Experience Modeling and Analyzing Medical Processes: UMass/Baystate Medical Safety Project Overview

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

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

Bin Chen
Dept. of Computer Science
University of Massachusetts
Amherst, MA 01003
chenbin@cs.umass.edu

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

Philip L. Henneman
Dept. of Emergency Medicine
Tufts-Baystate Medical Center
Springfield, MA 01199
philip.henneman@bhs.org

Lucinda Cassells
Baystate Regional Cancer Program
Baystate Medical Center/Tufts University School of Medicine
3400 Main Street, Springfield, MA 01107, USA
lucy.cassells@bhs.org

Wilson Mertens
Baystate Regional Cancer Program
Baystate Medical Center/Tufts University School of Medicine
3400 Main Street, Springfield, MA 01107, USA
wilson.mertens@bhs.org

ABSTRACT

This paper provides an overview of the UMass/Baystate Medical Safety project, which has been developing and evaluating tools and technology for modeling and analyzing medical processes. We describe the tools that currently comprise the Process Improvement Environment, PIE. For each tool, we illustrate the kinds of information that it provides and discuss how that information can be used to improve the modeled process as well as provide useful information that other tools in the environment can leverage. Because the process modeling notation that we use has rigorously defined semantics and supports creating relatively detailed process models (for example, our models can specify alternative ways of dealing with exceptional behavior and concurrency), a number of powerful analysis techniques can be applied. The cost of eliciting and maintaining such a detailed model is amortized over the range of analyses that can be applied to detect errors, vulnerabilities, and inefficiencies in an existing process or in proposed process modifications before they are deployed.

Categories and Subject Descriptors

D.2.4 [**Software Engineering**]: Software/Program Verification; J.3 [**Life and Medical Sciences**]: Health

General Terms

Reliability, Verification

IHI'10, November 11–12, 2010, Arlington, Virginia, USA.

1. INTRODUCTION

As noted in the 2009 US National Research Council report about healthcare [20], "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." The Medical Safety Project, a collaboration between the University of Massachusetts and Baystate Medical Center, is trying to address this concern by developing and evaluating technology to support process improvement for healthcare related processes.

Specifically we have developed a modeling language, called Little-JIL [1], that is designed to support the kinds of processes that frequently arise in healthcare. These processes often involve normative and non-normative situations, concurrent activities, and several different types of agents, such as doctors and nurses with various specialties, software systems such as electronic health records, and medical devices such as "smart" infusion pumps. The Little-JIL language is intended to enable the detailed specification of a wide range of process details, while still supporting the flexibility and freedom of choice that human agents expect. In addition, it is a language with well-defined semantics so that the resulting process definitions can be rigorously analyzed.

The process analysis technologies that we have applied to these process definitions to date include a model checker, a failure modes and effects analyzer, a fault tree analyzer, and a discrete-event simulator. Figure 1 provides an overview of

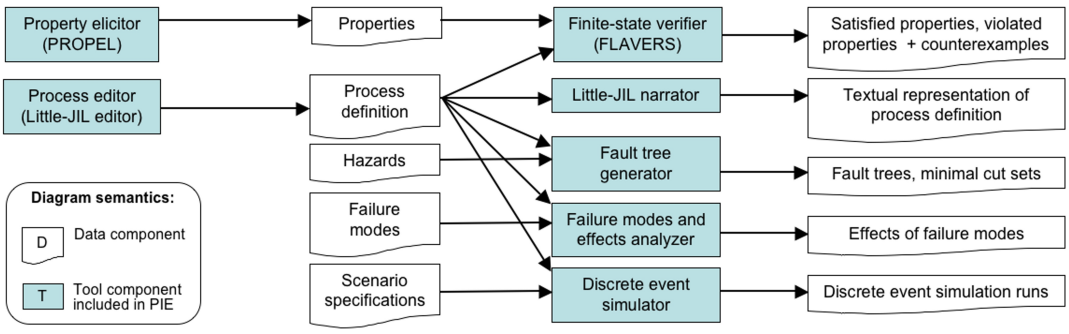

Figure 1: Architecture of the Process Improvement Environment.

how these technologies have been integrated into the Process Improvement Environment (PIE). As discussed in the next section, creating the process definition is a labor-intensive activity that must be done with considerable care. We leverage that investment by first carefully evaluating the process definition for accuracy and then by applying the different PIE analyzers to it to provide feedback about possible errors, vulnerabilities, or inefficiencies in the process.

In many respects, our work supports the classical cycle for continuous process improvement as introduced by Shewhart [18] and effectively applied by Deming [8]. The essence of this approach is to capture the process to be improved, compare its characteristics to those that are desired, identify weaknesses and shortcomings, propose and evaluate improvements, and then incorporate those improvements in the process to complete the improvement cycle and form the basis for a subsequent improvement cycle. This cycle has been renamed and modified in various ways over the past decades. In all of its manifestations, it has relied essentially on the ability to understand the process, understand its desired properties, and analyze the ways in which the process does or does not adhere to those properties. Typically, these analyses have been obtained informally. Recent research, including our own, has shown that processes and properties can be defined with precise notations and evaluated for various kinds of consistency using automated reasoning approaches. It is this rigorous approach for defining properties and properties, along with powerful reasoning techniques, that we bring to medical processes.

Most of the tools depicted in Figure 1 have been described in the literature. The contribution of this paper is to describe the overall environment and, by using related examples, show how the individual components interact with and benefit from each other. We also report on the lessons learned to date in evaluating this approach. We are currently involved in three extensive case studies: Breast Cancer Chemotherapy [3, 4], In-Patient Blood Transfusion [2, 5, 11, 12], and Emergency Room Patient Flow [17]. These case studies contain one common subprocess, Verify Patient ID [13], which is found in still broader classes of medical processes, and which we have also studied in isolation. The next section of this paper describes the Process Improvement Environment and provides a description of each of the major components. Section 3 discusses lessons learned from using this approach. Section 4 provides pointers to related work associated with the components and highlights the advantages of the technology we selected. We conclude with a brief summary of the benefits of this process-improvement approach and directions for future work.

