Online Deviation Detection for Medical Processes

Stefan C. Christov, George S. Avrunin, Lori A. Clarke
School of Computer Science, University of Massachusetts Amherst, MA

Abstract

Human errors are a major concern in many medical processes. To help address this problem, we are investigating an approach for automatically detecting when performers of a medical process deviate from the acceptable ways of performing that process as specified by a detailed process model. Such deviations could represent errors and, thus, detecting and reporting deviations as they occur could help catch errors before harm is done. In this paper, we identify important issues related to the feasibility of the proposed approach and empirically evaluate the approach for two medical procedures, chemotherapy and blood transfusion. For the evaluation, we use the process models to generate sample process executions that we then seed with synthetic errors. The process models describe the coordination of activities of different process performers in normal, as well as in exceptional situations. The evaluation results suggest that the proposed approach could be applied in clinical settings to help catch errors before harm is done.

1 Introduction

Human errors are a major concern in many medical processes. In 1998, an Institute of Medicine (IOM) report estimated that preventable medical errors cause the death of 98,000 people each year in the U.S.[1] More than a decade later, a 2009 U.S. National Research Council report[2] indicated that the problem with errors still persists and that "it is widely recognized that today's health care … suffers substantially as a result of medical errors." A recent study[3] from 2013 reported an even higher estimate of deaths per year, between 210,000 and 400,000, in the U.S. due to medical errors.

To help address this problem, we are investigating an approach for automatically detecting when performers of a medical process deviate from the acceptable ways of performing that process. Such deviations could represent *planning errors*[*] and, thus, detecting and reporting deviations as they occur (i.e., online) could help catch such errors before something bad happens. Here we use the word *process* to refer to the coordination of activities to accomplish a task or a goal, where the activities may be performed by humans, devices, or software systems. We refer to the *execution* of the process as the sequence of steps that are actually performed to accomplish the task or the goal. Our process models capture the process executions typically described in clinical guidelines or protocols, but also include additional detail, such as how guidelines should be executed when exceptional situations arise and customization that might occur for a particular clinical setting.

Detecting deviations could be challenging, however, because of the complexity of medical processes. Medical processes often contain multiple decision points that might require expert judgment, multiple possible exceptional situations that might require special handling, and multiple subprocesses that might be performed concurrently by different medical professionals. This typically results in a very large set of acceptable executions of a medical process. The proposed deviation detection approach uses techniques from software engineering for constructing a compact model describing all acceptable executions of a process. The large number of possible executions, however, combined with the fact that the process execution that human process performers intend to follow is often unknown (and could change as the process is being performed), makes online deviation detection difficult.

The proposed deviation detection approach relies on monitoring an executing process to determine whether the way the process is being performed deviates from the way it should be performed as specified by the process model. In particular, the approach aims to detect the manifestation of errors, i.e., *error phenotypes*[18], in a sequence of performed activities. Recent developments in healthcare and informatics should enable monitoring and recognizing many of the activities performed in real time, that is, as the process is being executed. For example, electronic medical records are being introduced in more hospitals and data entries in such records could be used to infer what activities are being/have been performed. Medical scribes are increasingly being used in medical processes to document the executing process[4] and thus could capture the activities as they are being performed. Computer vision techniques could also be utilized to recognize performed activities in a live video stream from cameras installed in a hospital room.

[*]A *planning error*, as defined by the IOM report[1], is "the use of a wrong plan to achieve an aim". Examples of planning errors are omitting an activity that should have been done (error of omission) or performing an activity that should not have been done (error of commission).

Even if we assume a perfect mechanism for monitoring an executing process, however, there are important issues related to the feasibility of the proposed deviation detection approach that need to be explored before the approach could be deployed in a clinical setting. One of these issues is potential harm due to *delayed deviation detection*. Delayed deviation detection arises when process performers deviate from the acceptable executions of a process, but this deviation cannot be detected until additional process activities are performed. Since the intent of process performers is often unknown to the monitoring system and some decisions at branch points in a process could depend on subjective (and potentially error-prone) human judgment based on experience and domain expertise, a monitoring system might not know which branch(es) should be taken in a given execution of the process. For example, suppose that at a branch point a human intends to perform one branch, but, due to distraction or fatigue, makes a commission error by first performing steps from another branch before performing the steps from the intended branch. In that case, a monitoring system will not be able to detect the commission error until the human performs the first step of the intended branch, or possibly until even later, if the two branches have common steps.

Ideally, we would like to detect deviations before harm is done as a result of these deviations, but the presence of detection delay could prevent achieving this goal in some situations. In this paper, we investigate how frequently delayed deviation detection could arise in medical processes, how often harm could occur because of the delay, and what the causes for detection delay could be. Harm could also occur as a result of a deviation even when there is no deviation detection delay, and we investigate this issue as well. We also investigate the performance of the deviation detection approach in terms of computation time. If the computation time to detect a deviation is too long, harm could be done before process performers are notified of the deviation.

Our initial evaluation of the deviation detection approach has been promising. Delayed deviation detection rarely arose in the medical processes we studied and when it did arise, no harm to the patient could occur as a result of the delay. The running time of the deviation detector is low relative to the human speed of performing process activities, suggesting that it could be used for online deviation detection.

Section 2 discusses related work, the proposed deviation detection approach, and some issues associated with that approach. The experimental methodology for evaluating the approach with respect to these issues is presented in section 3. The results are presented in section 4 and discussed in section 5. Section 6 summarizes the paper and describes future work.

