

Automatically Generating Narrative Documentation for Medical Procedures

Lori Clarke, chair
clarke@cs.umass.edu

George Avrunin, committee member
avrunin@cs.umass.edu

Heather Conboy, graduate student mentor
hconboy@cs.umass.edu

Samantha Kolovson, author
skolovso@cs.umass.edu

College of Information and Computer Sciences
University of Massachusetts Amherst
140 Governors Drive
Amherst, MA 01003

ABSTRACT

Title: **Automatically Generating Narrative Documentation for Medical Procedures**

Author: **Samantha Kolovson**

Thesis/Project Type: **Honors Project**

Approved by: **Lori A. Clarke, Computer Science**

Approved by: **George Avrunin, Mathematics**

This manuscript details an approach to automatically generating process narrative and post-process documentation for medical procedures. This approach allows the flexibility needed to make the documentation customizable, accurate, understandable, and therefore useful to medical professionals in practice. It is expected that this documentation will reduce time demands on clinicians currently tasked with producing this documentation, and errors caused by lack of communication, standardization, and copying errors.

1. INTRODUCTION

Medical processes are often complicated, involving a number of healthcare workers under intense pressure, and, in some cases, interacting with sophisticated medical devices. As a result, medical processes are often error prone, taking hundreds of thousands of lives and costing billions of dollars annually [4]. In addition, healthcare clinicians are required to produce handwritten or typed documentation which, given the number of patients they see, the complexity, and the attention to detail necessary, is time consuming and error prone. To improve accuracy and reduce workload, well-crafted, accurate documentation is needed. This project focuses on two types of process documentation. The first is process narrative documentation, which provides a careful description of a process and all its intricacies and which can be used for training and standardization of medical processes. And the second is post-process documentation, which provides a record of what happened when a process was enacted and which can be added to a patient's record. The goal of this project is to automatically generate both types of documentation so that they are customizable, understandable, and therefore useful to healthcare clinicians in practice.

1.1. Problem

Automatically generating both kinds of documentation introduced above, requires, finding a way to:

- 1) Describe a process using the right words and well enough that it can be understood and used effectively.
- 2) Capture what happened during the enactment of a particular process.

For the first problem, I was able to build upon and extend the existing Little-JIL Process Improvement Environment (PIE) [1], which provides a process narrative produced by the Little-JIL Narrator [3]. The existing version of the process narrative, produced by the Narrator, had several drawbacks. It contained computer science jargon, especially in its description of exceptional flow. For example, medical domain experts do not understand what an “exception” and a “handler” are as it relates to their work. They would be more comfortable with language describing a problem or unexpected circumstance and how to recover from it. The process narrative also includes symbols used in the Little-JIL language, which are unfamiliar and confusing to users with no Little-JIL experience. These symbols were elaborated upon in a legend, but only by accompanying them with more computer science terms. If a non-computer scientist sees a green triangle in the process narrative and accesses the legend to find out it indicates a “pre-requisite,” that person is still probably not going to fully understand what that means. Even if users familiar with the word “pre-requisite” they might not understand completely what that means for a process.

For the second problem, the PIE does not contain a tool that captures and records the details of a particular performance of a medical process. Therefore, a new tool needed to be created to produce this post-process documentation. The new tool, which is called the Post Documentation Generator, builds upon the approach taken in implementing the Little-JIL Narrator. Creating this new tool faced the complication that there is no standard format for presenting post-process documentation for medical procedures. The details that need to be captured and recorded differ by process and medical practice.

1.2. Background

The Medical Safety Project is being conducted by computer scientists at the University of Massachusetts, in conjunction with medical professionals at Baystate Medical Center, Mass General Hospital, and the VA Boston Healthcare System, to research and develop software that defines, models, and analyzes medical procedures. The tools created make up the PIE, for which the goal is to improve medical outcomes, such as reducing errors [1]. These tools are focused around the Little-JIL process modeling language that defines and models a process which other tools can analyze. The original systems and goals have recently been built on to address a new problem of monitoring a process while it is executing and providing a Smart Checklist to help guide the medical professionals [2]. The Smart Checklist uses Little-JIL process models to structure the checklist and handle process control flow.

My focus has been on improving the Little-JIL Narrator, created as part of the PIE, and creating the Post Documentation Generator that monitors the process with the Smart Checklist to collect important process information as a record of the process. The rest of this section will describe the three tools used or expanded on in this project: The Little-JIL process modeling language [18], Little-JIL Narrator [3], and the Smart

Checklist [2]. Figure 1 provides a high-level view of how these tools relate to each other.

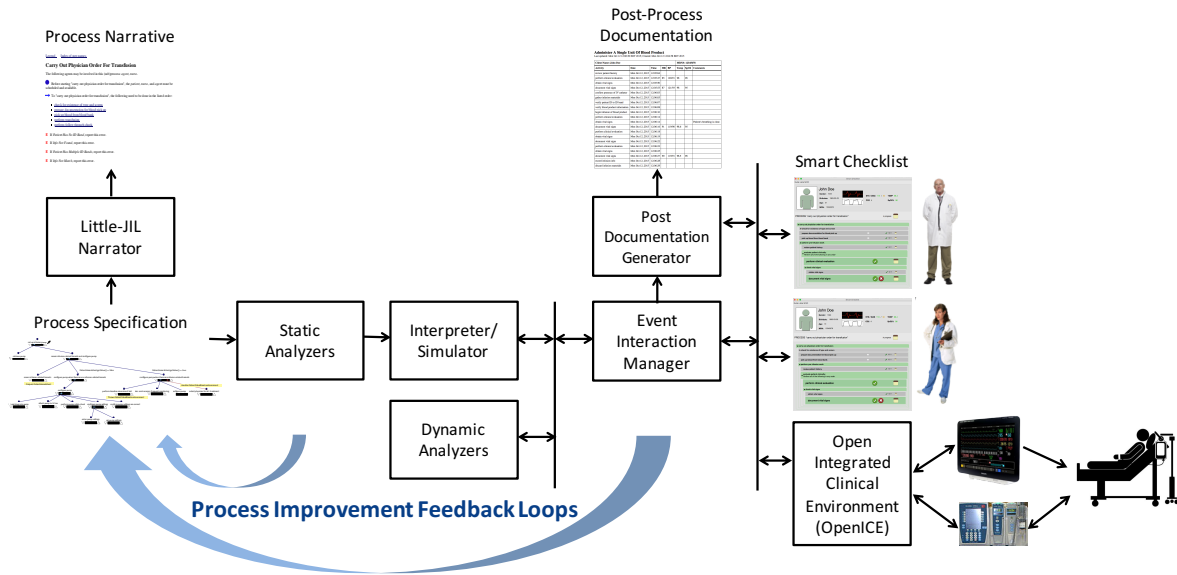


Figure 1: High-Level Concept Diagram

Little-JIL is the language used to define and model a process. A process is a series of steps taken to achieve a desired outcome, which in this context refers to a medical procedure. Little-JIL facilitates the construction of a process model which is a formal representation of a process. These models are able to represent a number of characteristics important to real-world processes: normative and non-normative flow, including exception handling; concurrent activities; artifacts, instances of a type of object that are consumed, produced, or used in a healthcare process [13]; agents, such as doctors and nurses with various specialties, software systems such as electronic health records, and medical devices [1]; and resources, artifacts or agents for which there is contention [13]. The Little-JIL language is intended to enable detailed, precise specification of a wide range of process details, while still supporting the flexibility and freedom of choice

that human agents expect [1]. And it is these rich, well-defined semantics that achieve this goal as well as the capacity to rigorously analyze process models.

Coupled with Little-JIL, the Little-JIL Narrator automatically generates a natural-language process description based on the process model [3]. Such textual representations are typically called a process guides, and in the case of a digital medium, an electronic process guides (EPGs). A process guide is a reference document intended to describe a particular process for the purpose of supporting human enactment and facilitating understanding of that process [8]. This project calls the process guide produced by the Narrator, the process narrative. To produce its process narrative and accompanying table of contents, index of steps, and legend, the Narrator takes a template-based approach. A set of templates of English phrases defines different semantic elements of the process modeling language so the generated process narrative is as precise and complete as the Little-JIL process model from which it is derived [3].

The process narrative, while useful for training, reference, communication, and increased understanding of a process model, it not interactive in a way that provides real-time guidance. A universal tool for guidance is a checklist, a list of action items or criteria arranged in a systematic manner, allowing the user to record the presence/absence of individual items listed to ensure that all are considered or completed [4]. The Smart Checklist walks domain experts, such as doctors and nurses, through a process while it is being performed. It uses a Little-JIL process model to organize the checklist hierarchically, handle non-normative situations, and follow the line of execution through the process model. The process narrative describes every possible step that can be taken

as part of the process, but only one path through the hierarchy is taken during execution. The Smart Checklist can follow any such unique process execution.

1.3. Goal & Motivation

The goal of this project is to automatically generate process narrative and post-process documentation that is customizable, understandable, and therefore useful to healthcare clinicians in practice.

Achieving this goal is motivated by addressing the overall problems faced by the medical community and the needs of the current versions of Little-JIL Narrator and Smart Checklist systems [1, 2, 3]. The automatic generation of documentation should help reduce medical errors by reducing the time demands on healthcare workers and improving the accuracy of the documentation. To reduce medical errors, it is also important that the documentation be useful. This project follows the approach taken when building the Little-JIL Narrator [3] that results in useful documentation by providing the flexibility to customize and add detail to create understanding. Furthermore, the contributions of this project build on this approach and apply it in new ways to improve the current system and create a new tool.

1.4. Example – Blood Transfusion

This section presents part of a Little-JIL process model for an in-patient blood transfusion process. The process model was elicited through collaboration between computer scientists and medical professionals as part of the Blood Transfusion benchmark and is described in full detail in [13]. For the purpose of this project I will focused on the part of

the process where the nurse is performing the actual transfusion and how the process model is presented in the process narrative, the Smart Checklist, and post-procedure documentation.

Little-JIL process models are a hierarchical decomposition of steps. Steps are the main building blocks of a Little-JIL process model, represented in the diagrams by black rectangles. Each step corresponds to an activity performed by human or automated agents. In this sub-process, all leaf steps, steps with no children, are performed by a nurse, and all other steps are initiated by the Smart Checklist.

The root of the sub-process is *carry out physician order for transfusion*, a sequential step, represented by the right arrow in the step rectangle, it indicates that its sub-steps must be performed in left to right order. So first the step *check for existence of type and screen* must be done, followed by *prepare documentation for blood pick up*, *pick up blood from blood bank*, *perform transfusion* and finally *perform follow through check*.

The remainder of this discussion will follow the decomposition of *perform infusion*, which is shown in [Appendices 7.1.2-7](#). *Perform infusion*, shown in [Appendix 7.1.2](#), is a parallel step, represented by the equals symbol in the step rectangle, it indicates that its sub-steps are performed in any order, including simultaneously. In this case, there is just one sub-step, *administer a single unit of blood product*. Also shown in this diagram is the handler step, *discontinue the infusion*. *Administer a single unit of blood product* as well as some of its sub-steps can throw the exception *receive order to discontinue infusion*. When a Little-JIL step throws an exception, the exception propagates up the step hierarchy until a matching exception handler is found, and then

executed [13]. So, when *administer a single unit of blood product* throws *receive order to discontinue infusion* the matching exception handler *discontinue infusion* is found under *perform transfusion* and executed. All three sub-steps of *administer a single unit of blood product*, *perform pre-infusion work*, *infuse unit of blood product*, and *perform post-infusion work*, shown in [Appendix 7.1.3](#), can also throw this exception and the handler is found and executed similarly.