2. PROCESS MODELING AND ANALYSIS ENVIRONMENT

There are four columns to Figure 1. The leftmost column includes the tools used to create the process definition (the Little-JIL editor) and property specification (the PROPEL property elicitor). In addition to the resulting process definition and property specifications, the second column indicates the other information that is needed to support each of the analysis capabilities, which are shown in the third column. The fourth column indicates the outputs from these analysis capabilities. We expect other capabilities to be added as we, or others, discover or develop promising technologies.

The following subsections describe each of the components in the architecture and present examples from the chemotherapy case study.

2.1 Little-JIL Process Definition Language

A Little-JIL process definition consists of a coordination specification, a resource specification, and an artifact specification. Here we are primarily concerned with the coordination specification, which describes the ordering of tasks associated with the process, the agents responsible for accomplishing those tasks, and the communication and coordination among those tasks. Figure 2 shows the top-level diagram of a Little-JIL definition of a chemotherapy process.

The main building block in the Little-JIL process language is the step. A Little-JIL step corresponds to a task, a unit of work, and is iconically represented by a black bar. Little-JIL steps can be hierarchically decomposed into substeps to an arbitrary level, depending on the amount of detail desired. In Figure 2, for example, the root step *perform chemotherapy process* is decomposed into the two substeps *create and process consult note* and *prepare for and administer chemotherapy*, which are then individually decomposed further. The left most substep of *prepare for and administer chemotherapy*, named *perform consultation and assessment*, is decomposed but in a separate diagram, as shown in Figure 3. Both Figure 2 and Figure 3 use ellipses to indicate that subsequent decomposition details have been removed.

Prerequisites (or postrequisites) can be associated with any step and indicate what should be true before the step starts (or after the step completes its associated task). Req-

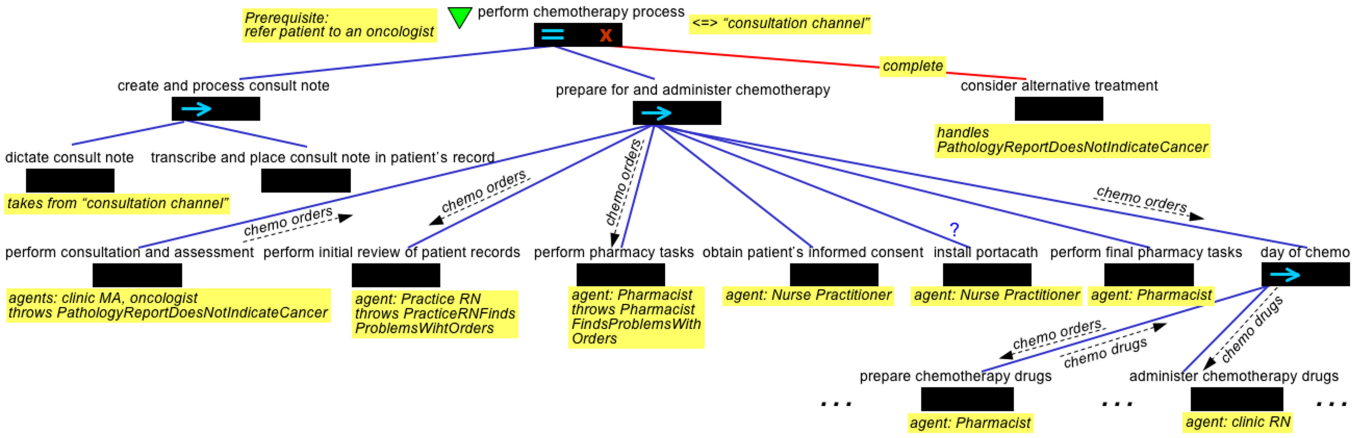

Figure 2: An example chemotherapy process.

uisites are defined by elaborated Little-JIL diagrams and thus can be arbitrarily complex, but must return a Boolean value. In Figure 2, the root step *perform chemotherapy process* has a prerequisite (indicated by the darkened (or green) triangle pointing downwards) specifying that performing the chemotherapy process can start only after the patient has been referred to an oncologist.

For non-leaf steps (i.e., steps with substeps), the left hand side of the step icon indicates the order in which its substeps are to be executed. *Perform chemotherapy process* is a parallel step (indicated by the equal sign on the left of the step bar), which means that its substeps can be executed in any order, including in parallel. There is, however, an additional constraint on when the step *create and process consult note* can start execution. Its leftmost substep, *dictate consult note*, retrieves an artifact from the *consultation channel* and cannot start until another step places an artifact in this channel. In Little-JIL, channels are used for communication and synchronization of parallel threads of execution. The *consultation channel* is declared at the root step and is thus visible by all the direct and indirect descendants of the root step. The step *dictate consult note* can be executed only after the step *perform patient consultation* (shown in Figure 3) completes and writes to the channel. The channel is thus used to specify that an oncologist cannot dictate the consult note before consulting the patient. But, since the consult note is primarily used for billing and legal purposes and does not directly affect the patient's treatment, the doctor may choose to dictate the consult note right after performing the patient consultation or later, while the tasks in *prepare for and administer chemotherapy* are already underway.

The step *prepare for and administer chemotherapy* is a sequential step (indicated by the arrow in the step bar), which means that its substeps need to be executed in left to right order. These substeps represent the main phases of the chemotherapy process and illustrate the variety of agents involved in it. The activities involved in the step *perform consultation and assessment* (which is further elaborated in Figure 3) require a clinic medical assistant, who obtains the patient's history and measures patient's height and weight, and an oncoloist who performs the patient consultation and creates a treatment plan and medication orders. The step *perform initial review of patient records* is performed by a practice registered nurse (RN), who reviews the treatment plan and orders created by the oncologist. The step *perform pharmacy tasks* is done by a pharmacist, who also verifies and approves the treatment plan and orders. A nurse practitioner then obtains the patient consent and optionally (indicated on the diagram by the question mark which represents an optional step) installs a portacath, if the patient needs one. On the day before the actual administration of chemotherapy, a pharmacist performs final tasks, such as preparing drug bins for the next day and making sure that patient orders are not missing. Finally, on the day of chemotherapy, a pharmacist and a clinic RN carry out the tasks needed to administer chemotherapy to a patient.

