Enhancing The Quality of Medical Processes Using Software Engineering Methodologies

[1]Muhsina Muhammed,[2]Meenu shaji, [3]Jissymol Jiji

MCA Student, MCA Student, MCA Student
Department of Computer Sciences,
Santhigiri College of Computer Sciences, Vazhithala, Thodupuzha, India

Abstract: In this paper we describe some of the key observations resulting from our work on using software engineering technologies to help to detect errors in medical processes.In many ways medical processe are similar to distributed systems in their complexity and proneness to contain errors.We have been investigating the application of a continuous processes in which detailed and semantically rich models of the medical processes are created and then subjected to rigorous analysis. The technologies we applied helped to improve the understanding about the processes and led to the detection of errors,

Terms: Software Engineering, Medical Processes, Modelling Processes

I. INTRODUCTION

Medical processes are similar to distributed systems in their complexity and proneness to contain errors. Here is the review about the application of continuous process improvement approach to medical process it contains semantically rich models of medical processes and which is subjected to rigorous analysis. The technologies we applied help to improve understanding about the process and led to detection of errors and improvement to those processes. This paper summarizes the reduction of errors in medical care. There is surprising similarity between healthcare systems and software systems, Healthcare systems typically involve many different types of human agents (e.g., doctors and nurses with different specializations and roles, pharmacists, lab technicians, and support staff), hardware devices (e.g., infusion pumps, radiation therapy machines, and patient monitoring devices), and software applications (e.g., computerized physician order entry systems. decision support systems, and electronic medical records).

The University of Massachusetts Medical Safety project has been investigating how software engineering technology, originally developed to improve the quality of software systems, could be effectively applied to improving the quality of medical processes. They have undertaken several case studies of validation tools for error detection. These case studies have involved developing models of healthcare processes that are unusually detailed and meaningfully broad. Analyse these process model using finite state verification and other analysis technique and working with professionals to improve the process when errors occurs.

The approach to process improvement that we have been developing is based on creating detailed, semantically rich models of the processes and then applying a number of different analysis techniques to try to detect errors in those processes.

II. LITERATURE SURVEY

The approach to process improvement that we have been developing is based on creating detailed, semantically rich models of the processes. With help from the medical professionals, they are the case domain experts create models and applied analysis techniques to those models if any errors occurs finding out that from where these errors occurs and provide solution through any modification. The model of the modified process is then carefully reevaluated to assure that it has successfully dealt with the uncovered error and has not introduced new errors. This approach provides a technological basis for process improvement applied not just to healthcare systems but also more broadly to other classes of human-intensive systems.

Here they present some observations that derive from their experiences using software engineering modeling and analysis techniques to create, validate, and improve medical processes. This project has been using specific software technologies to gain a deep understanding of medical processes and the nature of medical process improvement problems.

III. METHODOLOGY USED IN MODELING PROCESS

Specifically uses **Little-JIL** process definition language to model the processes, the **PROPEL** is an approach to explain properties and represent properties, **FLAVERS and SPIN** are **FINITE STATE VERIFICATION** systems which detect defects in the Little-JIL process.

Little-JIL

Little-JIL is a new language for programming the coordination of agents and is an executable, high-level process programming language with a formal syntax and rigorously defined operational semantics. First approach Little-JIL provides a rich set of control structures while relying on separate systems for support in areas such as resource, artifact, and agenda management. The second is that processes can be executed by agents who know how to perform their tasks but can benefit from coordination support. Use of the Little-JIL process definition language in this project has suggested the importance of certain features such as support for abstraction, hierarchical decomposition, concurrency, exception handling, and operations designed to support the flexibility that human agents expect for doing their tasks (e.g., non-deterministic choice). In addition, it is important that the language have well-defined semantics so that process models described in the language can be rigorously analyzed.