Exception handlers are themselves steps and can thus be hierarchically decomposed to an arbitrary level of detail and can throw exceptions [13]. For example, the three aforementioned sub-steps each have an exception handler *handle suspected transfusion reaction*, shown in [Appendix 7.1.2](#). This handler is executed when the exception *reaction suspected* is thrown by select sub-steps of *perform pre-infusion work*, *infuse unit of blood product*, and *perform post-infusion work*. And it throws *receive order to discontinue infusion*, which of course executes *discontinue infusion*.

This covers an example of non-normative flow in Little-JIL diagrams. There are a few other exception handlers found under *perform pre-infusion work*, shown in [Appendix 7.1.4](#). After *administer a single unit of blood product* the rest of the normative flow can be followed in [Appendices 7.1.4-7](#).

The Little-JIL language is designed to represent real-world processes in the necessary level of detail while remaining concise. It accomplishes this with its rich semantics and simplistic diagrams. And while the language and the diagrams may be easy enough for computer scientists to understand, experts in other domains might prefer a textual representation. The Little-JIL Narrator automatically generates a textual narrative from a

Little-JIL process model. Appendices 7.4 and 7.5 show what the documentation for the blood transfusion would look like if it were generated at this moment.

The legend, shown in [Appendix 7.4](#), serves as a complement to the process narrative, shown in [Appendix 7.5](#), and can be viewed using a link at the top of the process narrative. The first part of the legend is an introductory description which defines process, step, resources, artifact, requisites, sub-steps, exceptions, and agents. Using these terms, the description constructs an understanding of the process narrative structure. Following the introduction is a table defining the Little-JIL symbols that may appear in the process narrative. For example, the right arrow for sequential step is a row in the table with a definition similar to the one given above. While these symbols are not critical to the process narrative, they provide visual structure and are useful if you are working with the process model and narrative concurrently.

In the process narrative, each step has its own section. Each step is identified by its name in a large, bold title at the top of the section. The rest of the section contains several subsections mirroring the step attributes defined in the legend introduction: resources (including agents), artifacts, requisites, sub-steps, and exceptions. Subsections are marked with the Little-JIL symbols defined in the legend table, which provides visual clues to what the subsection is describing. For example, the step section for *Administer Single Unit of Blood Product*. The first subsection says that the *bloodProduct* must be provided to this step. This is further indicated by the down arrow, meaning that this section is listing “in parameters,” artifacts that must be provided to start a step. Below this section are the resources that may be utilized in this step, nurse and patient, indicated by the open blue circle. Then the step’s sub-steps and direction that they must be

performed sequentially. And finally a large subsection with all the exceptions that may be thrown. These are indicated with a red capital “E”. In this case, most of the exceptions do not have handlers and simply say “report this error”. The previously discussed, *receive order to discontinue infusion*, does execute a handler which is cleanly described by saying, “discontinue the infusion and then continue with the next step.” Throughout each step section any referenced steps or handlers are displayed as hyperlinks to enable quick navigation through the long document.

The process narrative is useful for training purposes, establishing standards, and understanding Little-JIL, but as a process guide, it does not provide a record of any process that was performed. It is a static text-based document generated from a static process model. However, the Smart Checklist is a dynamic execution of a process model, which captures information as the process is in motion. Alone the Smart Checklist shows which steps have been completed and when and any notes entered. This information is lost once the application is closed. The Post Documentation Generator gathers this information and more as the Smart Checklist is running and automatically generates documentation in a usable format to be saved as a record. The current generated post-process documentation has two formats, table and narrative, shown in Appendices 7.6.1 and 7.6.2.

[Appendix 7.6.1](#) shows the generated post-process documentation table. The title of the table is the name of the root step in the process, “Carry Out Physician Order For Transfusion,” in this case. Under the title are the timestamps for when the document was created, marking the start of the process, and updated, marking the last time there was an

update to a step. The updated timestamp is useful because the post-documentation can be viewed mid-process and someone might need to view that information quickly. It may also be useful, as further discussed in [Section 5 Conclusions and Future Work](#), if time and date are not featured as columns in the table. The header of the table displays a row with the patient's name and medical record number and a second row with the title of each column. Currently the table features activity, date, time, clinician, and comments columns. Each row is a leaf step in the process model that is completed by a human agent. "Activity" shows the step name, "Date" and "Time" show when the step was completed, "Clinician" shows the clinician that completed the step, and "Comments" shows any comments the clinician entered for that step.

[Appendix 7.6.2](#) shows the generated post-documentation narrative. It is not as full featured as the table because the table was decided to be the more practical format for this documentation. This is further discussed in [Section 5 Conclusion and Future Work](#). The title of the narrative is the same as the table, but it is followed by an introductory statement. This statement declares when the process was started, the patient's name and medical record number, the clinician(s) involved, and the patient's vital signs when the process was started. The bulk of the narrative is one sentence paragraphs corresponding to each leaf step, similar to the rows in the table format. The basic sentence structure is "The [step name] was completed at [time] on [date]."

2. RELATED WORK

There are many drawbacks to the traditional way of describing and communicating processes and the existing tools developed in this area of research [3, 5, 8]. The previous

versions of the Little-JIL Narrator [3] and the Smart Checklist [2] aimed to address many of these issues. There are also many improvements possible through use of checklists in medicine, as studies have shown [4, 7, 14, 15, 16, 17]. This section will focus on some of the drawbacks of previous work and some of the studies that support checklist use in medicine.

Most existing process documentation is deficient in both form and content, leaving users dissatisfied and the documentation unused [8]. Some drawbacks of this documentation, which mainly exist in paper, are:

1. Traditional Guidebooks lack key information [8].
2. Readers of guidebooks can't easily navigate through the pages when their strategy of understanding does not match the document's flow [8].
3. Guidebooks are not easily updated and new versions distributed [3, 8].
4. Guidebooks either contain a mixture of information for different audiences, or multiple documents tailored to specific needs require a common process description [8].
5. Notes attached to paragraphs by one reader cannot be shared with others [8].
6. Guidebooks are not designed to store process status information [8].

The move to EPGs addressed item 2 because many tools were able to automatically generate hyperlinks to facilitate navigation [3]. However, these tools that proceeded the Little-JIL Narrator still lacked an easy update through automatic generation and especially lacked key information [3]. Since the process narrative is generated from a process model, any time the model is updated, the narrative can be regenerated. The lack of information is taken care of by the rich semantics of the Little-JIL language and the template approach. Because of this the process narrative includes a detailed description of control flow, providing support for exceptional behavior, concurrency, synchronization, and recursion [3].

While the Little-JIL Narrator originally provided a common process description and some customization, it did contain a mixture of terminology from different domains and was not able to customize towards one domain. The approach to further customization described in the Section 3 addresses these drawbacks. Additionally, items 5 and 6 above are addressed with the new Post Documentation Generator.

Checklists have been used in healthcare to test a new nurse's capability in performing a procedure [5]. The instructor holds the checklist to verify what the nurse does correctly or incorrectly. The checklist is simple with a column for "Procedure Steps", binary "Yes" and "No" columns, and a "Comments" column. One row contains a numbered step such as "3. Administers any pre-transfusion medications as prescribed." The instructor can check whether or not the nurse performed this step and leave any comments. But why should the nurse perform the whole procedure completely from memory when a detailed checklist exists in the same room?

Medicine is a high-intensity field of work where "levels of cognitive function are often compromised with increasing levels of stress and fatigue" [4]. The result of impaired cognitive function when delivering patient care are errors that could lead to death or harm. Checklists reduce the burden of memory recall and provide standardization and regulation of processes thus reducing errors and improving patient outcomes [4].

Checklists have been slow to become widely adopted in healthcare despite evident benefits and promotion in medical literature [4, 14]. One study evaluating the effects of a comprehensive surgical checklist showed the proportion of patients with one

or more complications falling from 15.4 to 10.6% and mortality from 1.5 to 0.8% - a substantial improvement in outcomes [14, 16]. The Surgical Patient Safety System (SURPASS) checklist used in this study is designed to incorporate all existing protocols and checks, promote communication, and directly prevent adverse events [16]. Another study implemented checklists in an evidence-based intervention to decrease catheter-related bloodstream infection rates in intensive care units (ICUs) [15]. The result was a large and sustained reduction, up to 66%, in rates of infection that was maintained over the course of the study period [15]. Similarly, a third study demonstrates use of a daily “Quality Rounds Checklist” (QRC) in sustaining reduction of ventilator-associated pneumonia [17].

These studies show that using checklists improves patient outcomes, but none of them provide any evidence to suggest checklists would still lead to improvements if they were incorporated as a computer-aid. And with healthcare going digital, it is counterintuitive to introduce paper checklists. An interesting study testing computer-aided decision support during the first 30 minutes of trauma resuscitation demonstrates a reduction in morbidity and aspiration pneumonia [7]. Overall, there was an 5.8% increase in error-free resuscitations from 16 to 21.8% [7].

3. APPROACH

This project’s main goal of automatically generating process documentation that is useful to healthcare clinicians in practice can be further broken down into two parts, each with its own feedback loop. The first part involves improving the existing process narrative documentation generated by the Little-JIL Narrator. The second part involves creating a

new tool, the Post Documentation Generator, to generate post-process documentation from monitoring information gathered through the Smart Checklist. For both parts, soliciting feedback at incremental stages from experts in computer science and healthcare served to keep the project focused.

The following subsections describe my approach to each part of the project. Section 3.1 describes my approach and implementation of the Little-JIL Narrator legend and then how I applied the same ideas to the process narrative. Section 3.2 describes my approach and implementation of the Post Documentation Generator. The last section, 3.3, discusses some of the interesting feedback I received.

3.1. Little-JIL Narrator

3.1.1. Legend

The legend is accessible through a link at the top of the process narrative. Many of the same symbols used in the Little-JIL process models are used in the process narrative. For users who are unfamiliar with Little-JIL symbols, the legend is meant to serve as a key. The original legend, shown in [Appendix 7.2](#), was precisely a key, matching symbols with terms, which was not as useful as intended. Anyone who was unfamiliar with Little-JIL symbols was likely unfamiliar with the terminology represented by the symbols because they are based in computer science.

To assist users in understanding the symbols regardless of their familiarity with Little-JIL and computer science technology, the legend needed to be more than a key. It needed to serve as a glossary of terms with simple explanations of the meaning behind each symbol-term pair. To address this problem, I added a third column to the existing

legend table containing these explanations. I also included an introductory description to aid the understanding of the process narrative structure. The new legend is shown in [Appendix 7.4](#).

The language used in these additions still contained computer science terminology that would be unfamiliar to non-computer scientists. To accommodate different domains and different groups within domains which might prefer customized phrasing, the descriptions could not remain hard-coded. I employed a technique already used in generating the process narrative that would swap out terms that had a “&” in front of them. These terms would be replaced with a term or phrase specified in a separate preference file. For example, “&exception”, could be replaced by “error”. All of these terms are specified in a preference file.

3.1.2. Process Narrative

The process narrative body contained many of the same language problems as the legend. This was especially apparent in the description of exceptional control flow with frequent use of the terms “exception” and “handler”. While the technique for swapping terms was already in place here it need to be expanded to cover these terms and others.

Since these terms were mostly the same used in the legend, the narrative could use similar code. What I found while trying to reuse code was a lot of repetitive code caused bugs and inconsistencies in the generated descriptions. To fix this I refactored the code so all the customizable terms in the legend and the process narrative were swapped by the same piece of code.