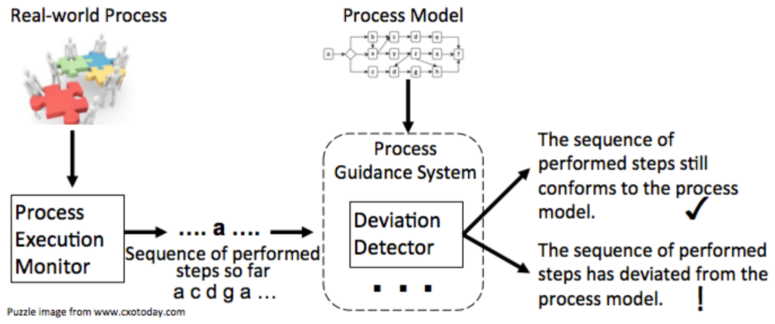
2 Background

2.1 Related Work

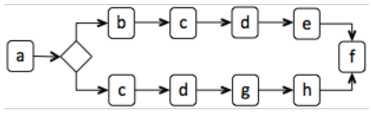
There are several approaches that aim to reduce the number of medical errors by encouraging conformance with some specification of the acceptable ways to perform a process (e.g., process aids such as checklists[5,6] and care sets[7]). Such process aids, however, tend to specify only the major steps during normal flow, omitting important details such as exceptional scenarios and concurrent process execution[8,9]. Such process aids often add to the workload of already heavily-burdened medical professionals. The use of checklists, for example, often requires medical professionals to check what needs to be done, to remember what steps they completed, and to determine the appropriate checklist to use in a given context.

To remove some of the burdens that process aids, such as checklists, place on medical professionals, there have been attempts to create systems that automatically check the compliance of an executing process with a specification of that process[11]. To support such systems, various medical guideline modeling languages have been developed[10]. Subsequent work has utilized formally specified medical guidelines to drive careflow management systems. For example, Fitzgerald et al. designed and deployed a careflow management system in a trauma center to guide medical professionals during the first 30 minutes of trauma resuscitation[11]. This system increased compliance with the modeled medical protocol and reduced error rates, but the model did not support complex process behaviors, such as concurrency and exception handling. The framework we are proposing, including the work reported in this paper, is intended to go beyond these limitations.

In prior work, we constructed detailed models of several medical processes[8,9,12] in consultation with medical professionals. We also constructed a framework and a set of tools for analyzing such models to detect various safety problems and evaluate proposed changes[12]. Although building and analyzing such detailed models takes considerable care and time, this work resulted in an improved understanding of the processes and the detection of several errors and vulnerabilities, leading to nearly 70% reduction in the number of errors reaching the patient in one of our studies[13].


Real-world Process
Process Model
Process Execution Monitor
.... a ....
Sequence of performed steps so far
a c d g a ...
Process Guidance System
Deviation Detector
The sequence of performed steps still conforms to the process model.
The sequence of performed steps has deviated from the process model.
Puzzle image from www.cxotoday.com

(a) Process Monitoring and Guidance Framework.



(b) An Example Control Flow Graph

Figure 1: The Process Monitoring and Guidance Framework and an example of a simple CFG.

We are now beginning to examine the potential for using such models, validated by the analyses, for the monitoring and guidance of executing processes.

2.2 Deviation Detection Approach

The proposed deviation detection approach is a component of a framework we are developing for process monitoring and guidance. This framework also supports other process guidance aspects, such as pro-actively informing process performers of pending activities depending on how the execution of the process develops and providing capabilities for inspecting process execution history.

In this paper, we focus on the deviation detection aspect of the framework. Figure 1(a) shows the relevant components of the framework. As a process is being performed, the *Process Execution Monitor* captures events associated with the performed steps and incrementally creates the *sequence of performed step events*. We use the term *step* to refer to an activity of interest performed as part of a process by humans, software or hardware devices. For this paper, we assume that the sequence of performed step events is accurate, i.e., the sequence contains events for all steps of interest that have been performed, these events are in the order in which the actual steps were performed, and events that are not of interest have been accurately removed[†]. The precise way that the Process Monitor captures events is beyond the scope of this paper.

Every time the *Deviation Detector* receives a step event from the Process Execution Monitor, it checks if the corresponding sequence of steps performed so far is one of the acceptable sequences as specified by the *Process Model*. If it is not, a deviation is detected and process performers are warned that a potential error might have occurred.

Process Model. The process model captures the acceptable ways to perform the corresponding process. This includes the nominal ways, but also how the process should be performed when various exceptional situations arise. The model can also capture different ways to perform the process depending on the level of expertise of process performers.

Eliciting information from domain experts to create such process models and translating the elicited information into an actual model are known to be difficult and time consuming. In our work, we have used several process elicitation methods, such as observations, structured, and unstructured interviews[13]. After creating process models based on the elicited information, we have used several formal analysis techniques to validate these models[12]. Regardless of how extensive the process elicitation and model validation efforts are, however, it is likely that process models will still contain some inaccuracies. Furthermore, process performers might execute a process in innovative and yet acceptable ways that are not captured in the process model. In practice, we expect process models to be continually updated as defects in the models are uncovered and/or as the modeled processes change.