Figure 3 shows how the exception handling features of Little-JIL can be used to capture non-typical scenarios. During execution of the step *perform consultation and assessment*, if the oncologist finds that the patient's biopsy does not indicate cancer, the exception *PathologyReportDoesNotIndicateCancer* is thrown. In this case, it is not necessary to administer chemotherapy; the oncologist considers an alternative treatment and the chemotherapy process is thus considered completed. In Figure 2, this is represented by the exception being propagated up the step tree until a matching exception handler is found. The step *consider alternative treatment* is the matching handler and it gets executed in response to the exception. The continuation after the handling of the exception is *complete* (indicated at the connector between the exception handler *consider alternative treatment* and the root step), which means that the step *perform chemotherapy process* is completed after the handler *consider alternative treatment* is done. Little-JIL supports several other continuation semantics. Exception handlers are regular steps, which means that they can be decomposed to an arbitrary level of detail and throw exceptions themselves.

The final tasks in *performing consultation and assessment* are to create a treatment plan and medication orders. Note, the step *create treatment plan* is a choice step (indicated by the circle with a horizontal line located in the step bar), which in this case means that the oncologist can choose which substeps to execute—the oncologist can either use a careset to generate a treatment plan or choose to create a treatment plan from scratch.

The full details of the language are beyond the scope of

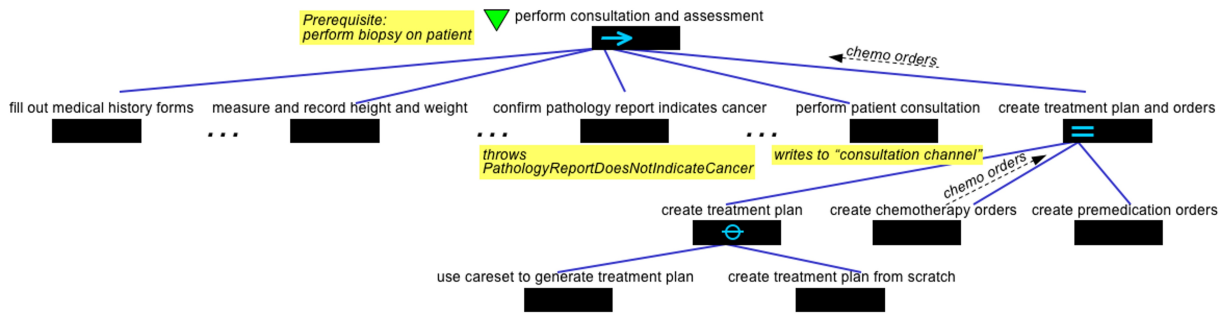

Figure 3: Elaboration of *perform consultation and assessment.*

this paper, but from the above description, it should be clear that the relatively concise representation shown in Figure 2 and Figure 3 models many complexities of the process. If one were to create a data or control flow representation of the process definition, it would be significantly larger and more difficult to follow.

Just as a program defines all the legal executions for that program, a process definition defines all the possible ways in which the process can be performed. A particular execution of the process corresponds to a trace through the process definition in which all the encountered pre and post requisites and conditional checks are satisfied. Medical processes are often quite complicated, especially when the possible exceptional conditions and concurrent behaviors are taken into account. It is these complexities that are often the source of errors and, thus, they need to be accurately captured in the process definition and evaluated carefully.

2.2 Process Narrative Generation

It has been our experience that healthcare professionals quickly become comfortable reviewing Little-JIL diagrams. Still, there are many subtle details reflected in a diagram that might be overlooked by the casual observer. The Process Narrative Generator is a tool that helps domain experts understand the details of a Little-JIL definition by providing a textual representation.

Figure 4 shows part of the textual description automatically generated by the Little-JIL Narrator from the process definition in Figures 2 and 3. Currently, there are four main components of a Little-JIL textual description: a table of contents, a main body section, an index of step names, and a legend. The *table of contents* is presented on the left side of the document and shows an outline of the step hierarchy and the type of each step (sequential, parallel, etc.) using the same iconography as the Little-JIL diagrams. In addition, if a step is an exception handler, the continuation action after handling the exception is also shown.

The *main body section* is presented on the right side of the document. It consists of a subsection for each step in the Little-JIL process definition. Each subsection provides a natural language description of the information associated with the step, such as the artifacts used or produced, resources required, the agent responsible for executing the step, and exceptions that the step can throw.

Since including all the information about each step may make the textual description rather long, the user can choose to hide the display of different kinds of information. To facilitate navigation, the Little-JIL textual description is a

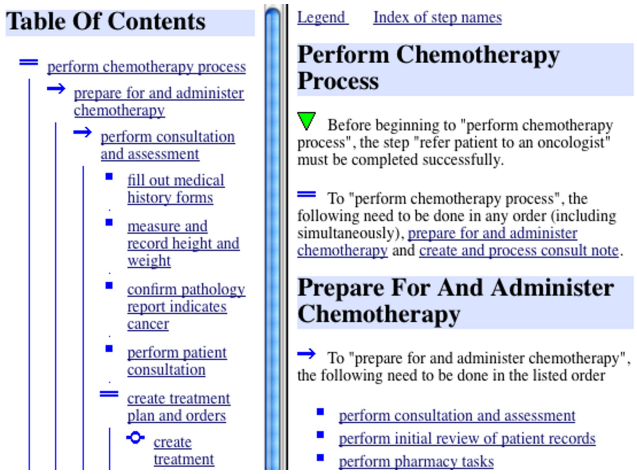

Figure 4: Part of the generated textual description of the Little-JIL chemotherapy process definition.

hyperlinked document. The user can, for instance, quickly explore the step hierarchy in the table of contents, click on a step name and be directed to the corresponding step section in the main body of the document. Similarly, the user may be reading the description for a step in the main body and click on one of its substeps to see the detail for that substep.

2.3 Requirements Specification

Properties are the requirements that are intended to be satisfied by the process. In PIE, each property is represented as a finite-state automaton (FSA). The events in the automaton must correspond to events in the process definition. In fact, determining what are the important properties for a process helps frame the scope and the granularity of the process definition. That is, if there is an important property that should hold for a process, then the details about the activities associated with that property should be represented in the process definition. For example, it may not be important to specify how a form is filled in, but it might be important to indicate that it was reviewed and signed by the attending physician. A property would indicate that such a signature is required before a medical procedure could commence, and the process model would provide the details of how this is to be accomplished within the process. As described below, we use model checking to determine if a property holds on all traces through the process.