3.2. Post Documentation Generator

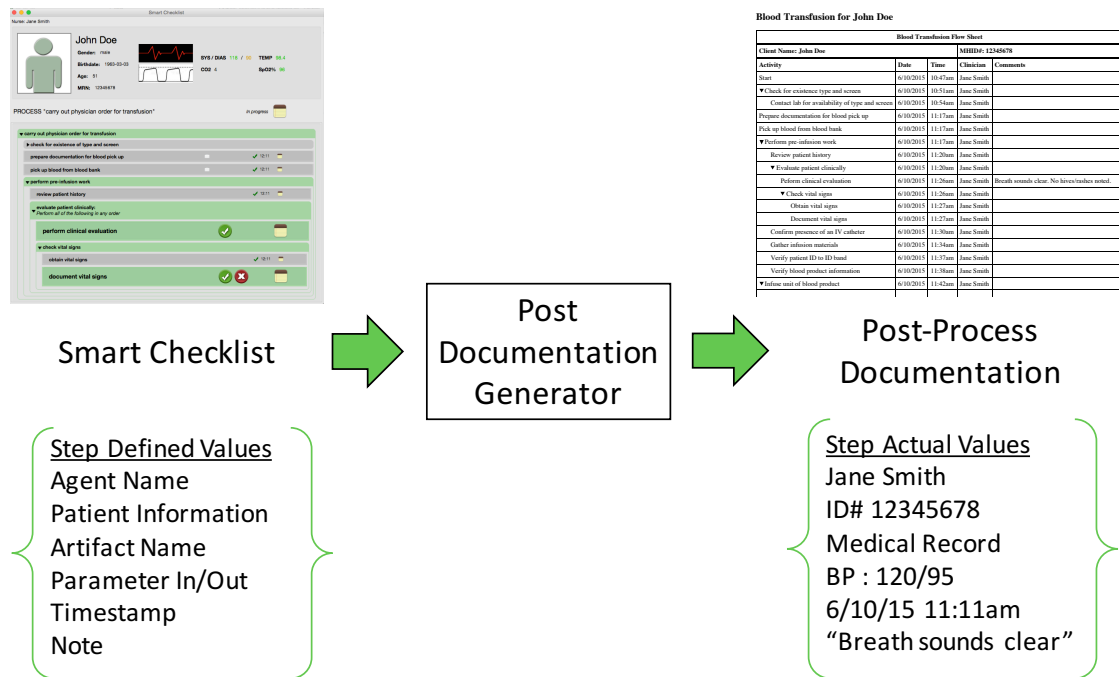


Figure 2: High-level overview of the Post Documentation Generator

Before starting to create the Post Documentation Generator, I needed some idea of what it should be creating. Given there is very little standard for what should be documented procedure to procedure, this presented a challenge. How much information should be included? In what format is this information easily read? Creating mock ups for review by computer scientists and nurses quickly narrowed the focus toward two ideas: a narrative similar to the existing process narrative and a summary table.

With a vision in mind, I moved on to creating a quick proof of concept prototype. The idea behind this and the real generator was that the post-process documentation could be generated in a similar way to the legend and process narrative. First an XML [11] document is generated from information gathered and then it is transformed into the desired format using XSLT [10].

First I wanted to prove I could get the information I needed. I went into the Smart Checklist and found where I could get the information and where I could have it output at the correct time. Figure 2 shows a high-level overview of the information gathered by the Smart Checklist as it monitors a process and what that information looks like generated in the post-process documentation. Once it was clear the information could be retrieved, the next step was to organize it in an XML document.

While this code in the Smart Checklist was not permanent, the XSLT code used for the XML to HTML transformation would be used in the real tool. Writing these files was the next step to show that something could be generated even if it wasn't in the best way. I wrote two files, one for the narrative format and one for the table format. Given that these resemble the HTML from the mockups, they haven't changed much throughout the process.

Once the in-line prototype worked, the code that generated the XML had to be refactored to be separate from the Smart Checklist. Because it was needed for the Smart Checklist functionality, there were already Property Change Listener methods in the executing process model and its steps. Property Change Listeners provide an interface to take an action whenever a specified property changes. All the post documentation tool had to do was listen in to these and the XML document could be produced in the same way. Whenever a step changes state, the XML generator is notified. So far the implementation only includes any action when a step finishes. At this time the XML generator captures the name of the activity, the date, the time, the agent, and any notes the user inputs for that activity. The generated XML document is then fed to both XSLT formats and produces the narrative and table post-process documentation in HTML.

Figure 3 shows on a low-level how the data flows through the system to generate the post-process documentation. The Little-JIL process model is interpreted by the Smart Checklist which represents each step as an agenda item. Each agenda item as a Property Change Listener being listened to by the Post Documentation XML generator. When an agenda item changes state the XML generator adds the information from that item to the XML document. The XML document is subsequently fed to the XSLT formatting weaver that produces the two post documentation formats. As the documentation is produced each time a step changes status, it can be viewed at any time throughout the process execution.

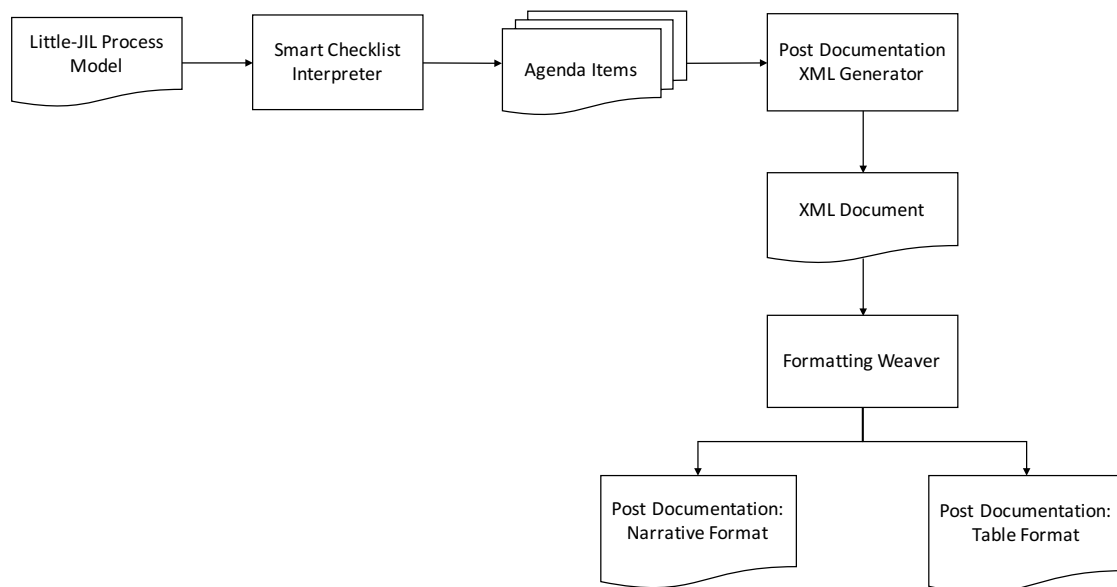


Figure 3: Data flow from the Little-JIL process model through generated post documentation

3.3. Evaluation and Feedback

At various points in the project I received feedback from Professor Beth Henneman, R.N., from the University of Massachusetts School of Nursing, and from Professor Jenna

Marquard from the University of Massachusetts College of Engineering. The original mockups I created were all in narrative format, one example is shown in [Appendix 7.6.7](#). They are difficult to read and it is hard to find any useful information quickly. Because of this Professor Henneman suggested a table format would be more practical. Additionally, she expressed that the purpose of the post-process documentation is to be able to easily spot where a process deviates from the norm. A table format would help fulfill this purpose.

After implementing the Post Documentation Generator, Professors Henneman and Marquard gave feedback on the generated post documentation table. Through discussion it became clear that having vital signs as columns in the table would be ideal ([Appendix 7.6.5](#)). Beth also indicated that there are some steps known to be “important” where vital signs or something else is recorded ([Appendix 7.6.4](#)). These would be the only steps necessary to record.

4. CONTRIBUTIONS

My project makes several contributions to the existing process improvement environment. The original legend was overhauled to be descriptive and customizable. This is helpful to users unfamiliar with computer science and Little-JIL terminology. I applied a similar strategy to the process narrative to give it customization as well.

The main contribution of my project is the Post Documentation Generator. It introduces the dynamic capture of information from the Smart Checklist and two standard formats for generated post-process documentation. At any point during a Smart Checklist

execution both a generated post-process documentation table and narrative can be viewed.

5. CONCLUSION AND FUTURE WORK

This project has made several improvements and additions to the previously existing software. However, there are more changes to be made that were outside the scope of this project. The process narrative needs an overhaul in terms of its sentence structure.

Vocabulary changes could not entirely remove the computer science tone of the documentation. Sentence templates that include real-world sentences for the narrative would enhance domain experts' understanding of the documentation.

The Post Documentation Generator has two areas in which features need to be added, support of Little-JIL language features and the GUI. The current post-process documentation assumes the process takes a normative path with no exceptional situations. This type of documentation would be most useful to clinicians if it includes an automatically generated indication of the process taking a non-normative path, highlighting the exceptions and how they were handled.

Little-JIL and the Smart Checklist are able to handle input and output of artifacts to steps. The Post Documentation Generator must be able to handle these as well. This would function best as another column in the post documentation table. There the artifacts could be listed or linked. Linking to artifacts like forms would be extremely valuable. Also agents are currently added to the post-process documentation as a hard coded name. For any processes that involve multiple agents the ability to record who

performed which steps would be important. To implement this the Smart Checklist needs a specification of who is performing each step taken from the process model. Then the Post Documentation Generator could capture this as well.

To make the Post Documentation Generator more customizable and easier to use for clinicians several GUI improvements could be made. It would be especially useful if users could change the post-documentation table to their specifications through row and column filters. Currently, the generator automatically filters out any non-leaf steps and only records leaf steps as rows in the table. A better way to do this would be to provide an option for if you want all the steps recorded or only leaf steps.

Additionally, the ability to filter columns would make this more useful. Users could select which columns they need or don't need recorded. For example, if the same clinician completes each step of the procedure being documented, the user could choose to filter out that column ([Appendix 7.6.3](#)). Also if the procedure occurred on a single day, the date column could be filtered out ([Appendix 7.6.6](#)). A master document with all the information would need to be kept on record in case an important column was accidentally marked to be filtered. Adding and removing columns doesn't entirely provide the flexibility some users may be looking for. Also being able to change the order of columns in the table would allow users to generate the documentation to their liking. Preferably this would happen through a drag and drop interface.

With some of these features added, especially language feature support like exceptional control flow, the Post Documentation Generator could be evaluated in a study with real nurses in simulation rooms. Nurses would be required to perform a procedure using the Smart Checklist and when they finish view the generated post

documentation. They could provide feedback on if it met their expectations compared to what they would have recorded. Or maybe they would be required to fill out the manual documentation so it could be compared to the generated documentation. This would generate suggestions for improvements and show how useful this documentation could be in the real world. It is important that what is produced addresses the original problems and does not create more.

This project is a good start in addressing the problems with medical documentation. The Little-JIL is now a more usable tool for those less familiar with Little-JIL and the Post Documentation Generator has the potential to be a valuable asset to the Smart Checklist and healthcare clinicians. Going forward, implementing the aforementioned features in the post-process documentation will be important to making it most useful in practice and ready to be evaluated on this usefulness.

6. SOURCES

[1] G.S. Avrunin, L.A. Clarke, L.J. Osterweil, S.C. Christov, B. Chen, E.A. Henneman, P.L. Henneman, L. Cassells, and W. Mertens, "Experience modeling and analyzing medical processes: UMass/Baystate medical safety project overview," *1st ACM International Health Informatics Symposium*. ACM, pp. 316-325, 2010.

[2] G.S. Avrunin, L.A. Clarke, L.J. Osterweil, J.M. Goldman, T. Rausch, "Smart Checklists for human-intensive medical systems," *DSN Workshops*, 2012.

[3] Stefan C. Christov, Tiffany Y. Chao, and Lori A. Clarke. "Generating Natural-language Process Descriptions from Formal Process Models." Department of Computer Science, University of Massachusetts, Amherst, MA 01003, 2012.

[4] B.M. Hales and P.J. Pronovost, "The checklist: a tool for error management and performance improvement," *Journal of Critical Care*, vol. 21, pp. 231-235, 2006.

