurrent software systems. *ACM Trans. Softw. Eng. Methodol.*, 13(4):359–430, 2004.

[9] S. Konrad and B. H. C. Cheng. Real-time specification patterns. In *Proc. 27th Intl. Conf. Softw. Eng.*, 372–381, 2005.

[10] S. T. Mark and M. A. Musen. A flexible approach to guideline modeling. In *Proc. AMIA Symp.*, 420–424, 1999.

[11] M. Peleg, A. Boxwala, O. Ogunyemi, Q. Zeng, et al. GLIF3: The evolution of a guideline representation format. In *Proc. AMIA Symp.*, 645–649, 2000.

[12] M. Peleg, S. W. Tu, J. Bury, P. Ciccarese, et al. Comparing computer-interpretable guideline models: A case-study approach. *JAMIA*, 10:2003, 2002.

[13] Y. Shahar, S. Miksch, and P. Johnson. The Asgaard project: A task-specific framework for the application and critiquing of time-oriented clinical guidelines. In *A. I. in Med.*, 29–51, 1998.

[14] *Computational Technology for Effective Health Care: Immediate Steps and Strategic Directions.* Nat. Acad. Press, 2009.

[15] A. ten Teije, M. Marcos, M. Balser, J. van Croonenborg, et al. Improving medical protocols by formal methods. *A. I. in Med.*, 36(3):193–209, 2006.

[16] A. van Lamsweerde. Formal specification: A roadmap. In *The Future of Software Engineering*, 147–159. 2000.

[17] J. M. Wilkinson and K. V. Leuven. Procedure checklist for administering a blood transfusion. http://davisplus.fadavis.com/wilkinson/Procedure_Checklists/PC_Ch36-01.doc.

[18] J. M. Wilkinson and K. Van Leuven. *Fundamentals of Nursing.* F. A. Davis Company, 2007.