In our experience, to support the complexity of medical processes, the process model needs to be written in a notation with rich and well-defined semantics. Specifically, such a notation should provide support for modeling human choice, exception handling, concurrency, and synchronization. For our work, we have chosen the Little-JIL[14] process modeling language, because it satisfies these requirements and because we have found that the Little-JIL's compact visual notation has facilitated communication with the medical professionals who have been helping us create

[†]To simplify the discussion, hereafter we shall refer to the sequence of step events associated with the sequence of actual performed steps as the sequence of performed steps.

and validate the models of the relatively large and non-trivial processes on which we have been working (two of these processes are briefly described in section 3.1). The proposed deviation detection approach, however, is independent of the process modeling language; it can be used with any language that has well-defined semantics that are also sufficiently rich to capture complex real-world processes. For the purposes of deviation detection, a Little-JIL model is translated into a typical control flow graph (CFG) representation of the possible step execution orderings associated with a process model, so in this paper only a simple control flow graph representation is shown.

Deviation Detector. The Deviation Detector traverses the CFG representation in a breadth-first way to determine whether the sequence of performed steps so far is a legal sequence of steps through the graph. For efficiency, this traversal is done incrementally, maintaining a frontier of possible nodes in the graph that could correspond to the last performed step. When this frontier becomes empty, i.e., when there is no path through the graph that corresponds to the sequence of performed steps, a deviation is detected.

2.3 Example of Applying the Deviation Detection Approach to a Medical Process

Consider the process of transfusing a unit of blood to a patient. This process usually starts after a nurse receives a physician's order to perform a transfusion. The nurse then needs to check if the patient's blood type and screen are known and, if they are not, obtains a blood sample from the patient and sends it to the lab for analysis. Once the type and screen are known, the nurse can request the appropriate blood product from the blood bank and pick it up when it becomes available. To administer a single unit of blood product, the nurse needs to provide some documentation and perform a series of checks to ensure that the right patient receives the right blood product at the right time. During the transfusion, the nurse needs to periodically check for complications (such as a transfusion reaction) and, if any arise, handle them appropriately. Once the transfusion is done, the nurse needs to evaluate the patient, create final documentation, and dispose of the infusion materials.

Most of the activities in the blood transfusion process consist of a fair number of subactivities that can occur in different orders and potentially different number of times. Problems could arise while performing many of these activities, making the blood transfusion process challenging to perform correctly and potentially error-prone[8].

A common error reported in the medical literature, and one that can cause severe harm to patients, is not fully following the procedure for verifying the patient's identity[8] while performing the series of checks before beginning the infusion. A possible instance of this error is omitting to verify that the patient is wearing the correct ID band. Consider the situation where in a busy emergency department patient X is wearing the incorrect ID band—that of patient Y. Perhaps a registration clerk had to place ID bands on several patients and inadvertently switched the ID bands. Suppose that patient Y is the one for whom a blood transfusion was ordered. Since patient X is wearing patient Y's ID band, if the nurse does not verify patient X's ID band prior to infusing the blood, patient X might receive the blood ordered for patient Y. Note that the nurse might still successfully perform other checks, such as the blood product checks, since they are done against the ID band and not against the patient's real identity.

Potential harm as a result of this error might be avoided if the nurse is warned that the process is being performed incorrectly before the infusion is started. The deviation detector could achieve this is by comparing the sequence of steps the nurse has performed against a model of the process. The sequence of steps that omits the step *verify that the patient is wearing the correct ID band* would not be a legal sequence through the process model and the deviation detector could establish that. Informing the nurse about such a deviation might help the nurse recover from the error before harm is done (i.e., infusing blood into the wrong patient).

2.4 Issues

Delayed Deviation Detection and Potential Harm Due to Delay. As discussed earlier, delayed deviation detection is an important issue to consider as it might constitute a significant threat to the usefulness of the proposed deviation detection approach, depending on how often delayed deviation detection arises and the severity of the consequences from the delay. For a concrete illustration of delayed deviation detection, consider the example control flow graph in Figure 1(b), assumed to have been created from a process model. In this CFG, nodes represent process steps and are labeled with letters. There is an edge from one node to another, if the step the first node represents can be immediately followed by the step the second node represents in the modeled process. The diamond node represents a branch point. When no branch conditions are modeled, such as when a medical professional needs to make a decision based on personal judgment and experience, the deviation detector cannot determine which branch should be taken in a particular process execution and must consider all options.

The two legal sequences of steps allowed by the CFG in Figure 1(b) are *abcdef* and *acdghf*. Suppose that during a particular execution of the process, the performers planned to carry out the sequence *abcdef*, but forgot to perform step b. In this situation, the deviation occurs when b is omitted, but it cannot be detected until e is performed, because *acd* is a valid sequence through the model. In this situation, we say that there is a detection delay of 2 (we measure the delay in terms of performed steps between the error—the omission of step b—and the step when the deviation is detected—step e). If the two branches in Figure 1(b) had a longer sequence of identical substeps (in this case this sequence is *cd*), the deviation detection delay could be even longer. In principle, the delay can be arbitrarily long and serious harm could potentially result from not detecting a deviation at the time when an error is made.

In this work, we explore the following research questions related to delayed deviation detection: How often do deviation detection delays occur in complex medical processes? How long are deviation detection delays? How harmful are deviation detection delays? What causes deviation detection delays and can they be reduced or avoided?