[5] J.M. Wilkinson and K.V. Leuven. Procedure checklist for administering a blood transfusion. [Online]. Available: http://davisplus.fadavis.com/wilkinson/ProcedureChecklists/PC_Ch36-01.doc

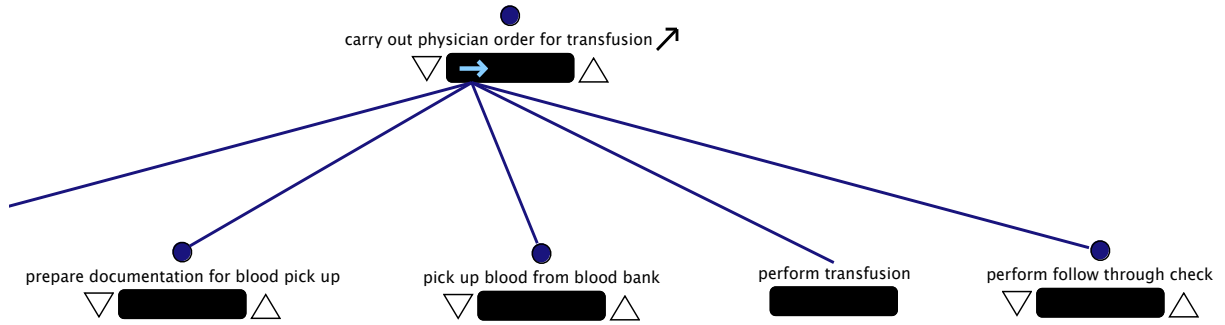
[6] World Health Organization, "Surgical safety checklist," 2008. [Online]. Available: <http://www.who.int/patientsafety/safesurgery/toolsresources/SSSLChecklistfinalJun08.pdf>

- [7] M. Fitzgerald, P. Cameron, C. Mackenzie, et al., "Trauma resuscitation errors and computer-assisted decision support," *Archives of Surgery*, vol. 146, no. 2, pp. 218–225, 2011.
- [8] M. Kellner, U. Becker-Kornstaedt, W. Riddle, J. Tomal, M. Verlage, "Process guides: Effective guidance for process participants," in *Proceedings of the International Conference on the Software Process*, pp. 11-25, 1998.
- [9] A. Wise, et al. "Using Little-JIL to Coordinate Agents in Software Engineering." *Automated Software Engineering Conference (ASE 2000)*, Grenoble, France, pp. 155-165, September 2000.
- [10] XSL Transformations (XSLT) version 2.0, www.w3.org/tr/xslt20, www.w3.org/TR/xslt20
- [11] Extensible Markup Language (XML) version 1.0 (Fifth edition), www.w3.org/tr/xml
- [12] L.J. Osterweil, L.A. Clarke, A.M. Ellison, R. Podorozhny, A. Wise, E. Boose, J. Hadley, "Experience in using a process language to define scientific workflow and generate dataset provenance." *ACM SIGSOFT International Symposium on Foundations of Software Engineering*, November 2008, pp. 319-329.
- [13] G.S. Avrunin, L.A. Clarke, L.J. Osterweil, S.C. Christov, E.A. Henneman, "A Benchmark for Evaluating Software Engineering Techniques for Improving Medical Processes," *International Conference on Software Engineering, Workshop on Software Engineering in Healthcare (SEHC '10)*, Cape Town, South Africa, May 2010.
- [14] J.D. Birkmeyer, "Strategies for improving surgical quality – checklists and beyond," *The New England Journal of Medicine*, vol. 363, pp. 1963-1965, 2010.
- [15] P. Pronovost, D. Needham, S. Berenholtz, D. Sinopoli, H. Chu, S. Cosgrove, B. Sexton, R. Hyzy, R. Welsh, G. Roth, J. Bander, J. Kepros, C. Goeschel, "An Intervention to Decrease Catheter-Related Bloodstream Infections in the ICU," *The New England Journal of Medicine*, vol. 355, pp. 2725-2732, 2006.
- [16] E.N. de Vries, H.A. Prins, R.M.P.H. Crolla, A.J. den Outer, G. van An del, S.H. van Helden, W.S. Schlack, M. Agnes van Putten, D.J. Gourma, M.G.W. Dijkgraaf, S.M. Smorenburg, M.A. Boormeester, "Effect of a Comprehensive Surgical Safety System on Patient Outcomes," *The New England Journal of Medicine*, vol. 363, pp. 1928-1937, 2010.
- [17] J. DuBose, P.G.R. Teixeira, K. Inaba, L. Lam, P. Talving, B. Putty, D. Plurad, D.J. Green, D. Demetriades, H. Belzberg, "Measurable Outcomes of Quality Improvement Using a Daily Quality Rounds Checklist: One-Year Analysis in a Trauma Intensive Care Unit With Sustained Ventilator-Associated Pneumonia Reduction," *The Journal of TRAUMA: Injury, Infection, and Critical Care*, vol. 69, pp. 855-860, 2010.
- [18] A.G. Cass, B.S. Lerner, E.K. McCall, L.J. Osterweil, "Little-JIL/Juliette: A Process Definition Language and Interpreter," in *Proceedings of the 22nd International Conference on Software Engineering*, pp. 754-757, 2000.

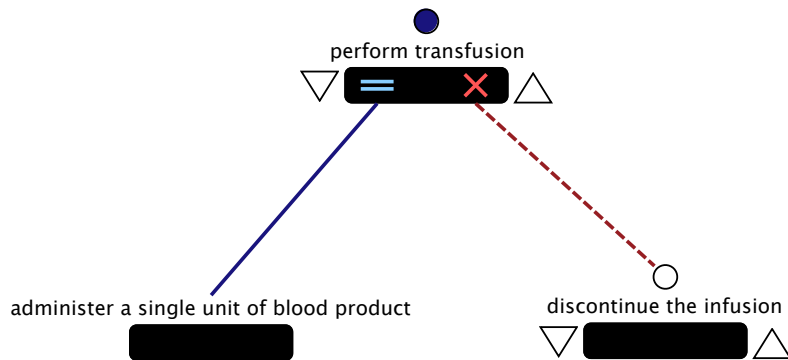
7. APPENDIX

7.1. Little-JIL Process Diagram Examples

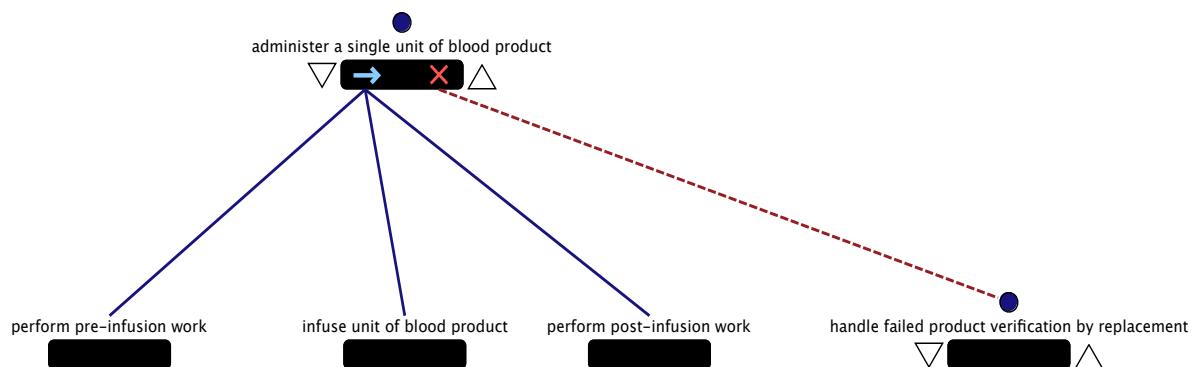
7.1.1. Carry out physician order for infusion



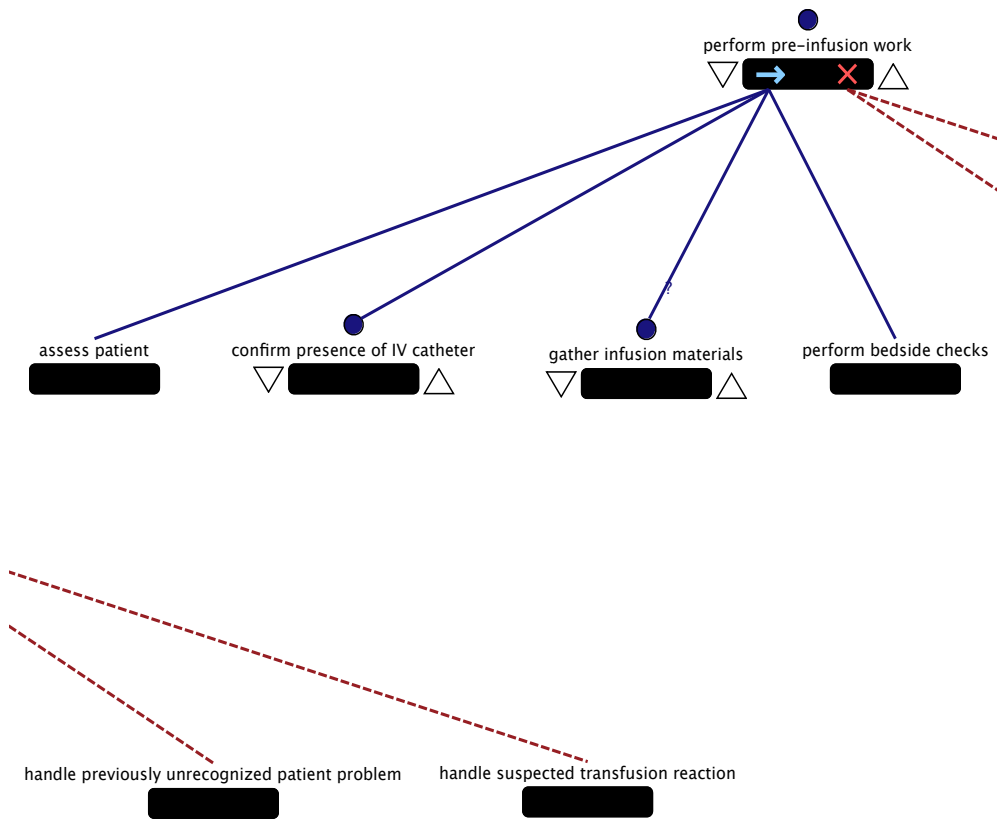
7.1.2. Perform transfusion



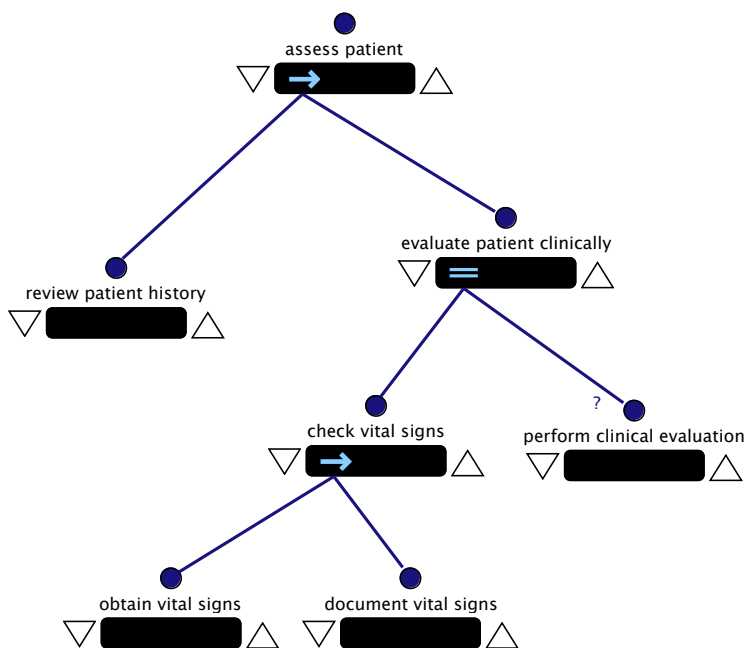
7.1.3. Administer a single unit of blood product