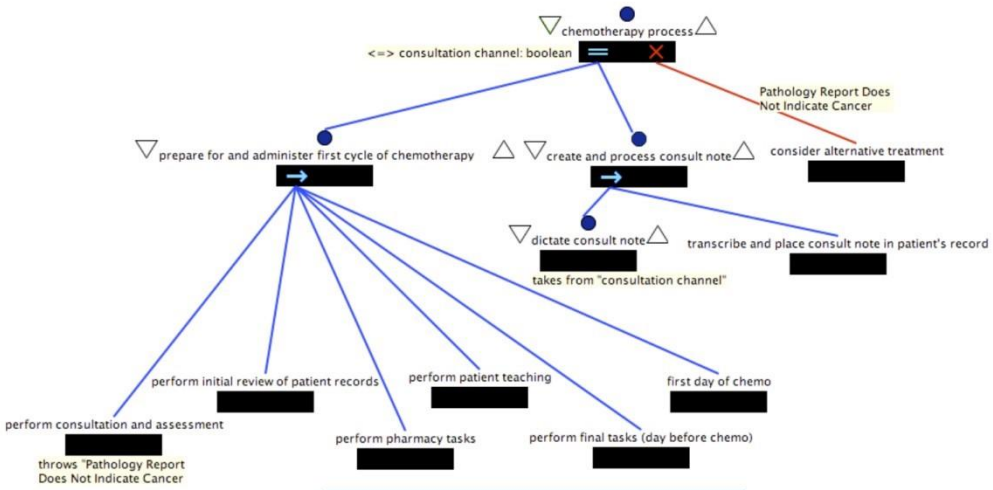

Figure 1. Top-level chemotherapy process definition in Little-JIL.

Figure 1 depicts a Little-JIL definition addressing a portion of a chemotherapy process which shows the view of top level process tasks, represented as steps here the root step is decomposed into two sub steps which executes parallel and the sub step further divided into leaf steps--Figure 1 shows that the root step i.e chemotherapy process and has a sub step consider alternative treatment that acts as an exception handler (indicated by "X") on the chemotherapy process step bar to which the step consider alternative treatment is connected.

While in the case of perform consultation and assessment, if the doctor determines that the patient's pathology report does not indicate cancer, the Pathology Report Does Not Indicate Cancer exception is thrown. The thrown exception does propagate until it reaches the matching handler. Step-sequencing specifications provide control over the order of step execution, Little-JIL also enables specification of synchronization through such constructs as a channel. In this figure a channel is used to specify that a doctor cannot dictate a consult note before evaluating the patients condition.

Thus, this example shows that the "dictate consult note "step can potentially execute in parallel with task in " prepare for and administer first cycle of chemotherapy " step. Process Modeling and Elicitation: Carefully modeling processes leads to better understanding about those processes. It was usual that the medical professionals did not fully understand the processes in which they were participants. Usually they knew their tasks, but often had misunderstandings about what others involved in the process actually did or how artifacts were used. By understanding the process better, errors in the process sometimes become apparent. Thus, the activity of modeling a process often leads to the discovery of process errors and always leads to better understanding.

MULTIFACETED LANGUAGE—LITTLE-JIL

Little-JIL language is multifaceted, in that the step definition has many different aspects.

For example, in defining a step in the process, one has to consider the preconditions, the postconditions, the exceptions that could be thrown or handled by the step, the artifacts that are input to or output from the step, resources that might be requested or released, which agents should execute the step, the sub steps that comprise the step, and the order in which these sub steps should be executed etc. Its not necessary to consider all these facets. . In their project, a first pass was made to understanding and representing the process involving step decomposition and control flow. Later passes to address other facets. Adding these additional facets, however, often resulted in changes to the overall step decomposition and control flow. Such changes are unavoidable, From the above list of facets, it is clear that Little-JIL supports the specification of a relatively broad range of semantic features of a process. It support for specifying some aspects such as concurrency, exception handling, scoping, late binding, and flexible control flow that supported the freedom of choice often desired by human agents. Thus, the language supported detailed specification of how exceptional conditions are identified and handled, how parallel tasks must be synchronized, and how the assignment of personal tasks is indeed.

Abstraction and hierarchical decomposition facilitates developing the process models incrementally. The process modelers have the choices about how many levels to which to decompose a task and the level of abstraction or granularity of that decomposition. Abstraction allows the activities associated with a task to be conceptualized.

IV. ANALYSIS BASED ON EXPERIMENT

Analysis is the corner stone of this approach if the system is interesting enough to warrant being modeled then the model is probably complex enough to warrant careful scrutiny by rigorous and automated analysis techniques. Without such scrutiny one should have serious concerns about the validity of the model and any decisions made based on that model

As the models are repeatedly validated using a range of analysis techniques, we increase our confidence in their accuracy. If decisions made using the models then fail to provide the expected results, that is a form of validation that should result in care full scrutiny to determine the cause and subsequent improvement to the model. The analysis considering here are *fault tree analysis*---to reveal vulnerabilities ,if steps in the process are not executed appropriately, *finite state verification*--to determine, if all traces through a model follows the properties that indicate the legal sequences of events, , *discrete-event stimulation*--to determine the aggregate behavior after a large number of traces have been executed. Each of the analysis provide different and distinctive kinds of feedback. Here mostly experienced with finite state verification.

