Using Event Streams to Validate Process Definitions

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

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

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

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

Jenna L. Marquard
Dept. of Mechanical
and Industrial Engineering
University of Massachusetts
Amherst, MA 01003
jlmarquard@ecs.umass.edu

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

Abstract

This paper describes preliminary work on validating process definitions by comparing a process definition to event streams derived from an actual execution of that process. The goal of this work is to find and remove discrepancies in the process definition before using that definition as the basis for various forms of analysis and decision making. The paper outlines important issues that need to be addressed and suggests possible approaches. The example used in this paper is based on a process from the medical domain.

1 Introduction

Medical errors are reported to be one of the leading causes of death in the United States. According to a report by the US Institute of Medicine (IOM), "To Err is Human" [19], approximately 98,000 people die each year in the United Stated alone because of avoidable medical errors. This report also asserts that "the problem is not bad people in health care – it is that good people are working in systems that need to be made safer." Another report by the IOM [21] states that "what is perhaps most disturbing is the absence of real progress in information technology to improve clinical processes" [italics ours]. Thus, both [19] and [21] suggest that medical processes need to be improved in order to reduce medical errors.

Important medical processes, however, are complex and non-trivial to reason about. Medical processes, such as chemotherapy or emergency department processes, involve concurrent executions and synchronization, allocation of highly contested resources, and multitasking performed by busy healthcare providers. Healthcare providers often face exceptional circumstances and find that for these circumstances the processes are often not well defined. Their executions of the process vary, increasing the opportunity for errors, and making it harder to capture the process in a form that is amenable to analysis that can in turn help identify sources of medical errors.

In this paper, we use the term *process definition* to refer to a formal model of a process created in a process modeling language with well-defined semantics. Such formal models, as recent work (e.g. [2, 6, 7, 23, 24]) has suggested, can serve as the basis for various kinds of formal analysis, such as Finite State Verification [12, 16] that helps detect traces though the model where important process requirements could be violated, Fault Tree Analysis [5, 25] that determines where a process might be vulnerable to errors if a task is performed incorrectly or input data is inaccurate, and simulations [8] that attempt to evaluate the impact of different resource allocations on quantities of interest such as patient waiting time. These formal analysis techniques can in turn help support continuous process improvement [11].

Since process definitions are the basis for various types of analyses, analysts and decision makers must be confident that definitions of large and complex medical processes are an accurate representation of the real processes before basing important decisions on the results of those analyses. A process definition can be inaccurate with respect to the real

process it models in many different ways, such as the omission or inclusion of steps or the under or over specification of the ordering among those steps. Among the factors that make creation of accurate process definitions hard are the omission of important details during elicitation, the need to combine different and possibly conflicting perspectives into a single process definition, access to only a small subset of process participants, and the inadvertent introduction of mistakes by the process definition creators.

The focus of this work is to study the problem of *validating* process definitions. *Validation*, as defined by the US Department of Defense [1], is "the process of determining the degree to which a model is an accurate representation of the real-world from the perspective of the intended uses of the model". This work concentrates on comparing event streams, the sequences of naturally occurring events that arise during the execution of an actual process, to a definition of that process in order to validate or find inaccuracies in that definition.

Although, we are still in a preliminary stage of applying and evaluating our approach to process definition validation, we have encountered a number of interesting issues that need to be addressed. This paper focuses on two major areas of concern: matching the observed events with the set of steps or tasks in the process definition and comparing the observed event streams with the possibly infinite set of sequences of steps allowed by a process definition.

In the next section of this paper, we discuss related work. Section 3 presents an example of a medical process, discusses the Little-JIL language used to create a definition of that process, and provides a short overview of a human simulation of that process used to obtain event streams. Section 4 outlines the two major issues that need to be addressed when comparing a process definition to event streams. The conclusion summarizes this work and discusses possible future directions.

2 Related Work

Comparing process definitions to event streams Cook and Wolf [9] discuss the problem of comparing event streams from an executing process to event streams induced by a process model. They cast the problem as a string comparison problem and propose two metrics for measuring the difference between two strings. One of the metrics uses the number of insertions, deletions, and substitutions needed to transform one string into the other. The second metric is an enhancement of the first one and takes into account blocks of consecutive rather than single insertions/deletions. To find the string from a process model that is the "closest" match to a given string corresponding to an event stream, Cook and Wolf use a heuristic search in the space determined by the model states (they assume the model is in the form of a finite-state machine), the position in the event stream, and the operation that created the search state (match, insertion, deletion). We expect to build on this work as we develop metrics for comparing event sequences to process definitions.

