# An Integrated Collection of Tools for Continuously Improving the Processes by Which Health Care Is Delivered: A Tool Report

Leon J. Osterweil, Lori A. Clarke, and George S. Avrunin

Department of Computer Science
University of Massachusetts, Amherst, MA 01003
{ljo,Clarke,avrunin}@cs.umass.edu

**Abstract.** This report will present a collection of tools that supports the precise definition, careful analysis, and execution of processes that coordinate the actions of humans, automated devices, and software systems for the delivery of health care. The tools have been developed over the past several years and are currently being evaluated through their application to four health care processes, blood transfusion, chemotherapy, emergency department operations, and identity verification. The tools are integrated with each other using the Eclipse framework or through the sharing of artifacts so that the internal representations generated by one might be used to expedite the actions of others. This integrated collection of tools is intended to support the continuous improvement of health care delivery processes. The process definitions developed through this framework are executable and intended for eventual use in helping to guide actual health care workers in the performance of their activities, including the utilization of medical devices and information systems.

## 1 Introduction

Many reports (e.g., [1]) have suggested that manifest shortcomings in the delivery of health care could be addressed by the application of appropriate information technologies. Because health care is often delivered through complex processes, involving the coordination of several different types of agents, process technologies should be useful in addressing some of these shortcomings. Our interest in the application of process technology to various domains is longstanding and has resulted in the creation of a growing collection of tools that have been of value in addressing problems in such diverse areas as software development, dispute resolution, and digital government. Our preliminary results in applying them to health care have been encouraging.

## 2 Our Approach

Broadly speaking our approach entails applying the notion of Continuous Process Improvement (CPI) to human-centric processes, such as those that often arise in health care. For such processes, our approach to CPI requires 1) defining a process

S. Rinderle-Ma et al. (Eds.): BPM 2009 Workshops, LNBIP 43, pp. 647–653, 2010.

precisely, 2) analyzing the process definition to determine the presence or absence of defects or vulnerabilities, 3) modifying the process definition to address identified defects as well as possible, and 4) observing the execution of the modified process to determine whether past defects have been removed or whether new defects are revealed. The last three steps may be repeatedly revisited.

This view has led us to develop process technologies that address needs in the areas of

- Process elicitation and definition
- Requirements specification and process analysis
- Process execution, simulation, and monitoring

# 3 Our Toolset

## 3.1 Process Elicitation and Definition Tools

The tools in this category revolve around the Little-JIL process definition language. Little-JIL incorporates features that are seldom found in current process modeling languages, but that seem important for the clear and precise definition of health care processes. Thus, for example, Little-JIL provides powerful support for exception management, which has been important in the clear, concise, and complete definition of processes as diverse as emergency division management and chemotherapy delivery. Language support for specifying concurrency and synchronization has been useful in representing interactions among doctors, nurses, and blood banks in blood transfusion processes. Little-JIL's emphasis on the use of abstraction and modularity has led to process definitions that foster reuse, clarity, and conciseness. The language's support for precise specification of artifact flow and resource utilization to complement specifications of activity coordination has been particularly useful in defining emergency division processes. A full description of Little-JIL can be found in [2]. Language features such as those just summarized seemed to us to offer advantages over existing process modeling languages that justified our extra work in defining and implementing the language, and in building tool support needed to make good use of it. bu

Little-JIL is rigorously defined by means of finite-state machine semantics, is supported by a visual interface, and has been used extensively to support the definition of processes in diverse domains. The development of health care processes is described in several papers (e.g. [3-5]), and will be demonstrated with the support of the Visual-JIL screen editor, which facilitates the creation and editing of Little-JIL process definitions. In recognition of the value of complementary forms of these process definitions, our toolset also includes facilities for displaying these definitions as linear trees and as natural language hypertext, both of which will be shown along with other tools that support the creation and display of various cross-reference reports and summaries.

In recent work we have been comparing detailed patient identification process definitions to determine how well these process definitions actually reflect the way that these processes are actually performed by real medical practitioners. We are doing this by comparing observed traces of actual patient identification process executions to Little-JIL process definitions hoping to improve the process definitions, and also improve our process elicitation procedures as well.

### 3.2 Analysis

A major goal of our work has been to demonstrate the feasibility of doing rigorous analysis of process definitions. This is possible only when processes are defined in a rigorously defined language. In defining our own process language we were also able to define its semantics. Little-JIL's semantics are sufficiently well-defined to support a variety of static analyses, of which two types will be demonstrated.

