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, 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 are 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., IOM reports) 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 particularly 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 seem to be of value in addressing problems in such diverse areas as software development, labor-management dispute resolution, and digital government, as well as health care. Our preliminary results have been encouraging.

2. Our approach. Broadly speaking our approach entails attempting to apply the notion of Continuous Process Improvement (CPI) [1] 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 precisely, 2) analyzing the process definition to determine the presence or absence of defects or vulnerabilities, 3) modifying the process definition in response to that feedback until the defects and vulnerabilities have been addressed as best they can be, and 4) observing the execution of the process to determine whether past defects have indeed 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. This section briefly describes the tools in each of the three categories listed above.

3.1. Process elicitation and definition. The tools in this category revolve around the Little-JIL process definition language. Space does not permit a full description of this language (details can be found in [2]), but the language 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, including health care [3-5]. The development of health care processes 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 in our toolset that support the creation and display of various cross-reference reports and summaries.

This collection of process definition tools has been used to support the elicitation of process definitions in the health care domain. Our experience in doing this, however, has also sharpened our awareness of the continuing uncertainty about how precisely process definitions actually reflect the details of real processes. In very recent work, we are comparing observed traces of actual process executions to Little-JIL process definitions [6]. We will also demonstrate some preliminary results of our efforts to quantify the closeness of these traces to these definitions. The intention of this work is to facilitate the systematization and sharpening of our process elicitation efforts.

3.2. Analysis: Our motivation for defining Little-JIL's semantics rigorously has been primarily aimed at assuring that the language could be used effectively to support the rigorous analysis of processes defined in Little-JIL. This has resulted in the development of a two main types of static analysis tools that will also to be demonstrated.

3.2.1. 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 certain bound on the loops) [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 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 collections of properties for our medical case studies [4].

3.2.2. 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 will demonstrate the generation of an FTA from a Little-JIL process definition and a specification of a specific hazard (e.g. the delivery of an incorrect type of blood to the patient bedside). We will also demonstrate how our tools can then be used to generate the minimal cut sets (MCSs) of such an FTA and how these MCSs can indicate vulnerabilities such as single points of failure, namely single steps whose incorrect performance can lead directly to specified hazards. We will also demonstrate our support for FMEA. For a single potential failure, such as a step being performed incorrectly, the potential resulting hazards can be shown. Thus, 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.

3.3.1. 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.

3.3.2. 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.

4. 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.

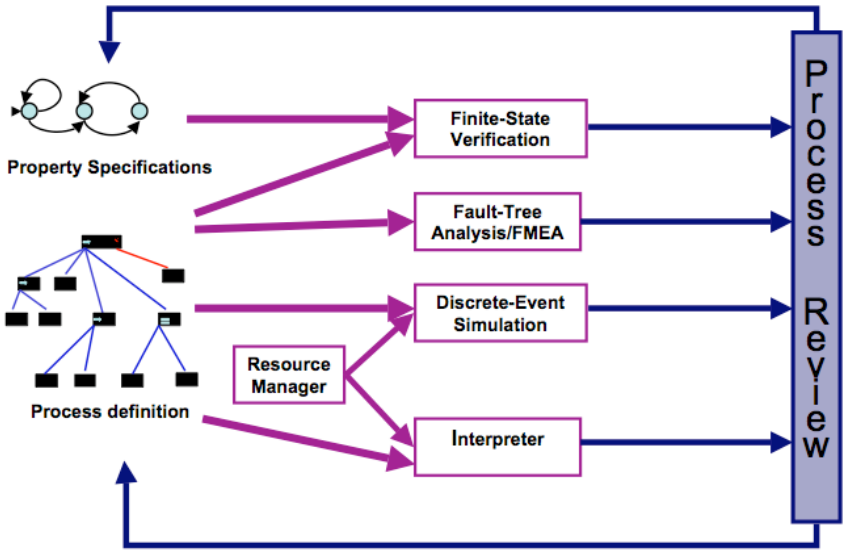

Figure 1: Continuous Process Improvement Environment

Creating a detailed, accurate process model involves considerable effort. It usually takes us several dozen meetings with domain experts before a reasonable process definition emerges. For each iteration, the appropriate domain experts review the process definition for accuracy and the computer scientists prepare questions to flesh out incomplete, inconsistent, or erroneous aspects of that model. It is unrealistic to expect domain experts to make such a large investment in time unless there will be considerable return on that investment. Thus, we use that single representation to drive a number of different analyzes, hopefully providing valuable feedback and justifying the modeling costs.

Multiple views of the process definition can be provided, including the natural language hypertext view and linear tree view mentioned above. Moreover, as shown in Figure 1, the Little-JIL process definition can be used to drive a number of different analyzes. We

have found that these analyzes find many subtle errors and provide valuable information for CPI activities.

4. Evaluation. As noted in section 3, 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. Our evaluations in these various domains are continuing and they will certainly lead to continued language and toolset enhancements.

5. Status. 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 tools for supporting Fault Tree analysis, the execution system, and discrete event simulation are not yet available for distribution.

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. Deming, W.E., Out Of The Crisis. M.I.T. PRESS (2000).
2. Wise, A., Little-JIL 1.5 Language Report. Department of Computer Science, University of Massachusetts, Amherst (2006) UM-CS-2006-51.
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 (2008) Invited Keynote, 889-898.
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 (2007) 49-57.

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 Information in Medicine. Special Topic on Model-Based Design of Trustworthy Health Information Systems 47 (2008) 392-398.
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. University of Massachusetts, Amherst (2009) UM-CS-2009-004.
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 (2004) 359-430.
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. ACM Press, Portland, OR (2006) 208-218.
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. Springer-Verlag LNCS, Shanghai, China (2006) 150-158.
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 (2008) 319-329.