There has been prior work on monitoring process execution and detecting deviations from an existing process definition. Cugola et. al. [10] have investigated a temporal logic-based approach that allows monitoring of processes during execution and detecting deviations from a previously created process definition. Their approach allows for deviations from the process definition to be tolerated during execution of the process as long as some global constraints are not violated.

Obtaining event streams from an actual process There are many ways for obtaining event streams corresponding to a process execution. For instance, process participants can be observed while they perform their work in general or a specific set of tasks. For this approach, events could be recorded by a human observer (this method is known as *shadowing* and has been used by the Industrial Engineering community [3]) or with some automated support such as a camera capable of event recognition or by capturing "observable" events associated with computer or hardware devices. Because processes may be very complicated and because process performers may be engaged in more than one process at a time, human simulation is sometimes used because it allows process participants to perform only one process in a simulated setting that can be carefully observed and monitored.

Event streams from a process can also be collected by interviewing process participants [15]. For example, different types of healthcare providers participating in a certain medical process – doctors, nurses, pharmacists – can be asked to describe step by step how they perform their tasks and the descriptions can be turned into event streams or scenarios.

3 An Example

3.1 The VPID process

The Verify Patient Identity (VPID) process is a subprocess of virtually any medical process involving the performance of patient-related tasks. The goal of the VPID process is to ensure that a task gets performed on the right patient. The VPID process is relatively straightforward but, at the same time, it is critical to safely performing medical processes. Unfortunately, the VPID process is sometimes neglected or performed haphazardly by healthcare providers [14]. Examples of dangerous medical errors resulting from incorrect performance of the VPID process are transfusing the wrong unit of blood, administering medication to the

wrong patient, or performing a procedure on the wrong patient.

There are several variations of the VPID process – sometimes the process does not directly involve the patient (e.g. computerized order entry), sometimes the patient already has an ID band that can be used as part of the patient's identity, and sometimes, for newly-admitted patients for instance, all identification information must be obtained directly from the patient. The example presented in this paper is the process that healthcare providers follow to ensure that the correct ID band is placed on a newly-admitted patient. We used an existing process definition of this variation of the VPID process and compared it to event streams from a human simulation of that process. In the next subsection, we briefly describe the Little-JIL process definition language, which was used to create a definition of the VPID process.

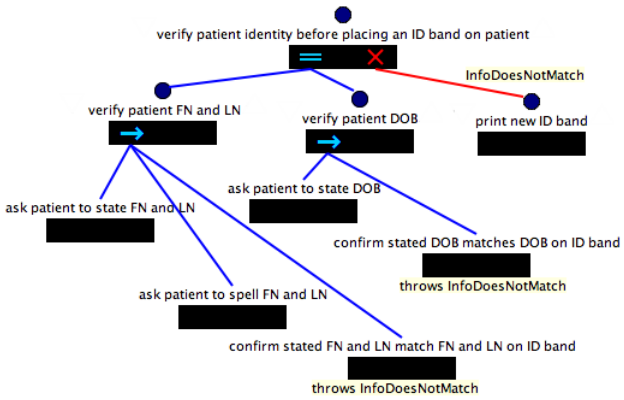

Figure 1. Part of original VPID process definition

3.2 Little-JIL and the VPID process definition

Little-JIL [4] is a process definition language with formal semantics and graphical syntax. Our experience with Little-JIL has indicated that its rich semantic features, such as support for specification of concurrency/synchronization, exceptional flow, and resources, make it suitable for capturing complex medical processes. Figure 1 shows part of the Little-JIL definition of the VPID process. The main building block of a Little-JIL process definition is the *step*, represented iconically by a black bar. A Little-JIL step corresponds to a unit of work, a task that a human or an automated agent executes in a process. Little-JIL supports hierarchical decomposition that allows steps at a higher level of abstraction to be decomposed into steps at a lower level of abstraction. For example, in Figure 1, the step *verify patient identity before placing an ID band on patient* is decomposed into the substeps *verify patient FN and LN* and *verify patient DOB* (where FN is an abbreviation for "first name", LN for "last name", and DOB for "date of birth"). Thus, a Little-JIL process definition can be viewed as a tree of hierarchically decomposed steps.