These questions could potentially be analytically explored by statically analyzing a set of process models. We have not been able to develop a reasonable static approach, however, that can handle multiple kinds of planning errors effectively (e.g., single-step omission, single-step commission, single-step substitution, omission of n consecutive steps, …, omission of n not necessarily consecutive steps, …, single subprocess omission, …, multiple subprocess omission, etc.). Thus, as a first step in exploring the above questions, we chose an experimental approach that applies the deviation detector to synthetically generated erroneous executions from realistic models of two medical processes.

Potential Harm When Deviations Are Detected without Delay. It is sometimes possible that deviations are detected immediately as an error is made (there is no detection delay), but harm could potentially still occur as a result of the deviations. For example, in the blood transfusion process discussed above, one of the checks that needs to be performed before infusing the blood is to ensure that the blood product has not expired. Suppose that the process model allows this check to occur immediately before infusing the blood. If the nurse forgets to check that the blood product has not expired, the deviation will be detected when the nurse starts the infusion. Even though there is no detection delay, harm could still occur as the patient might receive expired blood. This is an example of a vulnerability in the process model or in the process itself (assuming the model is accurate) with respect to the proposed deviation detection approach—a deviation cannot be detected until the potentially harmful step is already started.

In this work, we explore the following research questions related to the above issue: How often do situations arise where harm could occur even when deviations are detected without delay? What can be done to reduce or avoid such situations?

Running Time of the Deviation Detector. It is important that the deviation detector does not take too long to compute whether a deviation has occurred after a new step is added to the sequence of performed steps. Otherwise, if a deviation occurs, the warning about that deviation might get issued after harm was already done as a result of the deviation. We explore the following research question related to the performance of the deviation detector: What is the running time of the deviation detector when applied to realistic medical processes?

3 Methods

To perform an initial evaluation of the proposed deviation detection approach with respect to the above issues, we applied it to models of two medical processes—chemotherapy and blood transfusion. We generated sequences of performed steps from these models and then mutated the sequences to represent process executions with simple errors. Our evaluation approach is based on *mutation testing*[16], a common software engineering technique where a computer program is systematically mutated (usually by making a predefined set of simple changes to the source code) to evaluate software testing and analysis approaches.

3.1 Process models

The chemotherapy process model we used was elicited from medical professionals participating in an outpatient chemotherapy ordering and administration process in a regional cancer center in Western Massachusetts. At the time of writing this paper, the chemotherapy process model covers multiple phases of the chemotherapy process, including diagnosing the patient and ordering chemotherapy; thorough review of the treatment plan and medication orders by a medical assistant, a nurse, and a pharmacist; conducting an informational/teaching session with the patient and obtaining informed consent form; preparing chemotherapy drugs, performed by a pharmacist and pharmacy technicians; assessing the patient and administering the drugs, performed by a clinical nurse[13].

The blood transfusion process model is based on a standard blood transfusion checklist from the medical literature[5] and includes additional information about exceptional situations that might arise during the process and their handling. This model is part of a blood transfusion benchmark[15] that we developed with a nursing faculty member working on patient safety[8]. The model covers the process activities starting from receiving a physician order for transfusion to infusing the blood into the patient and performing subsequent follow-through checks. It specifies multiple phases of the process, including verifying that the patient blood type and screen are available (and if they are not, performing the subprocess of obtaining and labeling a blood specimen); ordering and obtaining a blood product from the blood bank; performing various verifications on the patient and on the blood product before starting the transfusion; monitoring the patient during the transfusion and appropriately reacting if a transfusion reaction is suspected.

The chemotherapy and blood transfusion process models are of significant size and complexity. The chemotherapy process model includes 283 steps performed by human process performers and specifies 59 exception handling situations[‡]. The corresponding low-level CFG representation has 2,358 nodes and 701,887 edges. The blood transfusion process model includes 102 steps, including 63 exception handling situations, and the corresponding CFG representation has 97,237 nodes and 242,442,845 edges. Even though the blood transfusion model has fewer steps than the chemotherapy model, the exception handling behavior in the blood transfusion process is more complex, requiring a large number of low-level CFG nodes and edges.

3.2 Sequences with synthetic errors

To evaluate the deviation detection approach, we applied it to step sequences containing typical planning errors. Although there is no standardized taxonomy of human errors[1,17], two error kinds appear in the intersection of most of the proposed planning error taxonomies[1,3,18]—omission and commission errors. An omission error occurs when one or more steps are not performed; a commission error occurs when one or more wrong step(s) are performed.

In our initial evaluation of the deviation detection approach, we focused on several kinds of planning errors that involve a single step and on the error of omission of a single subprocess. This decision was based on the errors available from a study of common errors in the blood transfusion process[8] and from conversations with blood transfusion and chemotherapy domain experts.

Single-Step Errors. From each of the two process models, we generated 50 sequences of steps by performing random walks through the process model. To represent "nominal-path" process executions, we also generated 50 sequences from each of the models but disallowed exceptional situations. Thus, we created 100 legal sequences for each process. Statistics about the lengths of the generated sequences are shown in Figure 2.