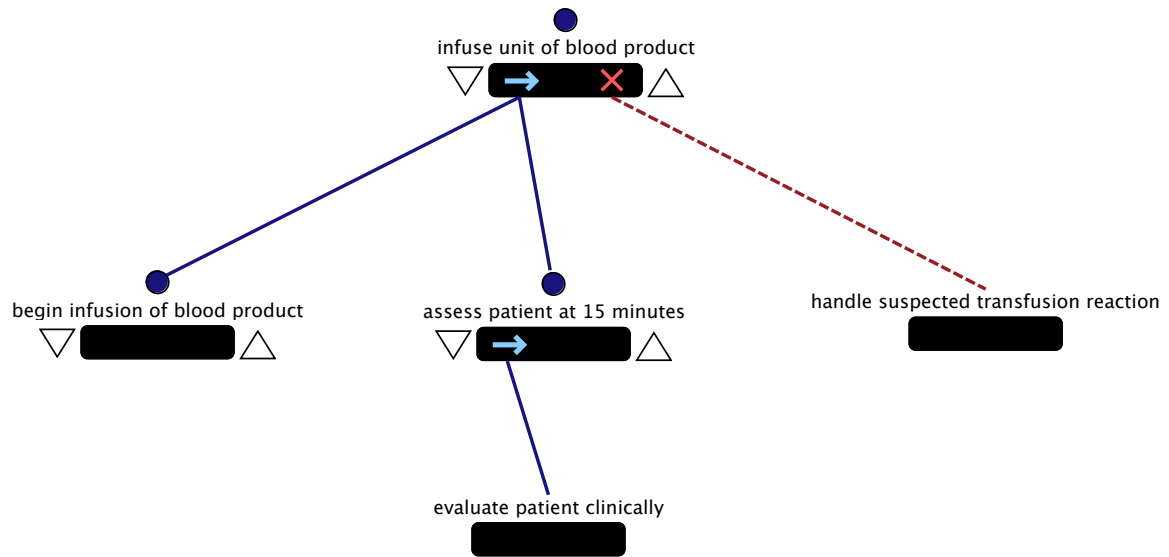
7.1.4. Perform pre-infusion work



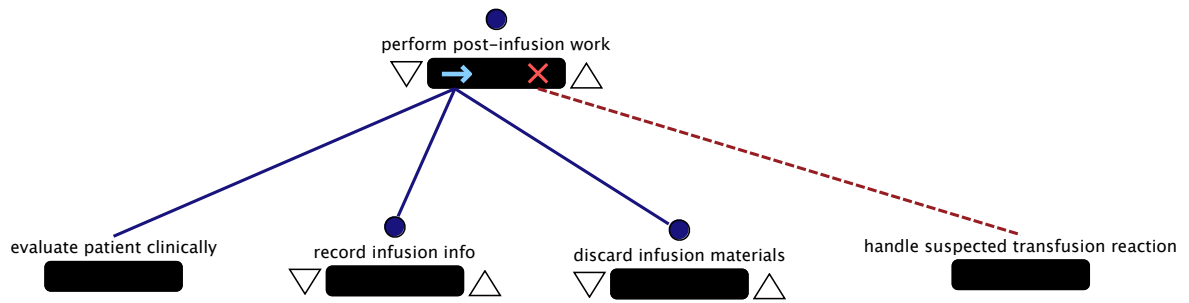
7.1.5. Access patient



7.1.6. Infuse unit of blood product




















7.1.7. Perform post-infusion work



7.2. Old Legend Example

Legend

Symbol	Meaning
	Pre-requisite
	Post-requisite
	In parameter
	Out parameter
	In/out parameter
	Resource acquisition
	Resource use
	Sequential step
	Parallel step
	Choice step
	Try step
	Exception
	Handler
	Rethrow
	Continue
	Restart
	Complete

7.3. Old Narration Example

[Legend](#) [Index of step names](#)

Carry Out Physician Order For Transfusion

- Before starting "carry out physician order for transfusion", the resources *patient*, *nurse*, and *agent* must be acquired.
- ➔ To "carry out physician order for transfusion", the following need to be done in the listed order
 - [check for existence of type and screen](#)
 - [prepare documentation for blood pick up](#)
 - [pick up blood from blood bank](#)
 - [perform transfusion](#)
 - [perform follow through check](#)
- E If *Info Not Found*, then rethrow the exception *Info Not Found*.
- E If *Patient Has No ID Band*, then rethrow the exception *Patient Has No ID Band*.
- E If *Patient Has Multiple ID Bands*, then rethrow the exception *Patient Has Multiple ID Bands*.
- E If *Info Not Match*, then rethrow the exception *Info Not Match*.

Check For Existence Of Type And Screen

- ↓ The *physicianOrder* is required to "check for existence of type and screen".
- ↑ Successful completion of the step "check for existence of type and screen" should yield the *typeAndScreen*.
- The resources *patient* and *nurse* are used in this step.
- ↔ To "check for existence of type and screen", the following should be tried, in the listed order until one succeeds, [contact lab for availability of type and screen](#) or [obtain type and screen](#).
- E If *Info Not Match*, then rethrow the exception *Info Not Match*.
- E If *Info Not Found*, then rethrow the exception *Info Not Found*.
- E If *Patient Has No ID Band*, then rethrow the exception *Patient Has No ID Band*.
- E If *Patient Has Multiple ID Bands*, then rethrow the exception *Patient Has Multiple ID Bands*.

Contact Lab For Availability Of Type And Screen

- ↓ The *physicianOrder* is required to "contact lab for availability of type and screen".
- ↑ Successful completion of the step "contact lab for availability of type and screen" should yield the *existingTypeAndScreen*.
- The resource *agent* is used in this step.
- E If *Lab Unknown*, then continue with the next step.

Obtain Type And Screen

- ↓ The *physicianOrder* is required to "obtain type and screen".
- ↑ Successful completion of the step "obtain type and screen" should yield the *newTypeAndScreen*.
- The resources *patient* and *nurse* are used in this step.
- To "obtain type and screen", [perform blood specimen obtaining process](#) and then [get validation of type and screen from lab](#).
- E If *Info Not Found*, then rethrow the exception *Info Not Found*.
- E If *Info Not Match*, then rethrow the exception *Info Not Match*.
- E If *Patient Has Multiple ID Bands*, then rethrow the exception *Patient Has Multiple ID Bands*.
- E If *Patient Has No ID Band*, then rethrow the exception *Patient Has No ID Band*.

Perform Blood Specimen Obtaining Process

- ↓ The *physicianOrder* is required to "perform blood specimen obtaining process".
- ↑ Successful completion of the step "perform blood specimen obtaining process" should yield the *bloodSpecimen*.
- The resources *patient* and *agent* are used in this step.
- E If *Patient Has Multiple ID Bands*, then rethrow the exception *Patient Has Multiple ID Bands*.
- E If *Info Not Found*, then rethrow the exception *Info Not Found*.
- E If *Info Not Match*, then rethrow the exception *Info Not Match*.
- E If *Patient Has No ID Band*, then rethrow the exception *Patient Has No ID Band*.

Get Validation Of Type And Screen From Lab

- ↓ The *physicianOrder* and *bloodSpecimen* are required to "get validation of type and screen from lab".
- ↑ Successful completion of the step "get validation of type and screen from lab" should yield the *validatedTypeAndScreen*.
- The resource *agent* is used in this step.

Prepare Documentation For Blood Pick Up

- ↓ The *physicianOrder* is required to "prepare documentation for blood pick up".
- ↑ Successful completion of the step "prepare documentation for blood pick up" should yield the *bloodProductDoc*.
- The resource *agent* is used in this step.

Pick Up Blood From Blood Bank

- ↓ The *bloodProductDoc* is required to "pick up blood from blood bank".
- ↑ Successful completion of the step "pick up blood from blood bank" should yield the *bloodProduct*.
- The resource *agent* is used in this step.

Perform Transfusion

- ↓ The *bloodProduct* is required to "perform transfusion".
- The resources *patient* and *nurse* are used in this step.
- To "perform transfusion", the following need to be done in any order (including simultaneously), administer a single unit of blood product.
- E If *Patient Has Multiple ID Bands*, then rethrow the exception *Patient Has Multiple ID Bands*.
- E If *Info Not Found*, then rethrow the exception *Info Not Found*.
- E If *Info Not Match*, then rethrow the exception *Info Not Match*.
- E If *Patient Has No ID Band*, then rethrow the exception *Patient Has No ID Band*.

Administer A Single Unit Of Blood Product

↓ The *bloodProduct* is required to "administer a single unit of blood product".

○ The resources *patient* and *nurse* are used in this step.

→ To "administer a single unit of blood product", [perform pre-infusion work](#), then [infuse unit of blood product](#), and finally [perform post-infusion work](#).

E If *Patient Has No ID Band*, then rethrow the exception *Patient Has No ID Band*.

E If *Info Not Match*, then rethrow the exception *Info Not Match*.

E If *Patient Has Multiple ID Bands*, then rethrow the exception *Patient Has Multiple ID Bands*.

E If *Receive Order To Discontinue Infusion*, then [discontinue the infusion](#) and then continue with the next step.

E If *Info Not Found*, then rethrow the exception *Info Not Found*.

Perform Pre-Infusion Work

↓ The *bloodProduct* is required to "perform pre-infusion work".

○ The resources *patient* and *nurse* are used in this step.

→ To "perform pre-infusion work", the following need to be done in the listed order

- [assess patient](#)
- [confirm presence of IV catheter](#)
- [gather infusion materials](#)
 - This step is optional.
- [perform bedside checks](#)

E If *Receive Order To Discontinue Infusion*, then rethrow the exception *Receive Order To Discontinue Infusion*.

E If *Info Not Match*, then rethrow the exception *Info Not Match*.

E If *Patient Has Multiple ID Bands*, then rethrow the exception *Patient Has Multiple ID Bands*.

E If *Failed Product Check*, then [handle failed product verification by replacement](#) and then continue with the next step.

E If *Patient Has No ID Band*, then rethrow the exception *Patient Has No ID Band*.

E If *Info Not Found*, then rethrow the exception *Info Not Found*.

Assess Patient

○ The resources *patient* and *nurse* are used in this step.

→ To "assess patient", [review patient history](#) and then [evaluate patient clinically](#).

E If *Reaction Suspected*, then [handle suspected transfusion reaction](#) and then restart "perform pre-infusion work".

E If *Problem Found In Patient History*, then [handle previously unrecognized patient problem](#) and then restart "perform pre-infusion work".

Review Patient History

○ The resource *agent* is used in this step.

E If *Problem Found In Patient History*, then rethrow the exception *Problem Found In Patient History*.

Evaluate Patient Clinically

○ The resources *patient* and *nurse* are used in this step.

→ To "evaluate patient clinically", the following need to be done in any order (including simultaneously), [check vital signs](#) and [perform clinical evaluation](#) (This step is optional.) .

E If *Reaction Suspected*, then rethrow the exception *Reaction Suspected*.

Check Vital Signs

○ The resources *patient* and *nurse* are used in this step.

→ To "check vital signs", [obtain vital signs](#) and then [document vital signs](#).

E If *Reaction Suspected*, then rethrow the exception *Reaction Suspected*.

Obtain Vital Signs

↓ The *vitals* is required to "obtain vital signs".

○ The resources *patient* and *agent* are used in this step.

Document Vital Signs

↑ Successful completion of the step "document vital signs" should yield the *vitals*.

○ The resource *agent* is used in this step.

E If *Reaction Suspected*, then rethrow the exception *Reaction Suspected*.

Perform Clinical Evaluation

- The resources *patient* and *agent* are used in this step.

Confirm Presence Of IV Catheter

- The resources *patient* and *agent* are used in this step.

Gather Infusion Materials

- The resource *agent* is used in this step.

Perform Bedside Checks

- ↓ The *bloodProduct* is required to "perform bedside checks".