Little-JIL steps can have *sequencing badges* (located in the left part of the step bar), which specify the order of execution of substeps. In Figure 1, the step *verify patient identity before placing an ID band on patient* is a *parallel* step (denoted by an equal sign on the left of the step bar), which means that its substeps can be executed in any order, including in parallel. Thus, before placing the ID band on a patient, a healthcare provider needs to verify the patient's first/last names and date of birth and this could be done in any order. The step *verify patient FN and LN* is a sequential step (denoted by the right arrow in the step bar), which means that its substeps need to be executed in left to right order. Thus, to *verify patient FN and LN*, one should first *ask patient to state FN and LN*, then *ask patient to spell FN and LN*, and finally *confirm stated FN and LN match FN and LN on ID band*. Another kind of a sequencing badge is used in Figure 3 (Figure 3 itself is explained in more detail later). The step *ask patient for FN and LN* in Figure 3 is a *choice* step (denoted by a circle over a line on the left of the step bar), which means that one of its substeps can be chosen for execution and when that substep is completed, the parent step is considered to be completed as well.

In addition to supporting specification of nominal flow, Little-JIL supports the specification of exceptional flow. A Little-JIL step can throw an *exception* when some part of the execution fails. The exception is then handled by the youngest matching *exception handler* attached to an ancestor of the step that threw the exception. In Figure 1, the step *confirm stated FN and LN match FN and LN on ID band* can throw the exception *InfoDoesNotMatch*. In that case, the exception handler *print new ID band* (in Little-JIL an exception handler is attached to an "X" badge on a parent step and can itself be a step), which matches the thrown exception, is executed.

The rest of Little-JIL's features are not relevant to this paper but a more complete treatment of the language can be found elsewhere [26].

3.3 Obtaining event streams from VPID process execution

In this work, we used event streams derived from a human simulation [13] of the VPID process discussed above. In particular, the event streams correspond to the actions that clerks performed while placing an ID band on a newly admitted patient. The ID band was created before the start of the simulation, and before placing it on a patient, the

1	asked patient “Are you John Smith?”
2	looked at name on ID band
3	answered patient's question about phone use
4	looked at name on ID band
5	asked patient to state DOB
6	looked at name on ID band
7	looked at DOB on ID band
8	said DOB doesn't match

Figure 2. An example event stream for the VPID process

clerks had to verify that the information on the ID band exactly matches the information obtained from the patient. In this simulation, the actions of the clerks were recorded by a human observer. To obtain another perspective, the clerks also wore an eye-tracking device that designated exactly where they were looking at each moment in time. A careful post-processing of the human observation and eye-tracking data was done to identify events and to combine the events from the two perspectives to create event streams [20]. Figure 2 illustrates an example event stream that can be derived from human simulation data corresponding to a clerk performing the VPID process prior to placing an ID band on a patient.

4 Process Definition Validation

As noted above, the main focus of this work is to investigate approaches to process definition validation. We separate process validation into two main problems: mapping events in the event streams to steps from the process definition to create *mapped event streams* and comparing the mapped event streams to the process definition.

4.1 Mapping events from an event stream to steps from a process definition

Once an event stream of a real process execution is obtained, the events need to be mapped to steps in the process definition to allow for the comparison of the event stream to possible executions of the process definition. The most straightforward type of mapping is what we call *direct mapping*. In this case, an event from an event stream maps precisely to a task (a step or throwing of an exception) specified in an existing process definition. For example, the event *asked patient to state DOB* from the event stream in Figure 2 (event 5) constitutes exactly the execution of the step *ask patient to state DOB* from the process definition in Figure 1. Similarly, the event *said DOB doesn't match* (event 8) corresponds to the throwing of the exception *InfoDoesNotMatch* by step *confirm stated DOB matches DOB on ID band*.

It is often the case, however, that an event in the event stream does not map precisely to a step in the process definition or an event in the process definition does not correspond to an event in the event stream. The remainder of this section discusses some of the reasons these mismatches might occur and how they might be treated.

Variant of a step An event observed during a process execution may not correspond exactly to a step in the process definition, but it may correspond to a variant of that step, i.e. a different way to perform a step in the process definition. We identified two types of mappings between an event and a step in this situation – *inferior variant* and *superior variant*. An event from an event stream is an inferior variant of a step in a process definition if the event constitutes an execution of the step but it is not the best or preferred way of performing the step. For example, the event *asked patient “Are you John Smith?”* in Figure 2 (event 1) is an inferior variant of the step *ask patient to state FN and LN* in Figure 1. If a patient is confused or does not understand the language of the healthcare provider, then the patient can nod affirmatively or answer “yes” to a question of the form “Are you X” even if “X” is not the patient’s name. Thus, in medical practice, the preferred way to obtain a name is to ask the patient to state, and sometimes even spell, their name. Similarly, it is possible that an event observed during a human simulation is a better way of performing a task than the step specified in a process definition, and in this case we say that the event is a superior variant of the step. For example, if the process definition in Figure 1 contained only the step *ask patient to state FN and LN* but not the step *ask patient to spell FN and LN*, then some domain experts could consider the event *asked patient to state and spell FN and LN* to be a superior variant of the step *ask patient to state FN and LN*.