**Finite-state Verification.** We will demonstrate how our FLAVERS finite-state verification tool [7] can be used to determine the presence or absence of defects in Little-JIL process definitions (at least up to a fixed bound loop iterations) [8]. FLAVERS compares all possible execution traces through a Little-JIL process definition to a Finite-state Automaton (FSA) definition of a property. When the FSA defines a desirable property (e.g. a safety property), then the FLAVERS analysis determines whether it is possible for there to be an execution of the process that violates the desirable property. When such a violation occurs, FLAVERS provides a counter example trace that demonstrates how the property can be violated. We have used FLAVERS to look for such process defects as allowing the possibility that a blood transfusion may be done without prior to obtaining informed consent, and administering chemotherapy without confirming that dosages have been computed with up-to-date height and weight information.

We will also demonstrate how our PROPEL tool [9] can be used to define FSAs that are then suitable for use by FLAVERS. PROPEL supports the specification of FSAs in three formats, disciplined natural language, question trees, and FSA depictions. The user can work with any one of the formats and have the results displayed in the others, or can work with the three formats simultaneously. We have used PROPEL to create properties such as the two described above that were the basis for actual finite state verifications [4].

**Fault Tree Analysis.** We will also demonstrate how Fault Trees can be generated from Little-JIL process definitions [10], and show how their analysis can be used to complement the finite-state verifications generated using FLAVERS. Finite-state verification can be used to identify defects in a process definition that result from the occurrence of undesired sequences of process events (e.g. beginning a transfusion before patient consent has been obtained). But these analyses assume that all process steps have been carried out correctly. We will demonstrate two types of analyses, Fault Tree Analysis (FTA) and Failure Mode and Effects Analysis (FMEA), that are centered on Fault Trees, and that explore the ramifications of incorrect performance of process steps.

We have generated an FTA from a Little-JIL definition of a blood transfusion process, and a specification of a specific hazard (the delivery of an incorrect type of blood to the patient bedside). We then used our FTA generation tool to generate the minimal cut sets (MCSs) of this FTA, and used it to show how these MCSs can indicate vulnerabilities such as single points of failure, namely single steps whose incorrect performance can lead directly to specified hazard. We also support FMEA, using it to show the consequences of a step (e.g. getting a blood type) being performed incorrectly. FTA and FMEA are complementary analyzes; one shows the possible

clusters of failures that could lead to a hazard and the other shows the hazards that could result from a particular failure.

### 3.3 Execution, Simulation, and Monitoring

We will also demonstrate some of the dynamic analysis capabilities in our toolset, including tools that support the actual execution of processes and tools that support simulations of process executions.

**Process Execution tools.** Little-JIL processes are defined hierarchically. Each step can be thought of as an instance of a procedure to which arguments and resources are bound at runtime. Child steps of a parent step should be thought of as subprocedures that can be invoked by the parent. When a leaf step is invoked, the actions associated with that step are to be performed by agents in any way that the agent may desire. Thus, in an important sense, the Little-JIL process definition is a specification of the order in which steps are to be executed, the assignment of specific agents to the execution of steps, and the flow of artifacts between agents as steps are assigned and executed. Our toolset contains a variety of tools needed to support such process execution. These tools include a manager of the resources that are to be used to perform process steps, an agenda manager that supports the distribution of tasks to (and back from) agents, and a subsystem that maintains effective communication between human participants and the other components of a running process. These tools and their coordination will be demonstrated.

**Process discrete event simulation tools.** We will also demonstrate how a Little-JIL process definition can be used to drive a discrete event simulation [11]. Many of the tools needed to support such simulations are already used to support process execution. But discrete event simulation also requires the creation of simulators of the behaviors of the agents that perform the various process steps. We will demonstrate the tools needed to simulate various agents. Perhaps of greatest interest are those tools that simulate the performance of humans in processes. Our demonstration will show how such human performance can be simulated at relatively basic levels, but still be sufficient to demonstrate the effects of varying resources. We have used this capability to begin explorations of how to vary the mix of human resources in a hospital Emergency Department in order to reduce patient waiting time, while keeping costs low.

Process-centered environment. Figure 1 is a conceptual view of how the capabilities of these tools can complement each other while supporting the overall goal of Continuous Process Improvement (CPI). The boxes in the middle of the figure represent different analyzers. Arrows to the left of these boxes represent the process definition artifacts (e.g. the Little-JIL coordination definition, and FLAVERS property specification) required by each of these analyzers. The arrows leaving these boxes represent the various analysis results produced. The figure indicates that these artifacts provide information whose consideration can be used to identify defects or shortcomings, which can then be taken as inputs to review processes that yield improvements (depicted as the long right-pointing arrows) to the process definition artifacts.