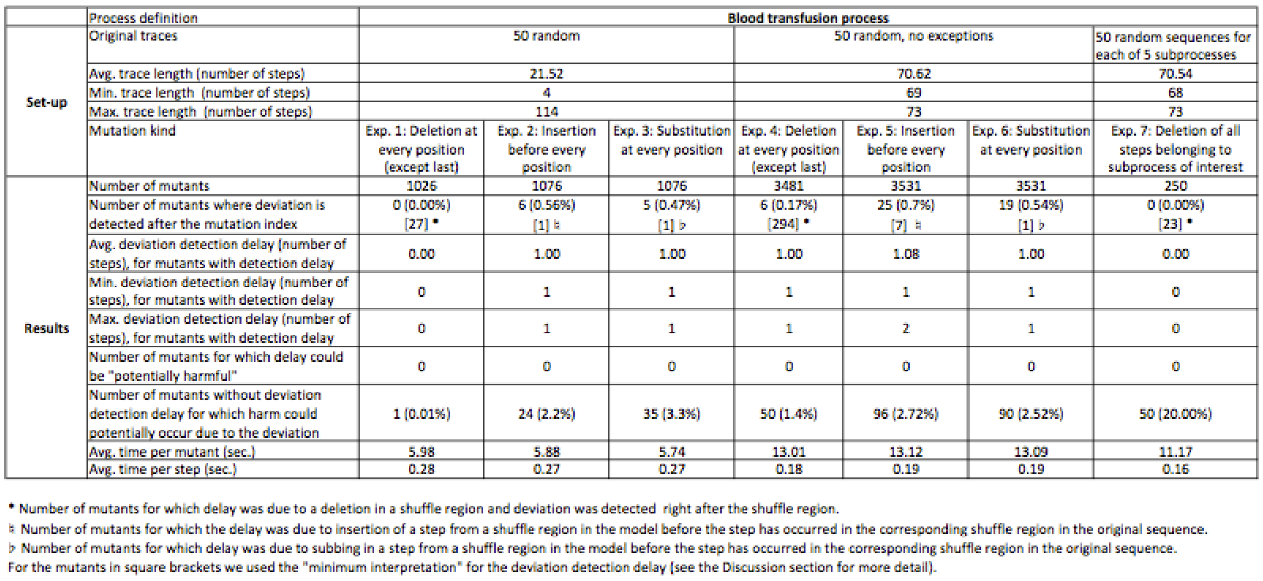
Each generated sequences was then mutated to represent a process execution in which the process performers deviated from the acceptable ways to perform the process. The applied mutations were deletion, insertion, and substitution of a single step at almost[§] every position of each sequence. A mutated sequence is called a *mutant*.

Subprocess Errors. Based on common blood transfusion and chemotherapy errors reported in the medical literature and on our interaction with domain experts involved in these processes, we identified 5 blood transfusion and 5 chemotherapy subprocesses deemed most likely to be omitted. The selected blood transfusion subprocesses were: (1) *ensure correct patient is present* (performed by a nurse before notifying the blood bank to prepare the blood to prevent the possibility the blood product to expire because the patient is not available for transfusion or the wrong patient is in the room); (2) *verify patient ID band* and (3) *verify blood product information* (part of the bedside checks performed by the nurse before beginning the infusion); (4) *assess patient* and (5) *evaluate patient clinically* (part of the clinical evaluation prior to the infusion, performed again by the nurse). The selected chemotherapy subprocesses were: (1) *record height and weight* (performed during patient registration by a clerk); (2) *confirm all necessary information is present* (part of the consultation and assessment, performed by an oncologist before creating the treatment plan and chemotherapy orders); (3) *confirm pretesting has been done* and (4) *confirm existence and not staleness of height/weight data* (part of the treatment plan and orders verifications performed by a Practice Registered Nurse (RN)); (5) *obtain patient informed consent* (performed by a Nurse Practitioner or a Clinic Nurse prior to chemotherapy administration).

For each identified subprocess, we generated 50 sequences from the corresponding process model, such that these sequences contain the subprocess. We then mutated each generated sequence by deleting all steps pertaining to the specific subprocess selected to be omitted. The deviation detection approach was applied to each mutant and various statistics were collected.

[‡]An exception handling situation usually involves multiple steps to deal with the exception. A step (e.g., *assess patient*), can be part of different subprocesses and be involved in different nominal and exceptional situations (e.g., patient develops an allergic reaction).

[§]There were a small number of cases, such as deleting the last step from a legal sequence, where the resulting mutated sequence did not correspond to a sequence with an error. Such cases were excluded from our analysis.



	Process definition	Blood transfusion process						
Set-up	Original traces	50 random			50 random, no exceptions			50 random sequences for each of 5 subprocesses
	Avg. trace length (number of steps)	21.52			70.62			70.54
	Min. trace length (number of steps)	4			69			68
	Max. trace length (number of steps)	114			73			73
	Mutation kind	Exp. 1: Deletion at every position (except last)	Exp. 2: Insertion before every position	Exp. 3: Substitution at every position	Exp. 4: Deletion at every position (except last)	Exp. 5: Insertion before every position	Exp. 6: Substitution at every position	Exp. 7: Deletion of all steps belonging to subprocess of interest
Results	Number of mutants	1026	1076	1076	3481	3531	3531	250
	Number of mutants where deviation is detected after the mutation index	0 (0.00%) [27] *	6 (0.56%) [1] ♮	5 (0.47%) [1] ♭	6 (0.17%) [294] *	25 (0.7%) [7] ♮	19 (0.54%) [1] ♭	0 (0.00%) [23] *
	Avg. deviation detection delay (number of steps), for mutants with detection delay	0.00	1.00	1.00	1.00	1.08	1.00	0.00
	Min. deviation detection delay (number of steps), for mutants with detection delay	0	1	1	1	1	1	0
	Max. deviation detection delay (number of steps), for mutants with detection delay	0	1	1	1	2	1	0
	Number of mutants for which delay could be "potentially harmful"	0	0	0	0	0	0	0
	Number of mutants without deviation detection delay for which harm could potentially occur due to the deviation	1 (0.01%)	24 (2.2%)	35 (3.3%)	50 (1.4%)	96 (2.72%)	90 (2.52%)	50 (20.00%)
	Avg. time per mutant (sec.)	5.98	5.88	5.74	13.01	13.12	13.09	11.17
	Avg. time per step (sec.)	0.28	0.27	0.27	0.18	0.19	0.19	0.16