Difference in level of granularity An event in an event stream may not correspond exactly to a step in the process definition because the event is at a different level of granularity than the steps in the process definition. For example, the events *looked at name on ID band* and *looked at DOB on ID band* from the event stream in Figure 2 are lower level events that the original Little-JIL definition of the VPID process in Figure 1 did not include. This process is defined at a higher level of abstraction, without specifying the detail of how the person verifying the patient identity must look at the ID band.

To compare an event stream to a process definition, however, one may need to map lower-level (higher-level) events to process steps. We have considered two approaches that support the mapping of low-level events from an event stream to process steps.

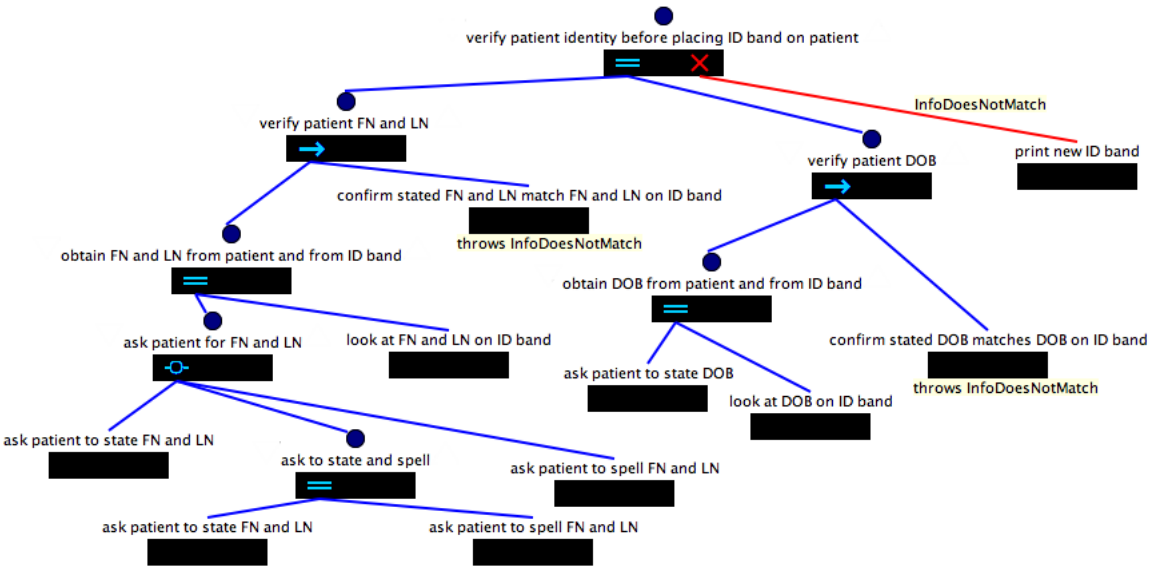

Figure 3. Part of VPID process definition after addressing some of the mapping issues

The first approach involves elaborating the process definition to a lower level of detail so that the steps that correspond to the lower-level events in the event stream are included in the process definition. Figure 3, which is a modified version of the VPID process from Figure 1, is an example of how a process definition can be elaborated to address the issue of difference in the level of granularity. The step *verify patient FN and LN* is elaborated by substituting the substeps *ask patient to state FN and LN* and *ask patient to spell FN and LN* in Figure 1 with the entire subtree rooted at step *obtain FN and LN from patient and from ID band* in Figure 3. Also the step *verify patient DOB* from Figure 1 is elaborated by substituting its substep *ask patient to state DOB* with the subtree rooted at the step *obtain DOB from patient and from ID band* in Figure 3. The modified process definition shown in Figure 3 captures two changes to the initial process definition in Figure 1. First, to accommodate the difference in level of granularity, the steps *look at FN and LN on ID band* and *look at DOB on ID band* have been added to the process definition so that the events *looked at name on ID band* and *looked at DOB on ID band* from the event stream in Figure 2 can be mapped to them. This also requires the addition of the parallel steps *obtain FN and LN from patient and from ID band* and *obtain DOB from patient and from ID band*) to capture the fact that asking the patient for his/her name (or date of birth respectively) and looking at the ID band for the name (or for the date of birth respectively) can be done in any order.