Note that this approach to process improvement amortizes the considerable cost of developing a process definition, by using that definition as input to several analyzers. Creating a detailed, accurate health care process model has in some cases taken several dozen meetings with domain experts. At a typical meeting, appropriate domain experts review the process definition for accuracy and the computer scientists prepare questions to flesh out incomplete, inconsistent, or erroneous aspects of the definition. It is most cost effective for such a large investment in time to yield the greatest return in benefits. Using the single articulate process definition to drive a number of different analyzes seems to improve the cost-benefit ratio of our approach.

## 4 Evaluation

The Little-JIL language has evolved through a series of modifications. Little-JIL version 1.5 is described in a generally available document [2]. The Visual-JIL screen editor is currently available for distribution, although some features of the 1.5 language version are not yet implemented. The FLAVERS finite-state verification system and PROPEL property elicitation system are both available for distribution. Thus it is possible to use publicly available tools to carry out finite-state verification of processes defined in Little-JIL against properties defined using PROPEL. The Fault Tree Analysis tools, the execution system, and discrete event simulation are not yet available for distribution.

Little-JIL and its toolset have undergone evaluation through their application to the definition and analysis of a range of health care processes. The language and tools have also been evaluated through their application to processes in such other domains as labor-management dispute resolution, elections [12], and scientific data processing [13]. These evaluations have resulted in modifications to Little-JIL, such as the creation of facilities to support preemption of tasks currently being executed. The evaluations have also led to changes to our analysis toolset.

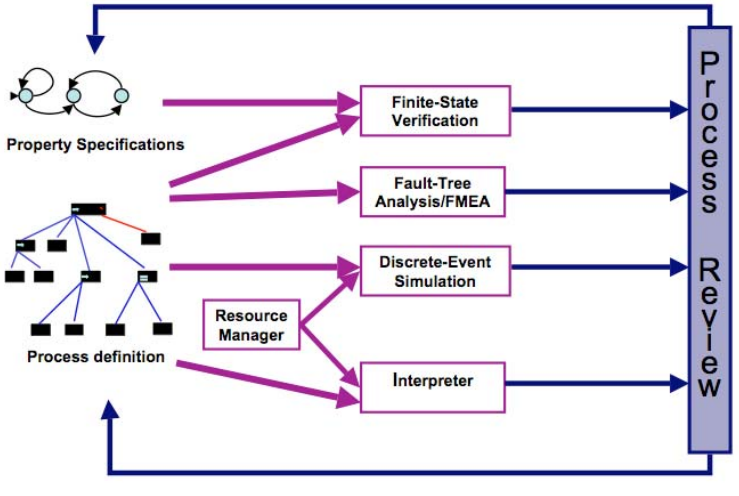


**Fig. 1.** Continuous Process Improvement Environment

## 5 Related Work

There has been some prior work in using process definition and analysis to improve medical processes. For example, the Protocure II project [14, 15] has goals that are quite similar to ours, but uses a rather different, AI-based, linguistic paradigm for defining processes. Noumeir has also pursued similar goals, but using a notation like UML to define processes [16]. Others (eg. [17]), view medical processes as workflows and use a workflow-like language to define processes and drive their execution. But, we note that these projects seem to place less emphasis on analysis.

There have been other approaches to improving medical safety, as well, but much of the emphasis of this work has been targeted towards quality control measures [18], error reporting systems [19], and process automation in laboratory settings [20], such as those where blood products are prepared for administration. In other work, Bayesian belief networks have been used as the basis for discrete event simulations of medical scenarios, and to guide treatment planning (eg. [21]).

**Acknowledgements.** The Little-JIL language and the suite of support tools described above are the products of a great deal of work by dozens of participants who are too numerous to be listed here. This work has been supported by numerous grants, including by the National Science Foundation under Award(s) CCF-0427071, CCF-0820198, and IIS-0705772. 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 National Science Foundation.

## References

1. Kohn, L.T., Corrigan, J.M., Donaldson, M.S. (eds.): To Err is Human: Building a Safer Health System. National Academy Press, Washington (1999)
2. Wise, A.: Little-JIL 1.5 Language Report. Department of Computer Science, University of Massachusetts, Amherst, UM-CS-2006-51 (2006)
3. Clarke, L.A., Avrunin, G.S., Osterweil, L.J.: Using Software Engineering Technology to Improve the Quality of Medical Processes. In: Proceedings of the Thirtieth International Conference on Software Engineering, Leipzig, Germany, pp. 889–898 (2008) (invited keynote)
4. Henneman, E.A., Avrunin, G.S., Clarke, L.A., Osterweil, L.J., Andrzejewski, C.J., Merrigan, K., Cobleigh, R., Frederick, K., Katz-Basset, E., Henneman, P.L.: Increasing Patient Safety and Efficiency in Transfusion Therapy Using Formal Process Definitions. Transfusion Medicine Reviews 21, 49–57 (2007)
5. Christov, S., Chen, B., Avrunin, G.S., Clarke, L.A., Osterweil, L.J., Brown, D., Cassells, L., Mertens, W.: Formally Defining Medical Processes. Methods of Info, in Medicine. Special Topic on Model-Based Design of Trustworthy Health Information Systems 47, 392–398 (2008)
6. Christof, S., Avrunin, G.S., Clarke, L.A., Henneman, P.L., Marquard, J.L., Osterweil, L.J.: Using Event Streams to Validate Process Definitions. U. of Mass., Amherst, UM-CS-2009-004 (2009)