* Number of mutants for which delay was due to a deletion in a shuffle region and deviation was detected right after the shuffle region.
♮ Number of mutants for which the delay was due to insertion of a step from a shuffle region in the model before the step has occurred in the corresponding shuffle region in the original sequence.
♭ Number of mutants for which delay was due to subbing in a step from a shuffle region in the model before the step has occurred in the corresponding shuffle region in the original sequence.
For the mutants in square brackets we used the "minimum interpretation" for the deviation detection delay (see the Discussion section for more detail).

Figure 2: Applying the deviation detection approach to the blood transfusion process model. The results for the chemotherapy process model are available at http://laser.cs.umass.edu/deviation-detection/amia2014.html.

4 Results

Figure 2 shows the results of applying the deviation detection approach to the blood transfusion process. Due to lack of space, we do not include the results table for the chemotherapy process, but make it available at http://laser.cs.umass.edu/deviation-detection/amia2014.html. The results from both processes were similar and are discussed in the next section. A deviation detection *delay* is defined as the number of steps between the *mutation index* (the index in the sequence of steps where the mutation was done for single-step errors or the index of the first mutation for subprocess errors) and the index where that sequence was recognized as not being a sequence from the process model.

5 Discussion

Delayed Deviation Detection and Potential Harm Due to Delay. Delayed deviation detection occurred infrequently—in less than 1% of the mutants from all experiments. We analyzed each mutated sequence with deviation detection delay by tracing that sequence through the corresponding process model to determine what structure(s) in that model caused the delay. The main cause for the delay was branching in the process models due to exception handling or optional steps/subprocesses. For example, suppose a process can be performed by executing step sequence *abcd*, when there are no exceptional situations. If an exceptional situation arises while performing *b*, however, then the sequence of steps *xyz* should be performed to address this situation before continuing with steps *c* and *d*, resulting in sequence *abxyzcd*. If the sequence *abcd* is mutated by inserting step *x* after step *b*, the deviation cannot be detected until step *c* is performed (which is one step after the mutation index) because *abx* is a prefix of valid sequence through the process model.

Another reason for detecting deviations after the mutation index were *shuffled* steps, which are steps that should be done sequentially but are allowed to occur in any order. For example, suppose that a process is performed by doing step *a*, followed by steps *b*, *c*, and *d* in any order, and then step *e*. Thus, if the step sequence *abcde* is mutated by deleting step *b*, a deviation will not get detected until step *e* is performed because *acd* is a prefix of a valid sequence through the process model. This results in a deviation detection delay of 2, using the measure of delay previously described.

Shuffled steps are different from other branch points in a process where there is more than one step to perform next, because any of the shuffled steps are acceptable to be performed next. Since shuffled steps are common in the processes we studied, we decided that deviation detection delays due to shuffled steps should be measured differently to avoid distorting the results. For example, if a mutation deletes one of several shuffled steps in a sequence of steps, we measure the delay as the number of steps between the last shuffled step in the sequence and the step where a deviation is detected. We call this the *minimum interpretation of the delay* and we use similar minimum interpretations of delay when insertion and substitution mutations involving shuffled steps are performed.

In each of the experiments with the blood transfusion and chemotherapy processes, the average deviation detection delay was small—the largest average was 2.31 steps—and in most cases it was close to 1. In critical processes, such as medical procedures, however, even a delay of 1 could be harmful, if some potentially dangerous step, such as *administer chemotherapy medications*, is performed before detecting the deviation.

To address the question of whether/how often harm might occur due to delayed deviation detection, we identified a set of potentially harmful steps for both processes. These are steps, such that if an error has occurred prior to their performance, performing them could potentially result in immediate harm. Two such steps from the blood transfusion and chemotherapy processes are *begin infusion of blood product* and *administer chemotherapy drug*, respectively.

Having identified the set of potentially harmful steps, we then inspected the mutated sequences described above to determine whether a potentially harmful step occurs between the mutation index and the index of deviation detection. For sequences for which this was the case, we manually analyzed whether the error (the mutation) could affect the potentially harmful step. If harm could occur as a result of the error, we counted the mutated sequence as one for which deviation detection delay could be potentially harmful. There were no cases where the delay could result in harm for the two processes we examined.

Potential Harm When Deviations Are Detected without Delay. We found some mutated sequences for which the deviation is detected at the index of mutation, i.e., there is no deviation detection delay, but harm could potentially still be done (third row from the bottom of the table in Figure 2). In experiments 2, 3, 5, and 6, this was due to mutating a sequence by inserting or substituting in a harmful step. In general, if the error is a commission of a harmful step (e.g., *administer a drug*), a deviation cannot be detected before the harmful step is started because the sequence of performed steps up to the index where the error is made would be a legal sequence through the process model.