- The resources *patient* and *nurse* are used in this step.

→ To "perform bedside checks", [verify patient ID to ID band](#) and then [verify blood product information](#).

E If *Info Not Match*, then rethrow the exception *Info Not Match*.

E If *Patient Has No ID Band*, then rethrow the exception *Patient Has No ID Band*.

E If *Failed Product Check*, then rethrow the exception *Failed Product Check*.

E If *Patient Has Multiple ID Bands*, then rethrow the exception *Patient Has Multiple ID Bands*.

E If *Info Not Found*, then rethrow the exception *Info Not Found*.

Verify Patient ID To ID Band

- The resources *patient* and *agent* are used in this step.

E If *Info Not Match*, then rethrow the exception *Info Not Match*.

E If *Patient Has Multiple ID Bands*, then rethrow the exception *Patient Has Multiple ID Bands*.

E If *Info Not Found*, then rethrow the exception *Info Not Found*.

E If *Patient Has No ID Band*, then rethrow the exception *Patient Has No ID Band*.

Verify Blood Product Information

- ↓ The *bloodProduct* is required to "verify blood product information".

- The resource *agent* is used in this step.

E If *Failed Product Check*, then rethrow the exception *Failed Product Check*.

Handle Previously Unrecognized Patient Problem

○ The resources *patient* and *agent* are used in this step.

Handle Suspected Transfusion Reaction

○ The resources *patient* and *agent* are used in this step.

E If *Receive Order To Discontinue Infusion*, then rethrow the exception *Receive Order To Discontinue Infusion*.

Infuse Unit Of Blood Product

↓ The *bloodProduct* is required to "infuse unit of blood product".

○ The resources *patient* and *nurse* are used in this step.

→ To "infuse unit of blood product", [begin infusion of blood product](#) and then [assess patient at 15 minutes](#).

E If *Receive Order To Discontinue Infusion*, then rethrow the exception *Receive Order To Discontinue Infusion*.

Begin Infusion Of Blood Product

↓ The *bloodProduct* is required to "begin infusion of blood product".

○ The resources *patient* and *agent* are used in this step.

Assess Patient At 15 Minutes

○ The resources *patient* and *nurse* are used in this step.

→ To "assess patient at 15 minutes", [evaluate patient clinically](#).

E If *Reaction Suspected*, then [handle suspected transfusion reaction](#) and then continue with the next step.

Perform Post-Infusion Work

○ The resources *patient* and *nurse* are used in this step.

→ To "perform post-infusion work", [evaluate patient clinically](#), then [record infusion info](#), and finally [discard infusion materials](#).

E If *Failed Product Check*, then rethrow the exception *Failed Product Check*.

Handle Previously Unrecognized Patient Problem

○ The resources *patient* and *agent* are used in this step.

Handle Suspected Transfusion Reaction

○ The resources *patient* and *agent* are used in this step.

E If *Receive Order To Discontinue Infusion*, then rethrow the exception *Receive Order To Discontinue Infusion*.

Infuse Unit Of Blood Product

↓ The *bloodProduct* is required to "infuse unit of blood product".

○ The resources *patient* and *nurse* are used in this step.

→ To "infuse unit of blood product", [begin infusion of blood product](#) and then [assess patient at 15 minutes](#).

E If *Receive Order To Discontinue Infusion*, then rethrow the exception *Receive Order To Discontinue Infusion*.

Begin Infusion Of Blood Product

↓ The *bloodProduct* is required to "begin infusion of blood product".

○ The resources *patient* and *agent* are used in this step.

Assess Patient At 15 Minutes

○ The resources *patient* and *nurse* are used in this step.

→ To "assess patient at 15 minutes", [evaluate patient clinically](#).

E If *Reaction Suspected*, then [handle suspected transfusion reaction](#) and then continue with the next step.

Perform Post-Infusion Work

○ The resources *patient* and *nurse* are used in this step.

→ To "perform post-infusion work", [evaluate patient clinically](#), then [record infusion info](#), and finally [discard infusion materials](#).

E If *Receive Order To Discontinue Infusion*, then rethrow the exception *Receive Order To Discontinue Infusion*.

Record Infusion Info

- ☐ The resource *agent* is used in this step.

Discard Infusion Materials

- ☐ The resource *agent* is used in this step.

Handle Failed Product Verification By Replacement

↑ Successful completion of the step "handle failed product verification by replacement" should yield the *newBloodProduct*.

- ☐ The resource *agent* is used in this step.

Discontinue The Infusion

Perform Follow Through Check

- ☐ The resources *patient* and *agent* are used in this step.











7.4 New/Current Legend Example









Legend

The overall **process** is described as a hierarchical decomposition of **steps** that must be performed to accomplish its goals. A step may be further elaborated by hierarchically decomposing it into sub-steps. A step can specify:

- resources (e.g., personnel, equipment) that may be allocated (i.e. scheduled and available) to take part in that step
- artifacts (e.g., medical record, diagnostic test order, after visit summary) that must be input to or output by that step
- pre- or post-requisite steps that must be successfully completed before or after a step respectively
- sub-steps that must be performed to complete that step
- exceptions (e.g., WrongPatient, MissingDiagnosticTestResults) that may occur when the step is performed

A step must designate one of the resources as an **agent**, which is the human or automated component responsible for performing that step. To perform a step, the assigned agent starts the step and then later completes it. When an elaborated step is started, its sub-steps are started. The elaborated step completes after the sub-steps complete. Each step successfully completes or fails to complete when one or more errors occur.

Symbol	Term	Meaning
	Resource Allocation	Before starting a step, the listed resource(s) may be assigned to take part in that step.
	Resource Utilization	To complete a step, the listed resource(s) may be employed.
	In parameter	To start a step, the listed artifact(s) must be provided.
	Out parameter	To successfully complete a step, the listed artifact(s) must be produced.
	In/out parameter	To start a step, the listed artifact(s) must be provided. During a step, the artifact(s) may be modified. To successfully complete a step, the artifact(s) must be produced.
	Pre-requisite	Before starting a step, pre-conditions must be satisfied or pre-processing steps must be successfully completed before performing the listed sub-steps.
	Post-requisite	After successfully completing a step, post-conditions must be satisfied or post-processing steps must be successfully completed after successfully completing the listed sub-steps.
	Sequential step	To perform a sequential step, the listed sub-steps must be performed in the given order until all of them are completed.
	Parallel step	To perform a parallel step, the listed sub-steps are performed in any order (including simultaneously) until all of them are completed.
	Choice	To perform a choice step, the listed sub-steps are performed in any order

	step	(including simultaneously) until one of them is completed successfully.
	Try step	To perform a try step, the listed sub-steps are performed in the given order until one of them is completed successfully.
	Exception	If an error occurs when a given step's sub-step is performed, that step fails to complete and reports the error.
	Handler	When a sub-step report that an error occurred, the error must be recovered from before continuing to perform the step.
	Rethrow	After failing to recover from an error reported by a step's sub-step, that step fails to complete and reports that error.
	Continue	After successfully recovering from an error reported by the current step's sub-step, that step continues to the next sub-step.
	Restart	After successfully recovering from an error reported by the current step's sub-step, that step is started again from its first sub-step.
	Complete	After successfully recovering from an error reported by a step's sub-step, that step is completed successfully.

7.5. New/Current Narration Example

[Legend](#) [Index of step names](#)

Carry Out Physician Order For Transfusion

The following agents may be involved in this (sub)process: *agent, nurse*.

● Before starting "carry out physician order for transfusion", the *patient, nurse*, and *agent* must be scheduled and available.

➔ To "carry out physician order for transfusion", the following need to be done in the listed order:

- [check for existence of type and screen](#)
- [prepare documentation for blood pick up](#)
- [pick up blood from blood bank](#)
- [perform transfusion](#)
- [perform follow through check](#)

E If *Patient Has No ID Band*, report this error.

E If *Info Not Found*, report this error.

E If *Patient Has Multiple ID Bands*, report this error.

E If *Info Not Match*, report this error.

Check For Existence Of Type And Screen

↓ The *physicianOrder* must be provided to "check for existence of type and screen".

↑ Successful completion of the step "check for existence of type and screen" should produce the *typeAndScreen*.

○ The *nurse* and *patient* may be utilized in this step.

↔ To "check for existence of type and screen", the following should be tried, in the listed order until one succeeds, [contact lab for availability of type and screen](#) or [obtain type and screen](#).

E If *Info Not Match*, report this error.

E If *Patient Has Multiple ID Bands*, report this error.

E If *Info Not Found*, report this error.

E If *Patient Has No ID Band*, report this error.

Contact Lab For Availability Of Type And Screen

- ↓ The *physicianOrder* must be provided to "contact lab for availability of type and screen".
- ↑ Successful completion of the step "contact lab for availability of type and screen" should produce the *existingTypeAndScreen*.
- The *agent* may be utilized in this step.
- E** If *Lab Unknown*, continue with the next step.

Obtain Type And Screen

- ↓ The *physicianOrder* must be provided to "obtain type and screen".
- ↑ Successful completion of the step "obtain type and screen" should produce the *newTypeAndScreen*.
- The *patient* and *nurse* may be utilized in this step.
- To "obtain type and screen", [perform blood specimen obtaining process](#) and then [get validation of type and screen from lab](#).
- E** If *Patient Has Multiple ID Bands*, report this error.
- E** If *Info Not Match*, report this error.
- E** If *Info Not Found*, report this error.
- E** If *Patient Has No ID Band*, report this error.

Perform Blood Specimen Obtaining Process

- ↓ The *physicianOrder* must be provided to "perform blood specimen obtaining process".
- ↑ Successful completion of the step "perform blood specimen obtaining process" should produce the *bloodSpecimen*.
- The *agent* and *patient* may be utilized in this step.
- E** If *Patient Has Multiple ID Bands*, report this error.
- E** If *Patient Has No ID Band*, report this error.
- E** If *Info Not Match*, report this error.
- E** If *Info Not Found*, report this error.

Get Validation Of Type And Screen From Lab

↓ The *physicianOrder* and *bloodSpecimen* must be provided to "get validation of type and screen from lab".

↑ Successful completion of the step "get validation of type and screen from lab" should produce the *validatedTypeAndScreen*.

○ The *agent* may be utilized in this step.

Prepare Documentation For Blood Pick Up

↓ The *physicianOrder* must be provided to "prepare documentation for blood pick up".

↑ Successful completion of the step "prepare documentation for blood pick up" should produce the *bloodProductDoc*.

○ The *agent* may be utilized in this step.

Pick Up Blood From Blood Bank

↓ The *bloodProductDoc* must be provided to "pick up blood from blood bank".

↑ Successful completion of the step "pick up blood from blood bank" should produce the *bloodProduct*.

○ The *agent* may be utilized in this step.

Perform Transfusion

↓ The *bloodProduct* must be provided to "perform transfusion".

○ The *patient* and *nurse* may be utilized in this step.

= To "perform transfusion", the following need to be done in any order (including simultaneously), administer a single unit of blood product.

E If *Patient Has No ID Band*, report this error.

E If *Patient Has Multiple ID Bands*, report this error.

E If *Info Not Match*, report this error.

E If *Info Not Found*, report this error.

Administer A Single Unit Of Blood Product

↓ The *bloodProduct* must be provided to "administer a single unit of blood product".

○ The *nurse* and *patient* may be utilized in this step.

→ To "administer a single unit of blood product", [perform pre-infusion work](#), then [infuse unit of blood product](#), and finally [perform post-infusion work](#).

E If *Patient Has Multiple ID Bands*, report this error.

E If *Info Not Found*, report this error.

E If *Patient Has No ID Band*, report this error.

E If *Receive Order To Discontinue Infusion*, [discontinue the infusion](#) and then continue with the next step.

E If *Info Not Match*, report this error.

Perform Pre-Infusion Work

↓ The *bloodProduct* must be provided to "perform pre-infusion work".