An FSA describes the acceptable (or unacceptable) order-

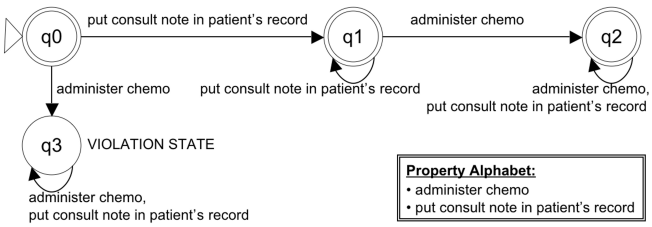

Figure 5: An example chemotherapy property.

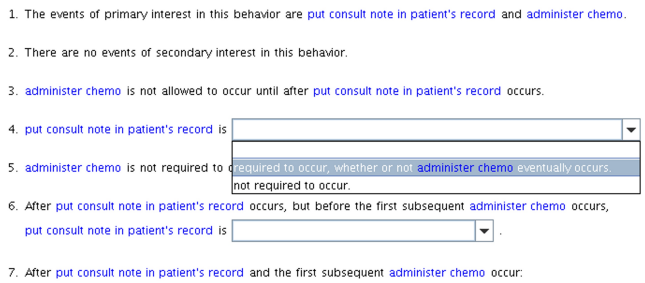

Figure 6: Partially instantiated DNL representation of the example property.

ing of selected events in any trace through the process. Consider the property that states: "Before chemotherapy can be administered to a patient, that patient's consult note needs to be put in that patient's record." The events of interest to this property are *administer chemo* and *put consult note in patient's record*. The FSA, shown in Figure 5, represents that the event *administer chemo* is not allowed to occur until after the event *put consult note in patient's record* has occurred. If *administer chemo* does occur before *put consult note in patient's record*, the FSA is driven from the initial state *q0* to the violation, non-accepting state *q3*. If, however, *put consult note in patient's record* precedes *administer chemo*, this corresponds to a process execution that satisfies the property and thus the FSA is driven to the accepting state *q2*.

It is surprisingly difficult to correctly encode all the important but subtle details about a property. For the property shown in Figure 5, for example, one has to decide if there needs to be a consult note if chemotherapy is not administered (represented as "no" by having state *q0* be an accepting state), or if the patient must get chemotherapy if the consult note is present (again, represented as "no" by having state *q1* be an accepting state).

Propel, for PROPerty ELucidator [6], is a system that helps users tease out and consider each of the issues associated with a property. Starting from a high-level natural language description of a property, Propel asks questions about the relationship among the identified events of interest until the appropriate property pattern [9] is ascertained. At that point, Propel can provide several alternative views of the pattern with the options that must be decided explicitly highlighted. The views include a question tree view where the user continues to be presented with questions about the property, a graphical view of the FSA with optional aspects of the representation (e.g., transitions, labels, and accepting states) highlighted for selection, and a disciplined natural language (DNL) view where the user selects the appropriate phrase associated with each option from a pull down menu. Figure 6 shows a partially instantiated DNL representation for the example consult note property with the pull down menu showing the choices associated with making the second state (the state after it has been determined that the consult is present) an accepting state or not.

After all the options associated with a pattern have been selected, the resulting property can be viewed as an FSA or as a natural language statement. Similar to the narrative view of a Little-JIL process definition, the resulting text is rather verbose, but it does accurately describe the selected property as represented by an FSA. The FSA formally depicts the sequences of events from the selected events of interest that satisfy (or do not satisfy) the property. This formal statement is the basis for rigorous analysis to determine if the process definition does indeed adhere to the property on all its possible traces .

2.4 Model Checking

Model checking is an analysis approach used to evaluate hardware and software systems. Model checking approaches typically take (or create) a model of the system and then evaluate if all possible traces through the model adhere to a specified property. If so, the approach has proven that the model satisfies the property. If not, a counterexample, a trace through the process definition that demonstrates the violation, is provided. By examining the counterexamples, users can usually determine the cause of the problem, which could be associated with the process definition, the property specification, or an actual error in the process that is being modeled.

When dealing with complex systems, the computation cost of this analysis can be prohibitive, and thus a range of optimization approaches have been developed. We have developed a translator that accepts a Little-JIL definition and a property of interest and then creates a highly-optimized model that can be used with the FLAVERS [10] and SPIN [14] model checkers. For the medical case studies we are investigating, we have been able to prove a number of important properties and have discovered some important errors, not only in the process definition and properties, but in the actual process as well.

The property shown in Figure 5 provides a good example of a property that is violated by the process definition given in Figures 2 and 3. FLAVERS found that there is an execution trace through the process on which the event *administer chemo* (bound to the step *administer chemoterapy drugs* in Figure 2) happens before the event *put consult note in patient's record* (bound to a substep of *transcribe and place consult note in patient's record*) has occurred. This is indeed possible since the process steps *administer chemotherapy drugs* and *file consult note in patient's record* can happen in parallel as they are descendents of the steps *prepare for and administer chemotherapy* and *create and process consult note* (in Figure 2), which can in turn happen in parallel.

This property violation reflects an interesting situation in the real-world chemotherapy process. After an oncologist performs the patient consultation, the oncologist needs to dictate the consult note. A transcriber, who is sometimes external to the hospital, listens to the dictation and then sends the transcription of the consult note back to be reviewed by the oncologist before being filed by a staff member

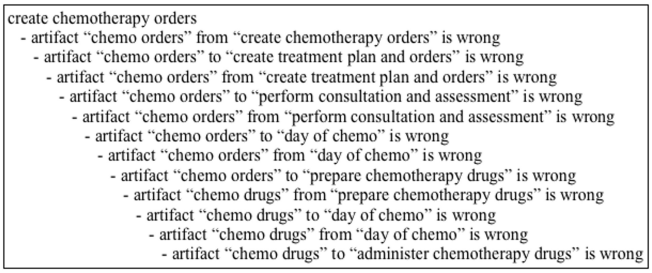

Figure 7: An example FMEA tree view result for the step *create chemotherapy orders*.

in the patient record. In addition to involving various people, this subprocess can be started quite late. Consult notes are often used for legal rather than strictly medical purposes and often doctors don't dictate them right after the patient consultation, but sometimes days or weeks after that. At the same time, the rest of the phases of the chemotherapy preparation and administration process can start right after the oncologist performs the patient consultation and creates the treatment plan and medication orders. Thus, it is sometimes possible that the chemotherapy drugs can be administered to a patient before the consult note makes it to the patient's record.