7. Dwyer, M.B., Clarke, L.A., Cobleigh, J.M., Naumovich, G.: Flow Analysis for Verifying Properties of Concurrent Software Systems. ACM Transactions on Software Engineering and Methodology 13, 359–430 (2004)
8. Chen, B., Avrunin, G.S., Henneman, E.A., Clarke, L.A., Osterweil, L.J., Henneman, P.L.: Analyzing Medical Processes. In: Proceedings of the Thirtieth International Conference on Software Engineering, Leipzig, Germany (2008) (to appear)
9. Cobleigh, R.L., Avrunin, G.S., Clarke, L.A.: User Guidance for Creating Precise and Accessible Property Specifications. In: Proceedings of the 14th ACM SIGSOFT International Symposium on Foundations of Software Engineering, pp. 208–218. ACM Press, Portland (2006)
10. Chen, B., Avrunin, G.S., Clarke, L.A., Osterweil, L.J.: Automatic Fault Tree Derivation from Little-JIL Process Definitions. In: Proceedings of the Software Process Workshop and Process Simulation Workshop, Shanghai, China. LNCS, pp. 150–158. Springer, Heidelberg (2006)
11. Raunak, M.S., Osterweil, L.J., Wise, A., Clarke, L.A., Henneman, P.L.: Simulating Patient Flow through an Emergency Department Using Process-Driven Discrete Event Simulation. In: Proceedings of the 31st International Conference in Software Engineering Workshop on Software Engineering and Health Care, Vancouver, Canada (2009) (to appear)
12. Simidchieva, B.I., Marzilli, M.S., Clarke, L.A., Osterweil, L.J.: Specifying and Verifying Requirements for Election Processes. In: Proceedings of the 9th Annual International Conference on Digital Government Research, Montreal, Canada (2008) (to appear)
13. Osterweil, L.J., Clarke, L.A., Podorozhny, R., Wise, A., Boose, E., Ellison, A.M., Hadley, J.: Experience in Using a Process Language to Define Scientific Workflow and Generate Dataset Provenance. In: Proceedings of the ACM SIGSOFT 16th International Symposium on Foundations of Software Engineering, Georgia, Atlanta, pp. 319–329 (2008)
14. Marcos, M., Galán, J.C., Wittenberg, J., van Croonenborg, J., Rosenbrand, K., Martínez-Salvador, B.: Construction of a Process Model for the Integration of Formal Methods in the Development of Medical Guidelines. In: eChallenges e-2006 Conference (October 2006)
15. ten Teije, A., Marcos, M., Balser, M., van Croonenborg, J., Duelli, C., van Harmelen, F., Lucas, P., Miksch, S., Reif, W., Rosenbrand, K., Seyfang, A.: Improving medical protocols by formal methods. Art. Intell. in Medicine 36(3), 193–209 (2006)
16. Noumeir, R.: Radiology interpretation process modeling. Journal of Biomedical Informatics 39(2), 103–114 (2006)
17. Ruffolo, M., Curio, R., Gallucci, L.: Process Management in Health Care: A System for Preventing Risks and Medical Errors. In: van der Aalst, W.M.P., Benatallah, B., Casati, F., Curbera, F. (eds.) BPM 2005. LNCS, vol. 3649, pp. 334–343. Springer, Heidelberg (2005)
18. Mintz, P.D.: Quality assessment and improvement of transfusion practices. Transfusion Med. 9, 219–232 (1995)
19. Kaplan, H.S., Battles, J.B., Van der Schaaf, T.W., et al.: Identification and Classification of Events in Transfusion Medicine. Transfusion 38, 1071–1081 (1998)
20. Jensen, N.J., Crosson, J.T.: An Automated System for Bedside Verification of the Match Between Patient Identification and Blood Unit Identification. Transfusion 36, 216–221 (1996)
21. van der Gaag, L.C., Renooij, S., Witteman, C.L.M., Aleman, B.M.P., Taal, B.G.: Probabilities for a probabilistic network: A case-study in oesophageal cancer. Artificial Intelligence in Medicine 25(2), 123–148 (2002)