In experiments 1, 4, and 7, the cases where potential harm could occur, even when there is no deviation detection delay, were due to omitting a step or a subprocess that a) can immediately precede a harmful step, and b) can affect that harmful step. For example, in the blood transfusion process model, the subprocess of verifying the blood product immediately precedes the step *begin infusion of blood product*. Thus, if steps from this subprocess (such as *ensure that blood product has not expired*) are omitted, or the subprocess is omitted altogether, the deviation cannot be detected until the step *begin infusion of blood product* is started. The 50 cases in experiment 7 of mutants in which harm could occur even when there is no detection delay are due to omitting the subprocess *verify blood product information*.

Such structures represent process vulnerabilities with respect to the proposed deviation detection approach. They also represent vulnerabilities in general, as there is little opportunity for process performers to realize that an error is made before they start a potentially harmful step. A possible strategy to deal with such process vulnerabilities is to introduce a non-harmful step before the potentially harmful one. In fact, such steps are already in place in some medical procedures, such as surgeries where the clinicians performing the surgery are required to stop at certain points of the procedure and confirm that everyone has performed the necessary steps and is aware of the relevant information before proceeding further. The presence of such verification steps would allow the proposed deviation detection approach to detect deviations when such verification steps are performed and before potentially harmful steps are started.

Running time of the deviation detector. The running time results for the blood transfusion process are shown in the last two rows of the table in Figure 2. It took less than 6 seconds to determine whether a sequence of about 21.5 steps, on average, is a deviant; for the mutated sequences where exceptions were disallowed, it took around 13 seconds to determine whether a sequence of about 70.5 steps, on average, is a deviant. This results in less than a third of a second per step in a sequence. The running time of the deviation detector was similarly low for the chemotherapy process. Given that humans usually take more than a third of a second to perform an activity in a medical process, the running time results indicate that the deviation detector could detect deviations in real time before harm is done as a result of a deviation.

Threats to validity. The experimental evaluation was synthetic—we did not monitor a real executing process and did not apply the deviation detection approach to real process executions. The experimental evaluation was based on models of two medical processes. Even though the models were relatively large and complex in terms of covering a large set of process executions (including exceptional executions and concurrency within a single process execution), the results might change if the size or the complexity of the models increases or models of different processes are used.

We mutated the generated sequences by performing single-step deletions, insertions, and substitutions and also deletion of subprocesses. The results might change if mutations that represent different kinds of errors are performed or if multiple errors are considered.

Limitations of proposed deviation detection approach. Even though the preliminary investigation of the proposed deviation detection approach is promising, there are several research challenges that need to be tackled before the approach can be applied in clinical settings. As discussed in section 2.2, the deviation detection approach relies on receiving an accurate sequence of performed steps of interest from the Process Execution Monitor. Capturing what human process performers do in an accurate and timely manner, however, may be difficult. We expect the use of electronic devices in processes will facilitate process execution monitoring. For example, starting to receive infusion data from an infusion pump could be automatically interpreted as having started the step *begin infusion of blood product*; similarly, when patient height and weight data are entered in an electronic medical record, this could be an indication that the step *measure height and weight* was performed.

While the use of electronic devices in processes increases the opportunities for monitoring process executions, events from electronic devices could be misinterpreted. Furthermore, electronic devices cannot capture certain steps in processes, such as cognitive steps (e.g., a doctor making sure the patient information on two artifacts matches). Human scribes, whose participation in medical processes seems to be increasing[4], can capture some steps that electronic devices cannot.

The success of the proposed deviation detection approach depends on the accuracy and the appropriateness of the process model as well. Creating a high-quality process model, however, can be challenging, given the complexity of some medical processes. We have been investigating techniques for eliciting and validating process models and have applied them successfully to several real-world processes[9,13,19]. We believe the criticality of certain processes warrants the time and effort needed to create high-quality process models that can in turn be leveraged to support continuous process improvement via various static analyses (e.g., model checking[20], fault-tree analysis[21], and failure-mode and effets analysis[22]) and to support deviation detection and other aspects of online process guidance (e.g., smart checklist[23]).

6 Conclusion and Future Work

This paper proposes an approach for online deviation detection to catch human errors in complex medical processes before harm is done. The approach relies on a mechanism for monitoring an executing process and on a detailed model that specifies how the process should be performed. We identify important issues related to the feasibility of this approach and evaluate that approach by applying it to detailed realistic models of two medical processes.

The initial evaluation of the deviation detection approach is promising. Delayed deviation detection rarely arose in the studied medical processes and when it did arise, no harm to the patient could occur as a result of the delay. The running time of the deviation detector is low relative to the human speed of performing process activities, suggesting that it could be used for online deviation detection. We also identify some vulnerabilities in the studied process models with respect to online deviation detection and suggest approaches for reducing such vulnerabilities.

The initial investigation of the deviation detection approach has suggested some interesting future research directions. When a deviation is detected, it might be useful to not only inform process performers about it, but to also provide some additional information that could help them identify potential error(s) and plan recovery actions. Examples of such information are possible indexes in the sequence of performed steps where an error might have occurred, as well as a ranked list of potential errors. We are exploring approaches for providing such information.

In this work, we studied the delayed deviation detection issue empirically. It will be useful, however, to also develop analytical approaches for determining an upper bound on possible deviation detection delays given a process model and assumptions about the errors that might occur. We are currently exploring such approaches. We are starting by assuming that only simple errors can occur (e.g., a single omission of a single step) before considering more complex errors and combinations of different errors.