Another example of an interesting error occurred in the Blood Transfusion case study where a deadlock was detected. We did not expect deadlock to arise in human-intensive processes, but this was indeed the case. In certain circumstances, the bloodbank would be waiting on a type and screen from a nurse and the nurse would be waiting on a confirmation from the bloodbank. When this situation occurs, the nurse eventually notices that the patient had been waiting a long time and calls the bloodbank to check on the status of the request. The deadlock is then manually resolved by those involved and the process continues. Thus, there was some awareness that this situation could arise but not a good understanding of how the process could be changed to eliminate this problem. Our analysis detected the deadlock and provided counterexamples to help explain when it occurred. Interestingly, the initial correction proposed by the medical professionals, while eliminating the deadlock, could sometimes lead to a long delay in the transfusion. Our tools detected this, and verified that a different process modification eliminated both the deadlock and the delay. This experience points out how complicated medical processes can be and the benefits of using analysis techniques to help reason about them.

2.5 Failure Mode and Effects Analysis

Failure Mode and Effects Analysis (FMEA) is a hazard analysis technique that can be used to evaluate the impact of individual failures on the overall system. A failure mode represents a specific case in which some part of the system fails to meet its intent or requirements.

Figure 7 shows an example output from applying the FMEA tool to the Little-JIL chemotherapy process definition in Figures 2 and 3. The first line in Figure 7 shows the step *create chemotherapy orders* and the second line shows one potential failure mode of that step, namely that it produces the wrong *chemotherapy orders*. The rest of Figure 7 shows the propagation of the effects of that failure mode. The last line is particularly interesting because it indicates that one of the potential effects of this failure mode is that the step *administer chemotherapy drugs* takes as input the wrong chemotherapy drugs! Since chemotherapy drugs are highly toxic, this can have fatal consequences in the real-world process and therefore its consideration is of practical significance.

Identifying potential effects of a set of failure modes using FMEA often raises other interesting questions. For example, given an effect of a failure mode one might wonder if there are other failure modes that can lead to this effect. An analysis technique that can answer such questions and that complements FMEA in our Process Improvement Environment is Fault Tree Analysis (FTA).

2.6 Fault Tree Analysis

Fault Tree Analysis [22] is a hazard analysis technique used to systematically identify and evaluate all possible causes of a given hazard. A hazard in a safety critical system is "a state or set of conditions of the system that, together with certain other conditions in the environment, will lead inevitably to an accident" [15] . Given a potential hazard in a system, FTA deductively identifies events (component failures, human errors, etc.) in the system that could lead to the hazard and produces a fault tree, which provides a graphical depiction of all possible parallel and sequential combinations of those events. Once a fault tree has been derived, qualitative and quantitative analysis can be applied to provide information, such as specific sequences and sets of events that are sufficient to cause a hazard. This information can then be used as guidance for improvements to the design of the system.

Figure 8 shows an example of a fault tree automatically generated from the Little-JIL definition of the chemotherapy process shown in Figures 2 and 3. Fault trees can become quite large and because of this, we have developed some optimization techniques to help reduce the size of the generated trees. The hazard in the fault tree in Figure 8 (which is shown as the root node A) is that the *chemo drugs* are wrong when they are input to the step *administer chemotherapy drugs*. The fault tree indicates that this can happen when the *chemo drugs* that are output from the step *prepare chemotherapy drugs* are wrong. The OR-gate underneath node B indicates that this can happen when the step *prepare chemotherapy drugs* produces wrong chemo drugs (node C) or when the *chemo orders* output by the step *perform pharmacy tasks* (node D) are wrong. The AND-gates underneath node D indicate that the *chemo orders* output by the step *perform pharmacy tasks* are wrong only when the events in nodes G, H, and F all happen.

In addition to automatically generating fault trees from Little-JIL process definitions, the Fault Tree Analyzer can also automatically find minimal cut sets. A *minimal cut set* is a set of events that together can cause the hazard to occur, whereas none of its proper subsets is sufficient for this to be true. There are two minimal cut sets for the fault tree shown in Figure 8: {Step "prepare chemotherapy drugs" produces wrong "chemo drugs"} and {Step "create chemotherapy orders" produces wrong "chemo orders", Exception "PracticeRNFindsProblemWithOrders" is not thrown by step "perform initial review of patient records", Exception "PharmacistFindsProblemWithOrders" is not thrown by step "perform pharmacy tasks"}.

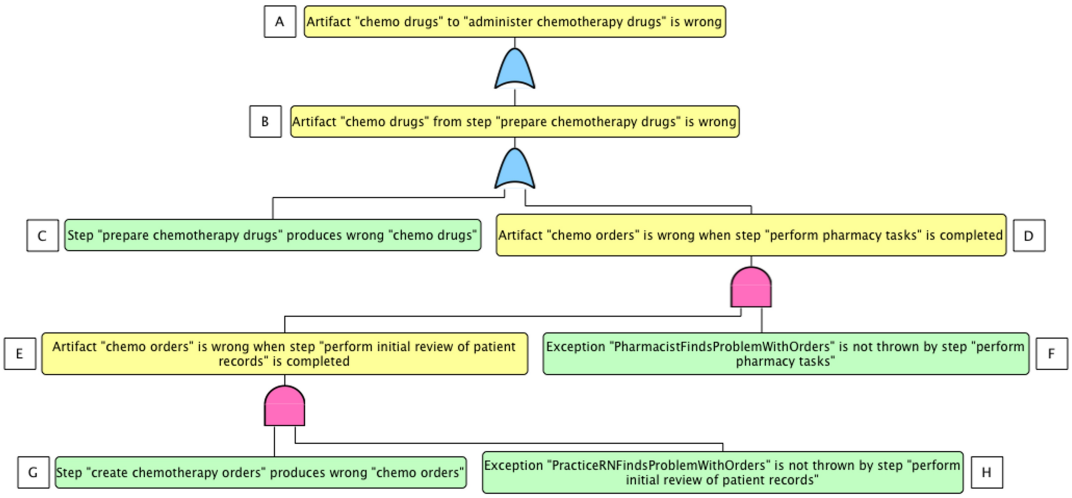

Figure 8: A fault tree automatically generated from the Little-JIL chemotherapy process definition.