Example:

Determining if it is the "right patient" for that might require checking that the name on the ordered medical procedure matches the name on the wrist-band ID, matches the name on the medical chart, and matches the name and date of birth provided by the patient assuming the patient in the conscious state . These refined requirements were still not precise enough to form the basis for verifying the processes.

The **Propel system** was designed to help elicit these types of details from domain experts and then to represent them as a finite-state automaton that can serve as the basis for finite-state verification. These low-level, detailed, and narrowly focused requirement specifications are frequently called properties. Before verification could be done using these properties, the events mentioned in the properties had to be mapped to events in the process.

These low-level, detailed, and narrowly focused requirement specifications are frequently called properties. Before verification could be done using these properties, the events mentioned in the properties had to be mapped to events in the process. This mapping was usually straightforward, but needed to be done with care, since an event in the property might map to different events in the process definition. Taken together, these properties and the bindings of event names to step names provided a detailed and rather process-specific description of the overall requirements for the process.

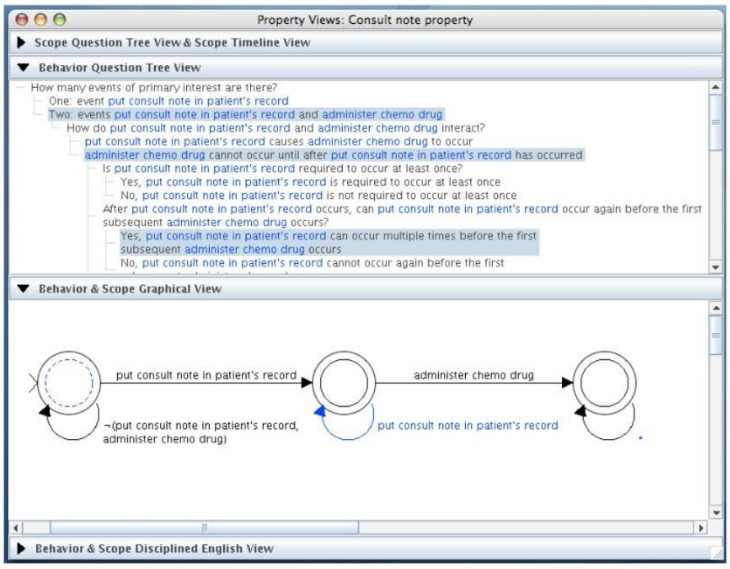

Figure 3. Screen shot of the PROPEL question tree and finite-state automaton templates used during the development of a property.

Its an example of the finite-state automaton view and the question tree view for a simple property that was developed to assure that there is a signed consult note in the patient's record before chemotherapy is administered. To represent the properties, Propel provides templates for commonly occurring verification property patterns [10]. These templates explicitly indicate the options that need to be considered for each pattern Propel provides the specifier with three alternative representations of the templates: disciplined, natural language text where the options are represented as phrase choices; finite-state automata graphs where options are represented by optional transitions, labels, and accepting states; and by question trees that first select the appropriate pattern based on answers to a few initial questions, but then continue to pose questions about all the options associated with the selected pattern template.

The medical professionals initially had difficulty understanding the difference between a process model, an operational view, and a requirement or property specification medical professionals tended to think in terms of "war stories" about what went wrong. We could sometimes map such a story to an appropriate set of properties. More work is definitely needed to determine how to better exploit these war stories, which are examples of scenarios that led to process errors. *Developing the properties provided valuable feedback about the current process model*. We usually started developing the process models before the properties, based on the medical guidelines and the information provided by the medical experts who were working with us. We made note, however, of any requirements that were mentioned during this process elicitation. Those requirements, plus existing guidelines or protocols, were the initial set of high-level requirements. *specification and handling of exceptions have been particularly interesting, and often problematic*

Most of the errors that we have found in the process models and in the processes themselves involve exceptions errors are most likely to occur during the handling of exceptional cases. This leads to variation in how medical professionals respond to these situations and, consequently, is more likely to lead to errors. The analysis tools that they used should be significantly improved to handle exceptional cases better.

Finite-state Verification: Finite-state verification problems are known to often explode in size, making it impractical to analyze large systems. These optimizations are conservative but often reduce the size of the model by introducing some imprecision.(lack ogf accuracy).Using finite-state verification, we mostly found errors in the process models, as opposed to errors in the actual processes themselves. Before process models can be used for decision-making, they should be carefully validated. Using analysis techniques, we found numerous defects in our process models, but we also found some interesting and subtle errors in the actual processes that could have serious consequences. Analysis of the process models were also useful in helping to determine the causes of these errors and in evaluating alternative potential solutions for correcting them

The set of property specifications associated with a process model continued to grow as the process was modified, as defects were found in the process models, and as errors were found in the processes themselves or in the clinical setting.