Given the initial evaluation of the deviation detection approach and the recent developments in healthcare that could enable monitoring of executing processes, we believe the proposed approach can augment current approaches for online guidance (such as checklists) in medical processes and help catch errors before harm is done.

Acknowledgments

This material is based upon work supported by the National Science Foundation under awards IIS-1239334 and CMMI-1234070. The authors gratefully acknowledge the contributions of Lee Osterweil, Heather Conboy, Elizabeth Henneman, Jenna Marquard, and of many members of the staff of the D'Amour Center for Cancer Care, who graciously donated their time and expertise.

References

[1] Kohn LT, Corrigan JM, Donaldson MS, editors. To Err is Human: Building a Safer Health System. Washington, DC: National Academies Press; 1999.

[2] Stead WW, Lin HS, editors. Computational Technology for Effective Health Care: Immediate Steps and Strategic Directions. National Academies Press; 2009.

[3] James JT. A New, Evidence-based Estimate of Patient Harms Associated with Hospital Care. Journal of Patient Safety. 2013;9(3):122–128.

[4] Hafner K. A Busy Doctor's Right Hand, Ever Ready to Type. The New York Times. 2014 January 12;.

[5] Wilkinson JM, Leuven KV. Procedure checklist for administering a blood transfusion;.

[6] World Health Organization. Surgical Safety Checklist; 2008.

[7] Mertens WC, Brown DE, Parisi R, Cassells LJ, Naglieri-Prescod D, Higby DJ. Detection, Classification, and Correction of Defective Chemotherapy Orders Through Nursing and Pharmacy Oversight. Journal of Patient Safety. 2008;4(3):195–200.

[8] Henneman EA, Avrunin GS, Clarke LA, Osterweil LJ, Andrzejewski Jr C, Merrigan K, et al. Increasing Patient Safety and Efficiency in Transfusion Therapy Using Formal Process Definitions. Transfusion Medicine Review. 2007 January;21(1):49–57.

[9] Chen B, Avrunin GS, Henneman EA, Clarke LA, Osterweil LJ, Henneman PL. Analyzing medical processes. In: ICSE '08: Proceedings of the 30th International Conference on Software Engineering. ACM; 2008. p. 623–632.

[10] Peleg M, Tu S, Bury J, B MBC, Fox J, Greenes RA, et al. Comparing Computer-Interpretable Guideline Models: A Case-Study Approach. JAMIA. 2002;10:2003.

[11] Fitzgerald M, Cameron P, Mackenzie C, Farrow N, Scicluna P, Gocentas R, et al. Trauma Resuscitation Errors and Computer-Assisted Decision Support. Archives of Surgery. 2011;146(2):218–225.

[12] Avrunin GS, Clarke LA, Osterweil LJ, Christov SC, Chen B, Henneman EA, et al. Experience modeling and analyzing medical processes: UMass/baystate medical safety project overview. In: Proceedings of the 1st ACM International Health Informatics Symposium. IHI '10. New York, NY, USA: ACM; 2010. p. 316–325.

[13] Mertens W, Christov S, Avrunin GS, Clarke LA, Osterweil LJ, Cassells LJ, et al. Using Process Elicitation and Validation to Understand and Improve Chemotherapy Ordering and Delivery. The Joint Commission Journal on Quality and Patient Safety. 2012 November;38(11):497– 505.

[14] Cass AG, Lerner BS, Stanley M Sutton J, McCall EK, Wise A, Osterweil LJ. Little-JIL/Juliette: a process definition language and interpreter. In: ICSE '00: Proceedings of the 22nd International Conference on Software Engineering. ACM; 2000. p. 754–757.

[15] Christov SC, Avrunin GS, Clarke LA, Osterweil LJ, Henneman EA. A benchmark for evaluating software engineering techniques for improving medical processes. In: Proceedings of the 2010 ICSE Workshop on Software Engineering in Health Care. SEHC '10. New York, NY, USA: ACM; 2010. p. 50–56.

[16] DeMillo RA, Lipton RJ, Sayward FG. Hints on Test Data Selection: Help for the Practicing Programmer. Computer. 1978 Apr;11(4):34–41.

[17] Henneman EA, Blank FSJ, Gattasso S, Williamson K, Henneman PL. Testing a classification model for emergency department errors. Journal of Advanced Nursing. 2006;55(1):90–99.

[18] Hollnagel E. The phenotype of erroneous actions. Int J Man-Mach Stud. 1993 July;39:1–32.

[19] Christov SC, Chen B, Avrunin GS, Clarke LA, Osterweil LJ, Brown D, et al. Formally Defining Medical Processes. Methods of Information in Medicine Special Topic on Model-Based Design of Trustworthy Health Information Systems. 2008;47(5):392–398.

[20] Clarke MCEM, Grumberg O, Peled DA. Model Checking. MIT Press; 2000.

[21] Vesely WE, Goldberg FF, Roberts NH, Haasl DF. Fault Tree Handbook (NUREG-0492). U.S. Nuclear Regulatory Commission, Washington, D.C.; 1981.

[22] Stamatis DH. Failure Mode and Effect Analysis: FMEA from Theory to Execution. American Society for Quality; 1995.

[23] Avrunin GS, Clarke LA, Osterweil LJ, Goldman JM, Rausch T. Smart checklists for human-intensive medical systems. In: 42nd International Conference on Dependable Systems and Networks Workshops (DSN-W), Workshop on Open, Resilient, Human-aware, Cyber-physical Systems, IEEE/IFIP; 2012. p. 1–6.