The second change captured by the process definition in Figure 3 is an interesting example of how the act of mapping event streams to process steps can help detect inaccuracies in the process definition. The change captured by the process definition in Figure 3 reflects the realization that the initial process definition in Figure 1 was too restrictive. While trying to accommodate the difference in the level of granularity of the event stream and the VPID process definition, one of the domain experts pointed out that asking the patient to state their name and then asking him/her to spell it (as specified by the steps *ask patient to state FN and LN* and *ask patient to spell FN and LN* in Figure 1) is not the only way to obtain the name of a patient. In fact, a healthcare provider can choose to obtain the patient's name in several ways, such as asking the patient to just state first name and last name, asking the patient to spell their first and last name, or asking the patient to both state and spell first and last name, in any order. This is captured by the subtree rooted at the choice step *ask patient for FN and LN* in Figure 3 and represents an improvement in the accuracy of the original process definition.

Another possible way to deal with events at a lower level of granularity than the process steps is to keep the process definition unchanged but to create a set of rules that define what lower level events from an event stream and executed in what order constitute the execution of a step from the process definition. For example, the initial process definition from Figure 1 can be kept and a rule can be created saying that "the occurrence of events *asked patient to state DOB* and *looked at DOB on ID band* in any order constitute the successful execution of the step *ask patient to state DOB*. Of course, elaborating the process definition to a lower level of detail or creating extra rules are basically equivalent approaches, where the first embeds the rules in the process

definition and the second keeps the rules and the process definition separate.

The above approaches deal with the case when an event in the event stream is at a lower level of granularity than the steps in the process definition, but it is also possible that an event is at a higher level of granularity than any steps in the process definition. In a hierarchical language it might be possible to find a location in the hierarchy where the appropriate step could be inserted. Alternatively, when this is not desirable or for languages that do not support hierarchical decomposition, external rules could be used that encode what low-level process steps are considered equivalent to a high-level event. An example of such a rule can be "the occurrence of event *verified patient DOB* constitutes the successful execution of the steps *ask patient to state DOB*, *look at DOB on ID band*, and *confirm stated DOB matches DOB on ID band*". A finite-state automaton or separate process definition could be used to encode the rule if the order in which the process steps are executed is important to the mapping.

In general, even if the language used for the process definition supports hierarchical decomposition, it may be hard to map high-level or low-level events to process steps when the conceptual task decomposition in the process definition is not somewhat similar to the events in the event stream. Although external rules, similar to the ones discussed above, could be applied, if the mapping is extensive, the resulting validation will probably be suspect.

Extra or irrelevant events Sometimes an event in the event stream may not match a step in the process definition, and this difference does not appear to be because of a mismatch in the level of granularity. These events may be either *extra events* or *irrelevant events*

Extra events appear to be relevant to the process and could impact reasoning about the correctness of the process. Such events may have been inadvertently omitted from the process definition, perhaps by mistake by the creator of the definition or perhaps because of an error in how the process was actually being executed.

Irrelevant events are those that seem to be extraneous to the process definition for the types of analyses desired. An example could be the event *answered patient's question about phone use* in the event stream in Figure 2. It could be decided that answering patient's questions concerning phone use does not affect the process of VPID and thus this event is irrelevant to the process of VPID captured by the process definition. This does not necessarily mean however that the event can simply be deleted from the event stream. Depending on the kind of steps being performed in the process, one may decide to keep such irrelevant events. For example, in the VPID process, a clerk is obtaining DOB from patient, remembering this information for a short period of time and then comparing it to the DOB on the ID band. One may insist that the clerk should not be distracted by performing other tasks between obtaining the DOB from the patient and comparing it to the one on the ID band. Thus, it may be desirable to keep irrelevant events in the event stream and take them into consideration during the analysis.