Other Analysis

Fault tree analysis: Fault-tree analysis creates a tree representation of how a hazard could occur in terms of what would need to fail or be faulty. The trees are then translated into Boolean flow equations that can be evaluated to determine minimum cut sets. Each minimum cut set indicates the collection of failures that would have to occur together for the hazard to arise. Drawback it must not be sufficiently complete and accurate fault-trees can be derived automatically from Little-JIL process definitions, using template. The resulting fault trees are surprisingly large and complicated. The process model should have first undergone rigorous validation before the corresponding fault-tree was generated and analyzed. Carefully validated process model can be used to derive a very large number of fault trees fault tree for a single hazard derived from blood transfusion process is given in figure 4.

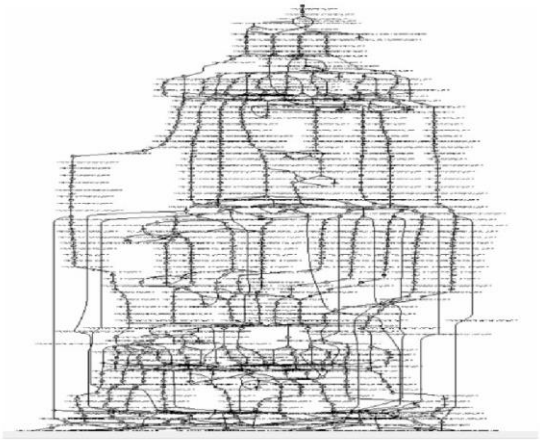

Figure 4. Example of a fault tree for a single hazard derived from a medical process.

Discrete-event simulations: Simulations can be used to evaluate performance and efficiency. For emergency room case study, one of the issues is how to determine the best resource mix that delivers optimal flow and minimal waiting times. For solving this problem simulations are used.

V. DISCUSSION

They have described how software engineering technologies and approaches can be incorporated into a Deming Cycle of continuous improvement to medical processes. Currently we are involved in three case studies emergency room case study, blood transfusion, and chemotherapy administration, each of which has different characteristics and places different demands on the supporting technology that we are developing. This work has led to a number of observations that seem particularly important. They want to know about how medical errors can occur, and how to guard against them. For definitive reasoning the process modeling language itself must have well-defined semantics. additional specification complexity is added when the model incorporates execution semantics.

Within their opinion, if a process is complicated enough to warrant precise and detailed modeling then the accuracy of the model requires careful scrutiny.This is particularly true for detailed models, such as the models they developing using Little-JIL. Their experience suggests that these sorts of detailed and complex process models should be developed incrementally so that highlevel, more-abstract views of the process can be validated before more-detailed models are developed. The scope and granularity of the model should be determined by the questions. There is no doubt that detailed models require more effort to develop and maintain, but provide more definitive, in-depth feedback

VI. FUTURE ENHANCEMENTS

This work has already suggested many directions for future research. A number of these directions are suggested by attempts to use process models to introduce automation. we would like to eventually use validated process models to guide medical professionals while they are actually executing their processes in a clinical setting. Doctor's hand held device could indicate the current process status for each patient and highlight the most urgent items according to the most recent recommended protocols. There are human interface issues in how to represent this information on the handheld device so that it can be immediately understood and used. How to determine and maintain coherence between the actual process and the process model. reduce the number of medical errors and help improve the efficiency of medical care. Medical protocols change frequently, however, so at least the generic versions of these models would need to be updated regularly and then re-customized.

REFERENCES

List and number all bibliographical references in 10-point Times, single-spaced, at the end of your paper. When referenced in the text, enclose the citation number in square brackets, for example: [1]. Where appropriate, include the name(s) of editors of referenced books. The template will number citations consecutively within brackets [1]. The sentence punctuation follows the bracket [2]. Refer simply to the reference number, as in “[3]”—do not use “Ref. [3]” or “reference [3]”. Do not use reference citations as nouns of a sentence (e.g., not: “as the writer explains in [1]”).

Unless there are six authors or more give all authors’ names and do not use “et al.”. Papers that have not been published, even if they have been submitted for publication, should be cited as “unpublished” [4]. Papers that have been accepted for publication should be cited as “in press” [5]. Capitalize only the first word in a paper title, except for proper nouns and element symbols.

For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation [6].

1. G. Eason, B. Noble, and I. N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955. *(references)*
2. J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.
3. S. Jacobs and C. P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.
4. K. Elissa, “Title of paper if known,” unpublished.
5. R. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in press.
6. Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interface,” IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982]. M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.