○ The *nurse* and *patient* may be utilized in this step.

→ To "perform pre-infusion work", the following need to be done in the listed order:

- [assess patient](#)
- [confirm presence of IV catheter](#)
- [gather infusion materials](#)
 - This step is optional
- [perform bedside checks](#)

E If *Failed Product Check*, [handle failed product verification by replacement](#) and then continue with the next step.

E If *Patient Has No ID Band*, report this error.

E If *Receive Order To Discontinue Infusion*, report this error.

E If *Info Not Found*, report this error.

E If *Patient Has Multiple ID Bands*, report this error.

E If *Info Not Match*, report this error.

Assess Patient

○ The *nurse* and *patient* may be utilized in this step.

➔ To "assess patient", [review patient history](#) and then [evaluate patient clinically](#).

E If *Reaction Suspected*, [handle suspected transfusion reaction](#) and then restart "perform pre-infusion work".

E If *Problem Found In Patient History*, [handle previously unrecognized patient problem](#) and then restart "perform pre-infusion work".

Review Patient History

○ The *agent* may be utilized in this step.

E If *Problem Found In Patient History*, report this error.

Evaluate Patient Clinically

○ The *patient* and *nurse* may be utilized in this step.

➔ To "evaluate patient clinically", the following need to be done in any order (including simultaneously), [check vital signs](#) and [perform clinical evaluation](#) (This step is optional) .

E If *Reaction Suspected*, report this error.

Check Vital Signs

○ The *patient* and *nurse* may be utilized in this step.

➔ To "check vital signs", [obtain vital signs](#) and then [document vital signs](#).

E If *Reaction Suspected*, report this error.

Obtain Vital Signs

○ The *patient* and *agent* may be utilized in this step.

Document Vital Signs

○ The *agent* may be utilized in this step.

E If *Reaction Suspected*, report this error.

Perform Clinical Evaluation

- The *agent* and *patient* may be utilized in this step.

Confirm Presence Of IV Catheter

- The *agent* and *patient* may be utilized in this step.

Gather Infusion Materials

- The *agent* may be utilized in this step.

Perform Bedside Checks

- ↓ The *bloodProduct* must be provided to "perform bedside checks".
- The *nurse* and *patient* may be utilized in this step.
- ➔ To "perform bedside checks", [verify patient ID to ID band](#) and then [verify blood product information](#).
- E** If *Failed Product Check*, report this error.
- E** If *Info Not Found*, report this error.
- E** If *Patient Has No ID Band*, report this error.
- E** If *Info Not Match*, report this error.
- E** If *Patient Has Multiple ID Bands*, report this error.

Verify Patient ID To ID Band

- The *agent* and *patient* may be utilized in this step.
- E** If *Info Not Found*, report this error.
- E** If *Info Not Match*, report this error.
- E** If *Patient Has No ID Band*, report this error.
- E** If *Patient Has Multiple ID Bands*, report this error.

Verify Blood Product Information

- ↓ The *bloodProduct* must be provided to "verify blood product information".
- The *agent* may be utilized in this step.

E If *Failed Product Check*, report this error.

Handle Previously Unrecognized Patient Problem

○ The *agent* and *patient* may be utilized in this step.

Handle Suspected Transfusion Reaction

○ The *agent* and *patient* may be utilized in this step.

E If *Receive Order To Discontinue Infusion*, report this error.

Infuse Unit Of Blood Product

↓ The *bloodProduct* must be provided to "infuse unit of blood product".

○ The *patient* and *nurse* may be utilized in this step.

→ To "infuse unit of blood product", [begin infusion of blood product](#) and then [assess patient at 15 minutes](#).

E If *Receive Order To Discontinue Infusion*, report this error.

Begin Infusion Of Blood Product

↓ The *bloodProduct* must be provided to "begin infusion of blood product".

○ The *agent* and *patient* may be utilized in this step.

Assess Patient At 15 Minutes

○ The *nurse* and *patient* may be utilized in this step.

→ To "assess patient at 15 minutes", [evaluate patient clinically](#).

E If *Reaction Suspected*, [handle suspected transfusion reaction](#) and then continue with the next step.

Perform Post-Infusion Work

○ The *patient* and *nurse* may be utilized in this step.

→ To "perform post-infusion work", [evaluate patient clinically](#), then [record infusion info](#), and finally [discard infusion materials](#).

E If *Receive Order To Discontinue Infusion*, report this error.

Record Infusion Info

○ The *agent* may be utilized in this step.

Discard Infusion Materials

○ The *agent* may be utilized in this step.

Handle Failed Product Verification By Replacement

↑ Successful completion of the step "handle failed product verification by replacement" should produce the *newBloodProduct*.

○ The *agent* may be utilized in this step.

Discontinue The Infusion

Perform Follow Through Check

○ The *patient* and *agent* may be utilized in this step.

7.6. Post Documentation Example(s)

7.6.1. Current generated post-procedure table

Carry Out Physician Order For Transfusion

Last updated: Sun Jan 03 12:13:28 EST 2016; Created: Sun Jan 03 12:10:17 EST 2016

Client Name: John Doe			MRN#: 12345678	
Activity	Date	Time	Clinician	Comments
contact lab for availability of type and screen	Sun Jan 03, 2016	12:10:22	Jane Smith	
prepare documentation for blood pick up	Sun Jan 03, 2016	12:10:25	Jane Smith	
pick up blood from blood bank	Sun Jan 03, 2016	12:10:27	Jane Smith	
review patient history	Sun Jan 03, 2016	12:10:30	Jane Smith	
perform clinical evaluation	Sun Jan 03, 2016	12:10:51	Jane Smith	Patient's breathing is clear.
obtain vital signs	Sun Jan 03, 2016	12:10:52	Jane Smith	
document vital signs	Sun Jan 03, 2016	12:10:54	Jane Smith	
confirm presence of IV catheter	Sun Jan 03, 2016	12:10:57	Jane Smith	
gather infusion materials	Sun Jan 03, 2016	12:11:00	Jane Smith	
verify patient ID to ID band	Sun Jan 03, 2016	12:11:03	Jane Smith	
verify blood product information	Sun Jan 03, 2016	12:11:07	Jane Smith	
begin infusion of blood product	Sun Jan 03, 2016	12:11:15	Jane Smith	
perform clinical evaluation	Sun Jan 03, 2016	12:11:29	Jane Smith	
obtain vital signs	Sun Jan 03, 2016	12:11:51	Jane Smith	
document vital signs	Sun Jan 03, 2016	12:13:09	Jane Smith	HR: 85 bpm, BP: 119/90, TEMP: 98.6, SpO2%: 95
perform clinical evaluation	Sun Jan 03, 2016	12:13:17	Jane Smith	
obtain vital signs	Sun Jan 03, 2016	12:13:21	Jane Smith	
document vital signs	Sun Jan 03, 2016	12:13:22	Jane Smith	
record infusion info	Sun Jan 03, 2016	12:13:24	Jane Smith	
discard infusion materials	Sun Jan 03, 2016	12:13:26	Jane Smith	
perform follow through check	Sun Jan 03, 2016	12:13:28	Jane Smith	

7.6.2. Current generated post-procedure narrative

Carry Out Physician Order For Transfusion

The procedure was started at 12:10:15 on Sun Jan 03, 2016 for patient *John Doe*, ID: 12345678. The clinician involved was *Jane Smith*. At that time *John Doe*'s vital signs were: HR: 85 bpm; BP: 120/91; TEMP: 98.5 degrees.

The activity contact lab for availability of type and screen was completed at 12:10:22 on Sun Jan 03, 2016.

The activity prepare documentation for blood pick up was completed at 12:10:25 on Sun Jan 03, 2016.

The activity pick up blood from blood bank was completed at 12:10:27 on Sun Jan 03, 2016.

The activity review patient history was completed at 12:10:30 on Sun Jan 03, 2016.

The activity perform clinical evaluation was completed at 12:10:51 on Sun Jan 03, 2016.

The activity obtain vital signs was completed at 12:10:52 on Sun Jan 03, 2016.

The activity document vital signs was completed at 12:10:54 on Sun Jan 03, 2016.

The activity confirm presence of IV catheter was completed at 12:10:57 on Sun Jan 03, 2016.

The activity gather infusion materials was completed at 12:11:00 on Sun Jan 03, 2016.

The activity verify patient ID to ID band was completed at 12:11:03 on Sun Jan 03, 2016.

The activity verify blood product information was completed at 12:11:07 on Sun Jan 03, 2016.

The activity begin infusion of blood product was completed at 12:11:15 on Sun Jan 03, 2016.

The activity perform clinical evaluation was completed at 12:11:29 on Sun Jan 03, 2016.

The activity obtain vital signs was completed at 12:11:51 on Sun Jan 03, 2016.

The activity document vital signs was completed at 12:13:09 on Sun Jan 03, 2016.

The activity perform clinical evaluation was completed at 12:13:17 on Sun Jan 03, 2016.

The activity obtain vital signs was completed at 12:13:21 on Sun Jan 03, 2016.

The activity document vital signs was completed at 12:13:22 on Sun Jan 03, 2016.

The activity record infusion info was completed at 12:13:24 on Sun Jan 03, 2016.

The activity discard infusion materials was completed at 12:13:26 on Sun Jan 03, 2016.

The activity perform follow through check was completed at 12:13:28 on Sun Jan 03, 2016.

7.6.3. Mock up with vitals and without Clinician column

Carry Out Physician Order For Transfusion

Last updated: Mon Oct 12 12:06:30 EDT 2015; Created: Mon Oct 12 12:04:17 EDT 2015

Client Name: John Doe					MRN#: 12345678		
Activity	Date	Time	HR	BP	Temp	SpO2	Comments
contact lab for availability of type and screen	Mon Oct 12, 2015	12:04:22					
prepare documentation for blood pick up	Mon Oct 12, 2015	12:04:25					
pick up blood from blood bank	Mon Oct 12, 2015	12:04:27					
review patient history	Mon Oct 12, 2015	12:05:04					
perform clinical evaluation	Mon Oct 12, 2015	12:05:47	85	120/91	98	96	
obtain vital signs	Mon Oct 12, 2015	12:05:50					
document vital signs	Mon Oct 12, 2015	12:05:53	87	121/93	98	95	
confirm presence of IV catheter	Mon Oct 12, 2015	12:06:03					
gather infusion materials	Mon Oct 12, 2015	12:06:05					
verify patient ID to ID band	Mon Oct 12, 2015	12:06:07					
verify blood product information	Mon Oct 12, 2015	12:06:08					
begin infusion of blood product	Mon Oct 12, 2015	12:06:10					
perform clinical evaluation	Mon Oct 12, 2015	12:06:12					
obtain vital signs	Mon Oct 12, 2015	12:06:14					Patient's breathing is clear.
document vital signs	Mon Oct 12, 2015	12:06:16	91	119/90	98.6	95	
perform clinical evaluation	Mon Oct 12, 2015	12:06:18					
obtain vital signs	Mon Oct 12, 2015	12:06:19					
document vital signs	Mon Oct 12, 2015	12:06:22					
perform clinical evaluation	Mon Oct 12, 2015	12:06:23					
obtain vital signs	Mon Oct 12, 2015	12:06:25					
document vital signs	Mon Oct 12, 2015	12:06:27	90	119/91	98.5	96	
record infusion info	Mon Oct 12, 2015	12:06:28					
discard infusion materials	Mon Oct 12, 2015	12:06:29					

7.6.4. Only “Important” Steps

Carry Out Physician Order For Transfusion

Last updated: Mon Oct 12 12:06:30 EDT 2015; Created: Mon Oct 12 12:04:58 EDT 2015

Client Name: John Doe					MRN#: 12345678
Activity	HR	BP	Temp	SpO2	Comments
perform clinical evaluation	85	120/91	98	96	
document vital signs	87	121/93	98	95	
perform clinical evaluation					Patient's breathing is clear.
document vital signs	91	119/90	98.6	95	
perform clinical evaluation					Patient's breathing is clear.
document vital signs	90	119/91	98.5	96	