Unobservable steps In addition to mapping the events from the event streams to steps from the process definition, one needs to consider the steps in the process that are not represented in the event stream. The process definition might contain *unobservable steps*, which are steps that represent tasks that cannot be observed in an execution of a process. Thus, these steps will not have corresponding events in the event streams. One kind of unobservable steps is the *cognitive step*, which is an action that an individual performs mentally and thus cannot be observed while the individual is performing a process. The step *confirm stated FN and LN match FN and LN on ID band* from the process definitions in Figure 1 and Figure 3 is an example of a cognitive, unobservable step. Unobservable steps can be some of the most important steps in processes and thus need to be dealt with when validating process definitions. The step *confirm stated FN and LN match FN and LN on ID band*, for instance, is an essential step as the activity of comparing the first and last names obtained from the patient to the ones obtained from the ID band is the core of the process of verifying patient identity and hence needs to be specified in the process definition. The fact that such unobservable steps are not part of the event stream does not imply that the participant did or did not execute them. Thus, before starting the comparison between an event stream and a process definition, it is important to identify the unobservable steps in the process definition and handle them appropriately.

We have considered three ways of dealing with unobservable steps, particularly cognitive steps, in a process definition to accommodate its comparison to an event stream. The first approach involves the use of a "think-aloud" protocol (a knowledge elicitation technique [15]) used while observing process performers in a real or simulated setting. This means that process performers are asked to verbalize certain mental activities they perform. This way, the cognitive steps are exposed to an observer and can be recorded as events in the event stream.

The second approach for dealing with unobservable steps involves embedding errors in a simulated process (a technique used in studying pilot decision making [22]). For example, in a human simulation of the VPID process, a wrong patient name can be printed on an ID band. If a healthcare provider participating in the human simulation places the ID band on the simulated patient, then it can be inferred that the healthcare provider either did not perform

the step *confirm stated FN and LN match FN and LN on ID band* or performed it incorrectly.

The third approach for dealing with unobservable steps involves the development of rules outside the process definition that specify under what circumstances an unobservable step is considered executed by a process performer. These rules are similar to the ones proposed for dealing with difference in the level of granularity between events and steps. In particular, they would specify what set of low-level events, perhaps executed in certain order, constitutes successful completion of a certain step from the process definition. An example of such a rule for the VPID process is "The occurrence of the events *asked patient to state DOB* and *looked DOB on ID band* (in any order), followed by the occurrence of the event *place ID band on patient* means that the step *confirm stated DOB matches DOB on ID band* was performed."

Distinguishing between cases and dealing with a mismatch An important issue that arises is to determine which of the above cases applies when mapping a particular event to a process step. For example, it is not always easy to decide whether an event corresponds exactly to a step or if it is a variant of a step; and when it is a variant, whether it is a superior or an inferior variant. Similarly, it may be hard to decide whether an event corresponds to a step omitted from the process definition or whether the event is at a different level of granularity.

If it has been determined that an event does not map exactly to a step and that the process definition needs to be elaborated or external rules need to be created, it needs to be decided how to change the process definition (or how to create a rule that captures the intended mapping). Both distinguishing between cases that apply when dealing with mappings and changing the process definition (creating rules) to support the mapping usually require the knowledge and personal judgment of a domain expert. For example, we had to consult healthcare professionals when elaborating the Little-JIL definition of the VPID process to incorporate the steps *looked at name on ID band* and *looked at DOB on ID band*.

4.2 Comparing a process definition to event streams

Once the individual events in the event streams have been mapped to steps in the process definition to create *mapped event streams*, these mapped event streams can be compared to the process definition. This involves comparing individual event streams to a process definition and then considering the implications of the results for how the whole set of event streams compares to a process definition.

The approach we are currently considering views the process definition validation problem as a string comparison problem. In particular, each event stream can be represented as a string such that each event in the stream corresponds to a symbol from a finite alphabet. The alphabet itself is the set of all steps/tasks corresponding to events observed during a process execution. A process definition, and in fact any computer program and many formal models, describes a (possibly infinite) set of event streams. For a process definition, an event stream corresponds to a possible execution or trace of a process definition, where the events are steps/tasks in that process definition. Thus, a process definition denotes a language of strings (event streams) and the process definition validation problem can be stated in terms of comparing the strings corresponding to actual executions of a process to the strings in the language denoted by the process definition.

Determining whether a given mapped event stream is a legal execution of the process specified by a process definition can be viewed as a set membership problem, i.e. "Is the string corresponding to the mapped event stream a member of the language denoted by the process definition?".

To be able to address the set membership problem in an automated way, we have used finite-state verification and in particular the FLAVERS system [12]. We first encode a mapped event stream from the human simulation as a property and then run FLAVERS to check whether there is a path through the VPID process definition (an execution of the process definition) that satisfies the property. If such a path exists, then the mapped event stream is a valid execution of the process, otherwise it is not.