The interpretation of the second minimal cut set is that for the hazard to occur, the oncologist who creates the orders must create wrong orders (the step *create chemotherapy orders* in Figure 3 outputs the wrong *chemo orders*), and then the Practice RN and the Pharmacist who check these orders must fail to detect the problem with the orders (the steps *perform initial review of patient records* and *perform pharmacy tasks* do not throw the exceptions indicating the detection of a problem with the orders). Since the pharmacist uses the orders to prepare the chemotherapy drugs on the day of chemo, having the wrong orders can lead to preparing and then using the wrong chemotherapy drugs. All three medical professionals—the oncologist, the practice RN and the pharmacist—must make mistakes for this hazard to occur, indicating that there are several redundancy checks in place to ensure that it is not easy for the hazard to occur.

The size of the first minimal cut set, however, is only 1. Its interpretation is that even if the chemo orders are correct when the pharmacist performs the step prepare chemotherapy drugs (see Figure 2), the pharmacist can still make a mistake and prepare the wrong drugs (perhaps by taking a drug with a similar name by accident or by accidentally taking a drug meant for another patient). A minimal cut set of size 1 indicates a single point of failure in the process definition and raises concerns that need to be discussed with the medical professionals involved in the process.

2.7 Simulation

The previously described analysis techniques focus on errors and vulnerabilities that can be detected statically, without the need for execution. Simulation, however, is an analysis approach that can use dynamic execution state to study a complementary range of issues. The Little-JIL discrete event simulator (JSim) takes as input a Little-JIL process definition along with specifications of how to simulate the behaviors of agents and a model of the resources. The initial application of this capability has been to study how different resource mixes and allocation strategies affect patient length of stay (LOS) in a hospital Emergency Department. This was done by simulating the processes used to treat patients, using a range of different patient arrival rates and a range of different combinations of Emergency Department resources (i.e. the doctors, nurses, and beds available for treating patients).

Essential to this work has been an innovative approach to scheduling resources, based upon an articulate model of those resources. Modeling the resources needed in healthcare domains, such as a hospital Emergency Department, is particularly challenging because the resources are extremely varied, ranging from human agents such as doctors, to hardware such as beds and X-ray devices, to software systems such as electronic health record systems, and to consumable resources such as blood units. Moreover, the roles that different resources can play, and the process steps that they can perform, vary with circumstances and perspectives. Thus, for example, a nurse may be unable to prescribe medication under ordinary circumstances, but may do so in an emergency; a bed may be needed to treat a patient, but under some circumstances a wheelchair might be used instead. If a discrete event simulation is to be used to project outcomes such as patient LOS, then details and constraints such as these need to be taken into account appropriately.

To address these challenges we have developed the ROMEO resource manager designed to support accurate specification of the constraints and the dynamic nature of the behaviors of healthcare resources, and TWINS, an incremental resource scheduling system [23] designed to support the kind of scheduling flexibility that replicates what actually happens in a highly disruption-prone environment, such as a hospital Emergency Department. A ROMEO specification of a healthcare resource consists of a set of Little-JIL steps that the resource is potentially capable of performing, where each step is "guarded" by a specification of the circumstances under which the step can actually be performed. Using this specification information ROMEO then accepts requests for the resources needed to perform a step and makes a decision about which resources (if any) are to be assigned to the step. This decision is guided by consideration of the simulated process's current status, attributes of the currently available resources, and upcoming resource requirements as projected by TWINS. TWINS uses a genetic algorithm to

Process Definition	Total # steps	# exceptional situations	# exception handling related steps (% total)
Chemo	467	59	207 (44%)
BT	248	15	37 (15%)

Table 1: Process definition size information for the Blood Transfusion and Chemotherapy case studies.

search the sizeable space of possible resource allocations for upcoming steps and selects the one that promises the most effective overall use of those resources.

A key hypothesis of our approach is that more detailed and precise system specifications lead to more accurate predictions. For instance, in a study aimed at determining the value of detail in the specification of resources, we executed simulations using resource specifications that did, and then did not, include specifications of the time periods during which resources could be expected to be available. The results from this study suggest that, when the patient arrival rate is high, resource contention increases and more precise resource descriptions provide better support for scheduling. With lower patient arrival rates, resource contention decreases and less precise resource descriptions support resource schedules that are increasingly close to those obtained with more precise resource descriptions. A similar study of simulations of versions of the process with varying degrees of concurrent activity also showed improvements in accuracy as the degree of concurrency approached what the medical professionals describe as realistic levels.

3. LESSONS LEARNED

Our major focus to date has been on developing promising basic technologies and using case studies to evaluate their applicability to medical processes. Future work is needed to evaluate their effectiveness in improving actual medical outcomes.

Table 1 provides data on the size of two of the largest process definitions we created and the number of exceptional situations that were considered. We found that the elicitation of the processes and properties took a considerable amount of time. For each of these case studies, more than a month of effort spread out over a year or more was spent eliciting and representing the process. A similar amount of time was spent on elicitation and specification of the properties. The elicitation was usually done by teams of two or more graduate students and one or more medical professionals.

The remainder of this section presents a brief summary of some of the more salient lessons learned.

Process elicitation and process languages.

The process language guided the elicitation process. In particular, rich semantic features of the Little-JIL language, such as requisites and exceptions, encouraged process definers to ask questions about corresponding aspects of processes. This helped elucidate process features, such as exception management that are important but are often overlooked during open, unstructured interviews. In fact, many of the difficulties we encountered in creating accurate process definitions arose because initial versions based on open, unstructured interviews were too restrictive or did not adequately capture all behaviors (e.g. exception management). Indeed we found that medical guidelines or protocols usually lack such critically important details. We did not encounter many real-time constraints, but perhaps this was because timing constraints are currently not a strong feature in Little-JIL.