7.6.5. Mock up with all columns

Carry Out Physician Order For Transfusion

Last updated: Mon Oct 12 12:06:30 EDT 2015; Created: Mon Oct 12 12:04:17 EDT 2015

Client Name: John Doe					MRN#: 12345678			
Activity	Date	Time	HR	BP	Temp	SpO2	Clinician	Comments
contact lab for availability of type and screen	Mon Oct 12, 2015	12:04:22					Jane Smith	
prepare documentation for blood pick up	Mon Oct 12, 2015	12:04:25					Jane Smith	
pick up blood from blood bank	Mon Oct 12, 2015	12:04:27					Jane Smith	
review patient history	Mon Oct 12, 2015	12:05:04					Jane Smith	
perform clinical evaluation	Mon Oct 12, 2015	12:05:47	85	120/91	98	96	Jane Smith	
obtain vital signs	Mon Oct 12, 2015	12:05:50					Jane Smith	
document vital signs	Mon Oct 12, 2015	12:05:53	87	121/93	98	95	Jane Smith	
confirm presence of IV catheter	Mon Oct 12, 2015	12:06:03					Jane Smith	
gather infusion materials	Mon Oct 12, 2015	12:06:05					Jane Smith	
verify patient ID to ID band	Mon Oct 12, 2015	12:06:07					Jane Smith	
verify blood product information	Mon Oct 12, 2015	12:06:08					Jane Smith	
begin infusion of blood product	Mon Oct 12, 2015	12:06:10					Jane Smith	
perform clinical evaluation	Mon Oct 12, 2015	12:06:12					Jane Smith	
obtain vital signs	Mon Oct 12, 2015	12:06:14					Jane Smith	Patient's breathing is clear.
document vital signs	Mon Oct 12, 2015	12:06:16	91	119/90	98.6	95	Jane Smith	
perform clinical evaluation	Mon Oct 12, 2015	12:06:18					Jane Smith	
obtain vital signs	Mon Oct 12, 2015	12:06:19					Jane Smith	
document vital signs	Mon Oct 12, 2015	12:06:22					Jane Smith	
perform clinical evaluation	Mon Oct 12, 2015	12:06:23					Jane Smith	
obtain vital signs	Mon Oct 12, 2015	12:06:25					Jane Smith	
document vital signs	Mon Oct 12, 2015	12:06:27	90	119/21	98.5	96	Jane Smith	
record infusion info	Mon Oct 12, 2015	12:06:28					Jane Smith	
discard infusion materials	Mon Oct 12, 2015	12:06:29					Jane Smith	

7.6.6. Mock up without date, time, or clinician columns

Carry Out Physician Order For Transfusion

Last updated: Mon Oct 12 12:06:30 EDT 2015; Created: Mon Oct 12 12:04:58 EDT 2015

Client Name: John Doe					MRN#: 12345678
Activity	HR	BP	Temp	SpO2	Comments
contact lab for availability of type and screen					
prepare documentation for blood pick up					
pick up blood from blood bank					
review patient history					
perform clinical evaluation	85	120/91	98	96	
obtain vital signs					Patient's breathing is clear.
document vital signs	87	121/93	98	95	
confirm presence of IV catheter					
gather infusion materials					
verify patient ID to ID band					
verify blood product information					
begin infusion of blood product					
perform clinical evaluation					
obtain vital signs					
document vital signs	91	119/90	98.6	95	
perform clinical evaluation					
obtain vital signs					
document vital signs					
perform clinical evaluation					
obtain vital signs					
document vital signs	90	119/91	98.5	96	
record infusion info					
discard infusion materials					

7.6.7. Mock up for better structured narrative format

Carry Out Physician Order for Transfusion

The procedure was started at 10:47am on June 10th, 2015 for patient *John Doe*, ID#12345678. The clinician involved was nurse *Jane Smith*. At that time John Doe's vital signs were: HR: 85 bpm; BP: 120/91; TEMP: 98.5 degrees.

▼ The activity **check for existence of type and screen** was started at 10:51am.

Jane Smith completed the activity **contact lab for availability of type and screen** at 10:54am.

Jane Smith completed the activity **prepare documentation for blood pick up** at 11:02am.

Jane Smith completed the activity **pick up blood from blood bank** at 11:14am.

▼ The activity **perform pre-infusion work** was started at 11:17am.

Jane Smith completed the activity **review patient history** at 11:20am.

▼ The activity **evaluate patient clinically** was started at 11:20am.

Jane Smith completed the activity **perform clinical evaluation** at 11:26am.

▼ The activity **check vital signs** was started at 11:26am.

Jane Smith completed the activity **obtain vital signs** at 11:27am.

Jane Smith completed the activity **document vital signs** at 11:27am. At that time John Doe's vital signs were: HR: 105 bpm; BP: 125/91; TEMP: 98.5 degrees.

Jane Smith completed the activity **confirm presence of IV catheter** at 11:30am.

Jane Smith completed the activity **gather infusion materials** at 11:34am.

Jane Smith completed the activity **verify patient ID to ID band** at 11:37am.

Jane Smith completed the activity **verify blood product information** at 11:38am.

▼ The activity **infuse unit of blood product** started at 11:42am.

Jane Smith completed the activity **begin infusion of blood product** at 11:42am.

▼ The activity **evaluate patient clinically** was started at 11:45am.

Jane Smith completed the activity **perform clinical evaluation** at 11:53am. Note: Transfusion going well. No reaction suspected.

▼ The activity **check vital signs** was started at 12:00pm.

Jane Smith completed the activity **obtain vital signs** at 12:05pm.

Jane Smith completed the activity **document vital signs** at 12:05pm. At that time John Doe's vital signs were: HR: 95 bpm; BP: 120/95; TEMP: 98.5 degrees.

▼ The activity **perform post-infusion work** started at 12:08pm.

▼ The activity **evaluate patient clinically** was started at 12:08pm.

Jane Smith completed the activity **perform clinical evaluation** at 12:11pm.

▼ The activity **check vital signs** was started at 12:11pm.

Jane Smith completed the activity **obtain vital signs** at 12:18pm.

7.7. XML example

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<DocInternalForm>
  <intro>
    <date>Sun Jan 03, 2016</date>
    <time>12:10:15</time>
    <patient>John Doe</patient>
    <patientID>12345678</patientID>
    <clinician>Jane Smith</clinician>
  </intro>
  <title>carry out physician order for transfusion</title>
  <created>Sun Jan 03 12:10:17 EST 2016</created>
  <modified>Sun Jan 03 12:13:28 EST 2016</modified>
  <step>
    <stepname>contact lab for availability of type and screen</stepname>
    <state>completed</state>
    <date>Sun Jan 03, 2016</date>
    <time>12:10:22</time>
    <clinician>Jane Smith</clinician>
    <comments/>
  </step>
  <step>
    <stepname>prepare documentation for blood pick up</stepname>
    <state>completed</state>
    <date>Sun Jan 03, 2016</date>
    <time>12:10:25</time>
    <clinician>Jane Smith</clinician>
    <comments/>
  </step>
  <step>
    <stepname>pick up blood from blood bank</stepname>
    <state>completed</state>
    <date>Sun Jan 03, 2016</date>
    <time>12:10:27</time>
    <clinician>Jane Smith</clinician>
    <comments/>
  </step>
  <step>
    <stepname>review patient history</stepname>
    <state>completed</state>
    <date>Sun Jan 03, 2016</date>
    <time>12:10:30</time>
    <clinician>Jane Smith</clinician>
    <comments/>
  </step>
  <step>
    <stepname>perform clinical evaluation</stepname>
    <state>completed</state>
    <date>Sun Jan 03, 2016</date>
    <time>12:10:51</time>
    <clinician>Jane Smith</clinician>
    <comments>Patient's breathing is clear.</comments>
  </step>
  <step>
    <stepname>obtain vital signs</stepname>
    <state>completed</state>
    <date>Sun Jan 03, 2016</date>
    <time>12:10:52</time>
    <clinician>Jane Smith</clinician>
    <comments/>
  </step>
  <step>
    <stepname>document vital signs</stepname>
```

```

        <state>completed</state>
        <date>Sun Jan 03, 2016</date>
        <time>12:10:54</time>
        <clinician>Jane Smith</clinician>
        <comments/>
    </step>
    <step>
        <stepname>confirm presence of IV catheter</stepname>
        <state>completed</state>
        <date>Sun Jan 03, 2016</date>
        <time>12:10:57</time>
        <clinician>Jane Smith</clinician>
        <comments/>
    </step>
    <step>
        <stepname>gather infusion materials</stepname>
        <state>completed</state>
        <date>Sun Jan 03, 2016</date>
        <time>12:11:00</time>
        <clinician>Jane Smith</clinician>
        <comments/>
    </step>
    <step>
        <stepname>verify patient ID to ID band</stepname>
        <state>completed</state>
        <date>Sun Jan 03, 2016</date>
        <time>12:11:03</time>
        <clinician>Jane Smith</clinician>
        <comments/>
    </step>
    <step>
        <stepname>verify blood product information</stepname>
        <state>completed</state>
        <date>Sun Jan 03, 2016</date>
        <time>12:11:07</time>
        <clinician>Jane Smith</clinician>
        <comments/>
    </step>
    <step>
        <stepname>begin infusion of blood product</stepname>
        <state>completed</state>
        <date>Sun Jan 03, 2016</date>
        <time>12:11:15</time>
        <clinician>Jane Smith</clinician>
        <comments/>
    </step>
    <step>
        <stepname>perform clinical evaluation</stepname>
        <state>completed</state>
        <date>Sun Jan 03, 2016</date>
        <time>12:11:29</time>
        <clinician>Jane Smith</clinician>
        <comments/>
    </step>
    <step>
        <stepname>obtain vital signs</stepname>
        <state>completed</state>
        <date>Sun Jan 03, 2016</date>
        <time>12:11:51</time>
        <clinician>Jane Smith</clinician>
        <comments/>
    </step>
    <step>
        <stepname>document vital signs</stepname>
        <state>completed</state>

```

```

        <date>Sun Jan 03, 2016</date>
        <time>12:13:09</time>
        <clinician>Jane Smith</clinician>
        <comments>HR: 85 bpm, BP: 119/90, TEMP: 98.6, SpO2%: 95</comments>
    </step>
    <step>
        <stepname>perform clinical evaluation</stepname>
        <state>completed</state>
        <date>Sun Jan 03, 2016</date>
        <time>12:13:17</time>
        <clinician>Jane Smith</clinician>
        <comments/>
    </step>
    <step>
        <stepname>obtain vital signs</stepname>
        <state>completed</state>
        <date>Sun Jan 03, 2016</date>
        <time>12:13:21</time>
        <clinician>Jane Smith</clinician>
        <comments/>
    </step>
    <step>
        <stepname>document vital signs</stepname>
        <state>completed</state>
        <date>Sun Jan 03, 2016</date>
        <time>12:13:22</time>
        <clinician>Jane Smith</clinician>
        <comments/>
    </step>
    <step>
        <stepname>record infusion info</stepname>
        <state>completed</state>
        <date>Sun Jan 03, 2016</date>
        <time>12:13:24</time>
        <clinician>Jane Smith</clinician>
        <comments/>
    </step>
    <step>
        <stepname>discard infusion materials</stepname>
        <state>completed</state>
        <date>Sun Jan 03, 2016</date>
        <time>12:13:26</time>
        <clinician>Jane Smith</clinician>
        <comments/>
    </step>
    <step>
        <stepname>perform follow through check</stepname>
        <state>completed</state>
        <date>Sun Jan 03, 2016</date>
        <time>12:13:28</time>
        <clinician>Jane Smith</clinician>
        <comments/>
    </step>
</DocInternalForm>

```