Determining whether a given event stream is a valid execution of a process definition is just a first step in the process definition validation problem. Since we formulate the FSV problem as a "none" property, the event stream represented by the property is a part of a valid execution of the process definition if the property is violated. The counter example trace is just a trace through the process that contains the event stream. Unfortunately, with this approach, there is no useful information provided about how to improve the process definition, when the event stream is not part of a valid execution of the process definition.

It may also be useful to characterize the difference between an event stream and a process definition. Cook and Wolf [9] have investigated an approach in which they find the execution of the process model that is "closest" to a given event stream and then find the locations of the insertions and deletions that need to be applied to the event stream to transform it into that "closest" process model execution. Other information that could be helpful when validating process definitions is identifying events in the event stream that were done in different order than the one specified in the process definition. Since the main goal of our

work is to improve the process definition, it may be useful to find automated ways that suggest how the process definition needs to be changed in order for a given event stream to become a valid execution.

In addition to comparing a single event stream to a process definition, it will be useful to compare a set of event streams to a process definition. For example, identifying a common pattern among a set of event streams that are not valid executions of a process definition could help identify areas to consider for process definition improvement or areas that require additional training of personnel. Perhaps the work on fault localization that considers sets of correct and incorrect executions could be applied to help localize the areas of the process that need such consideration [17, 18].

5 Conclusion

Process definitions can be used as the basis for several analysis techniques that can in turn be used to support continuous process improvement. In particular, process definitions and associated analysis methods seem to be suitable approaches for improving medical processes, which are known to contain faults that may lead to unnecessary pain and suffering by patients. The inherent complexity of medical processes, however, makes it hard to create a process definition that is an accurate representation of the real process and at the same time is rigorous and detailed enough to support meaningful analyses. Thus, it is of overriding importance that definitions of medical processes be validated before being used as the basis for analyses that will inform decision makers about vulnerabilities or lack of vulnerabilities in these processes.

This paper outlines some important issues that arise when trying to validate a definition of a medical process and suggests some approaches to deal with these issues. Our work is still in a preliminary stage but an interesting future direction for this research will be to extend existing techniques for process definition validation and detection of deviations (e.g. [9, 10]) to address the outlined issues.

In addition to supporting continuous process improvement, in the long term, process definitions can be used to support process guidance. After various analysis techniques are applied to improve the definition of a medical process, that process definition can be used to guide the tasks of various human and automated agents and detect undesirable deviations.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Awards CCF-0427071, CCF-0820198, CCF-0829901, 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.

The authors gratefully acknowledge the work of Elizabeth A. Henneman and Huong Phan, who made major contributions to the development of the Little-JIL definition of the VPID process, of Don Fischer, who made major contributions to the use of an eye-tracking device in the human simulations of the VPID process, and of Megan Campbell, Tuan Pham, and Qi Ming Lin, who helped with the extraction of event streams from human simulation data.

References

[1] Dod modeling and simulation management. *Department of Defense Directive 5000.59*, page 7, 2007.

[2] S. Bäumler, M. Balser, A. Dunets, W. Reif, and J. Schmitt. Verification of medical guidelines by model checking - a case study. In *Proceedings of the 13th SPIN Workshop on Model Checking Software*, pages 219–233, 2006.

[3] P. Carayon, A. S. Hundt, B. T. Karsh, A. P. Gurses, C. J. Alvarado, M. Smith, and P. F. Brennan. Work system design for patient safety: the seips model. *Quality and Safety in Health Care*, pages 50–58, 2006.

[4] A. G. Cass, B. S. Lerner, J. Stanley M. Sutton, E. K. McCall, A. Wise, and L. J. Osterweil. Little-jil/juliette: a process definition language and interpreter. In *ICSE '00: Proceedings of the 22nd International Conference on Software Engineering*, pages 754–757, New York, NY, USA, 2000. ACM.

[5] B. Chen, G. S. Avrunin, L. A. Clarke, and L. J. Osterweil. Automatic fault tree derivation from little-jil process definitions. In *SPW/ProSim*, volume 3966 of *LNCS*, pages 150–158, Shanghai, May 2006.

[6] B. Chen, G. S. Avrunin, E. A. Henneman, L. A. Clarke, L. J. Osterweil, and P. L. Henneman. Analyzing medical processes. In *ICSE '08: Proceedings of the 30th International Conference on Software Engineering*, pages 623–632, New York, NY, USA, 2008. ACM.