A visual representation of the process definition seemed useful. The medical professionals who participated in the case studies were not computer savvy and were initially leery about having to learn a process language. Having a visual representation that is relatively easy to understand seemed to help considerably. Being able to provide restricted views of the process definition also seemed to help, since low-level details about the process definition, such as the artifacts that needed to be provided to process activities, could be suppressed until the higher-level views of the major tasks and their decompositions were agreed upon. Having a natural language representation of the process was also extremely helpful, since it further lowered the technology hurdle for the medical professionals. Even some of the computer scientists found it useful to read through the English description to find inconsistencies or to convince themselves that the definition was accurate. Real processes are large and complex, however, so both visual and textual representations can quickly become ungainly, suggesting the need to develop effective process definition summarization technologies.

As has been observed with software development, the very act of trying to model a process leads to improved understanding of that process, to discovering process defects, and to identifying possible process improvements. Sometimes during elicitation, misunderstandings between participants arose. In one situation it was determined that an artifact that was being created was not being used subsequently. In another situation, a deadlock that involved two different medical professionals was discovered while eliciting the chemotherapy process. A nurse needed to obtain the height and weight of a patient to be able to verify drug dosages and to subsequently sign the patient's treatment plan. There was a hospital rule, however, that an appointment scheduler must wait for a signed treatment plan before scheduling a visit at which the patient's height and weight could be verified. Thus, having both the nurse and the scheduler follow the rules creates a deadlock situation, which was actually being observed in clinical practice. To break the deadlock, some nurses were signing an unverified treatment plan so that the scheduler could schedule a visit at which height and weight could be verified. The signing of an unverified and potentially incorrect treatment plan is dangerous, however, because it may mislead other medical professionals and perhaps result in the administration of the wrong dosage. The discovery of this deadlock led to a modification in the actual clinical process.

Property specification.

It was important to keep the properties we elicited and studied focused on high-level goals that should be true regardless of the process actually implemented. For example, most of our properties must be adhered to whether or not an electronic health record is used. That is, the source of the information might change, but not the ways it must be used.

We found that it is difficult to capture medical process requirements accurately and completely because they often must take into consideration a variety of subtle issues. ProPEL helped elucidate many of these issues, just as Little-JIL

helped elucidate analogously subtle but important process issues. We found, however, that we needed to extend PROPEL to explicitly deal with exceptional behavior. For example, it might be important to be able to specify a property requiring that information be obtained from the patient but only if the patient is conscious.

The elicitation of process requirements led to the discovery of errors in both the process model and in the process itself. It also helped define the scope and the granularity of the process model. We elicited some important properties that involved particular events that were sometimes not suitably specified in process definitions. Our finding that process and property elicitation complement each other in interesting and important ways has led us to believe that neither should be undertaken without the other.

Analysis tools and associated technologies.

As expected, most of the defects identified in model-checking of medical processes were defects in the process definitions, rather than in the processes themselves. Nevertheless, these errors helped us improve the process definitions, which in itself is important since the definitions were the basis for subsequent analyses. Model checking did, however, find some important and surprising process errors. For example, the violation of the consult note property described earlier was a reflection of a problem in the real chemotherapy process. Moreover, the processes we studied are sufficiently complex that it was not immediately clear how best to remove these errors without causing violations of other properties. Analysis of the process definition, which is an abstraction of the real process, thus facilitated consideration of improvements to the real process.

As researchers have noted in other contexts, FTA and FMEA are complementary analytic approaches, with the latter helping to expose potential hazards and the former exploring the ways in which such hazards might arise. Whether the hazards are suggested by FMEA or based on professional experience, fault tree analysis seems to be a promising approach for identifying process vulnerabilities due to incorrect performance of process activities. It usually takes considerable care to create a single fault tree, but with our approach considerable care is taken to create the process definition that is then used to automatically create fault trees for a number of potential hazards.

It is hard to model the complex ways in which resources are used in healthcare processes, but modeling healthcare process resources in detail seems to improve the accuracy of simulations of these processes. Such simulations should make it possible for proposed process modifications to be evaluated before being implemented.

Some overall lessons and observations.

Although the elicitation of accurate healthcare process definitions is difficult and expensive, the costs can be amortized by the use of such definitions in a variety of analyses. In addition to the several anecdotal examples of the benefits of process modeling and analysis mentioned earlier, this approach impacted the real-world processes and the medical professionals involved in several other ways. One of the senior pharmacists involved in the chemotherapy process took the detailed natural language version of the Little-JIL process definition and used it as a guide in his daily practice and as an aid in training junior pharmacists. One of the nurses involved in the specification of the blood transfusion process definition and properties changed the way she teaches the process to nursing students since the standard blood transfusion checklists in textbooks turned out to be less detailed and precise compared to the Little-JIL process definition.

This work has convinced us that continuous process improvement is indeed applicable in the domain of healthcare. A key benefit of rigorous analysis of process definitions is its value in identifying the presence or absence of defects in proposed new or modified processes prior to their deployment in actual practice.

4. RELATED WORK

Space limitations do not permit us to do a thorough review of the work on modeling and analysis or their application to medical processes. Here, we highlight just some studies that we deem most relevant.

There has been a significant amount of work on developing notations for modeling medical processes and guidelines. These notations aim to provide semantics for capturing features specific to the medical domain, such as representing human and automated agents, providing support for specifying real time constraints and for the integration of domain ontologies and patient records. The relative strengths and weaknesses of several of these notations have been explored by Peleg et al. [16].

In addition to modeling medical processes, there has been recent work on applying automated analysis techniques to models of medical processes. In [7], a cancer therapy process is modeled with UML message sequence charts that are then translated to labeled transitions systems (LTS) and the LTS analyzer is used to reason about properties of the model. In [21], a medical protocol is modeled in the Asbru language and the KIV theorem prover is used to reason about properties of this model.

5. CONCLUSION AND FUTURE WORK

The UMass/Baystate Medical Safety Project has been developing and evaluating tools and technologies for modeling and analyzing medical processes. It has drawn upon technology originally developed to support continuous improvement of software development processes that integrate the efforts of teams of software designers, coders, testers, etc., developing a variety of software artifacts. Applying this work to selected medical processes has demonstrated its value while also exposing needs for improvement, some of which can benefit software development as well as other domains.