[7] S. Christov, B. Chen, G. S. Avrunin, L. A. Clarke, L. J. Osterweil, D. Brown, L. Cassells, and W. Mertens. Rigorously defining and analyzing medical processes: An experience report. *Models in Software Engineering: Workshops and Symposia at MoDELS 2007, Nashville, TN, USA, September 30 - October 5, 2007, Reports and Revised Selected Papers*, pages 118–131, 2008.

[8] L. G. Connelly and A. Bair. Discrete event simulation of emergency department activity: A platform for system-level operations research. *Academic Emergency Medicine*, 11(11):1177–1185, 2004.

[9] J. E. Cook and A. L. Wolf. Software process validation: quantitatively measuring the correspondence of a process to a model. *ACM Transactions on Software Engineering and Methodology*, 8:147–176, 1999.

[10] G. Cugola, E. D. Nitto, C. Ghezzi, and M. Mantione. How to deal with deviations during process model enactment. In *ICSE '95: Proceedings of the 17th International Conference on Software Engineering*, pages 265–273, New York, NY, USA, 1995. ACM.

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

[12] M. B. Dwyer, L. A. Clarke, J. M. Cobleigh, and G. Naumovich. Flow analysis for verifying properties of concurrent software systems. *ACM Transactions on Software Engineering and Methodology*, 13(4):359–430, 2004.

[13] P. L. Henneman, D. L. Fisher, E. A. Henneman, T. A. Pham, M. M. Campbell, and B. H. Nathanson. Many health care workers do not verify patient identity prior to performing common tasks. *Annals Emergency Medicine (submitted)*, 2008.

[14] P. L. Henneman, D. L. Fisher, E. A. Henneman, T. A. Pham, Y. Y. Mei, R. Talati, B. H. Nathanson, and J. Roche. Providers do not verify patient identity during computer order entry. *Academic Emergency Medicine*, 15(7):641–648, 2008.

[15] Hoffman, Shadbolt, B. A. Mike, and K. Gary. Eliciting knowledge from experts: A methodological analysis. *Organizational Behavior and Human Decision Processes*, 62(2):129–158, May 1995.

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

[17] D. Jeffrey, N. Gupta, and R. Gupta. Fault localization using value replacement. In *ISSTA '08: Proceedings of the 2008 International Symposium on Software Testing and Analysis*, pages 167–178, New York, NY, USA, 2008. ACM.

[18] J. A. Jones, M. J. Harrold, and J. Stasko. Visualization of test information to assist fault localization. In *ICSE '02: Proceedings of the 24th International Conference on Software Engineering*, pages 467–477, New York, NY, USA, 2002. ACM.

[19] L. T. Kohn, J. M. Corrigan, and M. S. Donaldson, editors. *To Err is Human: Building a Safer Health System*. National Academies Press, Washington, DC, 1999.

[20] J. Marquard, S. Christov, P. Henneman, L. Clarke, L. Osterweil, G. Avrunin, D. Fisher, E. Henneman, M. Campbell, and T. Pham. Studying rigorously defined health care processes using a formal process modeling language, clinical simulation, observation, and eye tracking. In *Proceedings of the International Conference on Naturalistic Decision Making (accepted for publication)*, 2009.

[21] I. of Medicine. *Crossing the Quality Chasm: A New Health System for the 21st Century*. National Academies Press, Washington D.C., 2001.

[22] H. M. Soekkha. *Aviation Safety: Human Factors, System Engineering, Flight Operations, Economics, Strategies and Management*. Brill Academic Publishers, 1997. This book has no author, just editor. Some places online, "International Aviation Safety Conference" is mentioned as author.

[23] A. ten Teije, M. Marcos, M. Balser, J. van Croonenborg, C. Duelli, F. van Harmelen, P. Lucas, S. Miksch, W. Reif, K. Rosenbrand, and A. Seyfang. Improving medical protocols by formal methods. *Artificial Intelligence in Medicine*, 36(3):193–209, 2006.

[24] P. Terenziani, L. Giordano, A. Bottrighi, S. Montani, and L. Donzella. Spin model checking for the verification of clinical guidelines. In *ECAI 2006 Workshop on AI Techniques in Healthcare: Evidence-based Guidelines and Protocols*, August 2006.

[25] W. Vesely, F. Goldberg, N. Roberts, and D. Haasl. *Fault Tree Handbook (NUREG-0492)*. U.S. Nuclear Regulatory Commission, Washington, D.C., January 1981.

[26] A. Wise. Little-jil 1.5 language report. Technical Report (UM-CS-2006-51), Department of Computer Science, U. of Massachusetts, Amherst, 2006.