We recognize that the described process-centered approach requires an ongoing investment in developing and maintaining process models. We believe, however, that this will yield benefits worth the investment. In our studies, the team members involved gained valuable insight about real-world medical processes at every step, starting with process definition and property specification, and continuing with each analysis applied. Errors were uncovered at each step. Even when the errors were in the introduced artifacts (i.e., the process model or property), finding these errors improved the accuracy of these artifacts and the subsequent analyses based on them. We think it is particularly important to emphasize that this investment enables proposed process modifications to be carefully vetted before being introduced.

Although we believe that the current process improve-

ment environment provides significant benefits, we also believe that there are major gains to be made from future extensions. For example, we are currently exploring incorporating probabilistic modeling and analysis approaches as well as support for specification and analysis of real time constraints. One major new thrust of our research will be to use carefully validated process models as the basis for providing real-time guidance during the actual performance of medical processes. Of course, great care must be taken to assure that the guidance is useful, presented effectively, and can be easily ignored without negative consequences. Research needs to be done to find the sweet spot between support and annoyance, always keeping in mind the central goal of improved medical outcomes. Real-time guidance will also be used to pursue another key goal, namely lessening the burden of documenting medical processes. This seems particularly promising, as useful documentation could be created automatically by monitoring process progress, another area of interesting future research. Longitudinal studies of such monitored process behavior could also provide valuable insight. For example, such studies could suggest the most effective responses to certain exceptional situations. Finally, we note that although we have described our overall approach with the help of process examples drawn from a hospital setting, we believe that the approach is applicable to processes carried out in a range of environments, from hospitals, to home care, to nursing facilities.

We do not expect all medical processes to be amenable to the rigorous process improvement approach that we advocate, but we expect that there are many that will be. Moreover, we suggest that many processes may be carried out in broadly similar ways in many different locations, but with differences that appear only in lower level details. For such processes, we expect that the hierarchical decomposition supported by Little-JIL should facilitate carrying out analyses of the high-level process models. The results of these analyses would then be augmented by separate analyses of the specialized lower-level process models. Pursuing this suggestion is yet another area of future investigation.

6. ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Awards CCF-0820198, CCF-0905530 and IIS-0705772, and by a Gift from the Baystate Medical Center, Rays of Hope Foundation. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the NSF.

The authors gratefully acknowledge the contributions of Tiffany Chao, Rachel Cobleigh, Heather Conboy, Huong Phan, Mohammad Raunak, Danhua Wang, Alexander Wise, and Junchao Xiao.

7. REFERENCES

[1] A. G Cass, B. S. Lerner, S. M. Sutton, Jr, et al. Little-JIL/Juliette: a process definition language and interpreter. In *22nd Intl. Conf. on Softw. Eng.*, p. 754–757, 2000.

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

[3] S. Christov, B. Chen, G. S. Avrunin, et al. Formally defining medical processes. *Methods of Information in Medicine. Special Topic on Model-Based Design of Trustworthy Health Information Systems*, 47(5):392–398, 2008.

[4] S. Christov, B. Chen, G. S. Avrunin, et al. Rigorously defining and analyzing medical processes: An experience report. *Models in Softw. Eng.: Wkshps and Symposia at MoDELS 2007, Reports and Revised Selected Papers*, p. 118–131, 2008.

[5] S. C. Christov, G. S. Avrunin, L. A. Clarke, et al. A benchmark for evaluating software engineering techniques for improving medical processes. In *Intl. Conf. on Softw. Eng., Wkshp. on Softw. Eng. in Health Care*, 2010.

[6] R. L. Cobleigh, G. S. Avrunin, and L. A. Clarke. User guidance for creating precise and accessible property specifications. In *14th ACM SIGSOFT Intl. Symposium on Foundations of Softw. Eng.*, p. 208–218, 2006.

[7] C. Damas, B. Lambeau, F. Roucoux, et al. Analyzing critical process models through behavior model synthesis. In *31st Intl. Conf. on Softw. Eng.*, p. 441–451, 2009.

[8] W. E. Deming. *Out of the Crisis*. MIT Press, Cambridge, 1982.

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

[10] M. B. Dwyer, L. A. Clarke, J. M. Cobleigh, et al. Flow analysis for verifying properties of concurrent software systems. *ACM Transactions on Softw. Eng. and Methodology*, 13(4):359–430, 2004.

[11] E. A. Henneman, G. S. Avrunin, L. A. Clarke, et al. Increasing patient safety and efficiency in transfusion therapy using formal process definitions. *Transfusion Medicine Review*, 21(1):49–57, 2007.

[12] E. A. Henneman, R. Cobleigh, G. S. Avrunin, et al. Designing property specifications to improve the safety of the blood transfusion process. *Transfusion Medicine Reviews*, 22(4):291–299, 2008.

[13] P. L. Henneman, D. L. Fisher, E. A. Henneman, et al. Patient identification errors are common in clinical simulation. *Annals of Emergency Medicine*, 2009.

[14] G. J. Holzmann. *The SPIN Model Checker*. Addison-Wesley, 2004.

[15] N. Leveson. *Safeware: System Safety and Computers*. Addison-Wesley, 1995.

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

[17] M. Raunak, L. Osterweil, A. Wise, et al. Simulating patient flow through an emergency department using process-driven discrete event simulation. In *1st ICSE Wkshp. on Softw. Eng. in Health Care*, p. 73–83, 2009.

[18] W. A. Shewhart. *Economic control of quality of manufactured product*. D. Van Nostrand Company, Inc, 1931.

[19] D. H. Stamatis. *Failure Mode and Effect Analysis: FMEA from Theory to Execution*. American Society for Quality, 1995.

[20] W. W. Stead and H. S. Lin, editors. *Computational Technology for Effective Health Care: Immediate Steps and Strategic Directions*. Nat. Acad. Press, 2009.

[21] A. ten Teije, M. Marcos, M. Balser, et al. Improving medical protocols by formal methods. *Artificial Intelligence in Medicine*, 36(3):193–209, 2006.

[22] W. Vesely, F. Goldberg, N. Roberts, et al. *Fault Tree Handbook (NUREG-0492)*. U.S. Nuclear Regulatory Commission, 1981.

[23] J. Xiao, L. J. Osterweil, Q. Wang, et al. Dynamic resource scheduling in disruption-prone software development environments. In *13th Intl. Conf., FASE 2010*, 2010.