Generating Natural-language Process Descriptions from Formal Process Models

Stefan C. Christov[1], Tiffany Y. Chao[1], and Lori A. Clarke[1]

Dept. of Computer Science, University of Massachusetts, Amherst, MA 01003
{christov, clarke}@cs.umass.edu, tiffany.y.chao@boeing.com

Abstract. Process models are often used to support the understanding and analysis of complex systems. The accuracy of such process models usually requires that various stakeholders review, evaluate, correct, and propose improvements to these models. Some stakeholders, however, may not have the skills to understand the process models except at a relatively superficial level. To address this issue, we have developed an approach for automatically generating natural-language process descriptions based on formal process models. Unlike existing electronic process guides, these process descriptions are generated completely automatically and can describe complex process features, such as exception handling, concurrency, and non-deterministic choice. The generated process descriptions have been well-received by domain experts from several fields and have also proven useful to process programmers. We describe our template-based approach, the resulting hypertext representation, and report on a user study that compares the understandability of the generated textual description and a diagrammatic process representation.

Keywords: natural-language process description, electronic process guide, process modeling, continuous process improvement

1 Introduction

Process models are often used to describe the interaction among humans, software systems, and hardware components [8, 9, 11, 20]. For human-intensive systems, such as life-critical medical procedures, search and rescue operations, and command and control management, humans require extensive expertise and play a critical role in the success of the overall system mission [10, 14, 16]. Thus, the process models for such systems must accurately represent the important roles that such humans play. To develop accurate models of a human-intensive process, it is usually important that the various stakeholders carefully review, evaluate, correct, and propose improvements to these models. Some stakeholders (e.g., domain experts, process performers, user interface designers, even some programmers), however, might not be experts in process modeling. Consequently, such stakeholders may not have the skills to understand the process models except at a relatively superficial level. We have seen this problem in our own work on modeling medical procedures. Medical professionals may be able to point out glaring misrepresentations, but are not sufficiently versed in modeling to fully understand the implications, for example, of complex control flow such as the handling of exceptional situations and concurrent execution.

To help stakeholders with diverse backgrounds understand complex processes, we have developed the Little-JIL Narrator. The Narrator takes a Little-JIL [6] process model, a set of templates of English phrases that define the different semantic elements of the process modeling language, and customization rules, and weaves together a textual, hyper-linked, description of the process model. In addition to a natural-language description of the process model, this textual representation includes a table of contents and an index of process steps. The generated narrative has been well received by the non-computing domain experts with whom we have been working. Surprisingly, the computer scientists involved in these projects have also found it to be a useful representation for reviewing the evolving process models. A user-study we performed indicates that the ability of the narrative to support process understanding by computer science students is comparable to that of an existing diagrammatic process representation. Moreover, the narrative seems to be a useful training guide to help novices learn about a complex process.

Requirements and design documents often include a natural-language description of a system. There is often a mismatch, however, between these natural-language descriptions and a formal description of the same process [8]. The natural-language descriptions are often imprecise and incomplete, and quickly become out-of-sync with evolving formal descriptions. The generated narrative created by our prototype tool, however, is as precise and complete as the Little-JIL process model from which it is derived and can easily be rederived whenever the process model is changed. On the other hand, it is also a more detailed and verbose representation than most human-created natural-language descriptions. To overcome this awkwardness, customization rules can be used to selectively remove some of the details and to control the level of explanation.

In this paper, we describe the Little-JIL Narrator. The next section discusses related work. Section 3 presents a simplified Little-JIL process model based on a real-world medical process and then provides and explains the generated narrative. Section 4 then provides an overview of the Narrator design. Section 5 describes a user study we performed to evaluate the Narrator, our experience with using the Narrator with real-world processes and some issues that were encountered. The final section discusses future research directions and concludes the paper.

2 Related Work

The importance of describing processes to various stakeholders, perhaps from different backgrounds, is well-established. In some early efforts to accomplish this, organizations created paper-based process guides or manuals that describe, largely in natural language, the process of interest. It has been observed, however, that such paper-based guides are difficult to navigate due to their inherently linear structure and significant size, time-consuming to develop, hard to keep in sync with the evolving process (or its formal description), and nearly impossible to customize. Thus, such paper-based guides are not very effective and rarely created or used [13].

To alleviate some of the disadvantages of paper-based process guides, organizations have resorted to electronic process guides (EPGs), which usually contain hyperlinks that facilitate navigation. Using hyperlinks addresses to some extent the navigation problems with paper-based process guides. Manually creating and maintaining EPGs remains a large problem. For example, [5] reports that during the maintenance of a V-Modell guide, "when changing the glossary structure to tool-tip style, some 2000 links had to be updated." As a result, some tools have been proposed to automatically generate EPGs, such as Adonis [1], ARIS [2], Eclipse Process Framework (EPF) Composer [12], and Spearmint [4]. Such tools use a process model as input to the generator that automatically creates a hyperlinked EPG based on that process model. The process model is usually captured in some notation that supports constructs such as activities and their hierarchical decomposition, artifacts and resources, roles, and the relationships between these constructs (e.g., which role is responsible for which activity, what resources are needed to perform an activity, and what artifacts are produced by an activity). Process engineers can then associate detailed natural-language descriptions with these components of a process model.

Once a process model is in place, an EPG generator can use it to create a hyperlinked document describing the process. EPF Composer and Spearmint, for instance, create a document that has a section showing the decomposition of the process activities (as a summary view of the process) and another section with a detailed natural-language description of the selected activity/artifact/role. These sections, however, contain little indication of the order, or the flow of control among the process activities. The section that provides a summary view of the process, for example, shows the hierarchical decomposition of the process activities, but not the order in which they need to be executed. The detailed, natural-language description of the selected activity may contain some information about control flow, but this information is included in a non-systematic way as it is up to the person who writes the description to decide how much control flow information to include.

Another concern with the natural-language process descriptions generated by the tools mentioned above is related to the lack of semantic richness of the process notations used to create the process models that are in turn the basis for the generation of EPGs. Such process notations often lack support for complex exception handling behavior, concurrency and synchronization mechanisms, or even mechanisms for simpler constructs such as looping. For example, [17] reports that the EPF Composer was not able to represent parts of a software development process because EPF Composer could not adequately model the looping at the activity level.

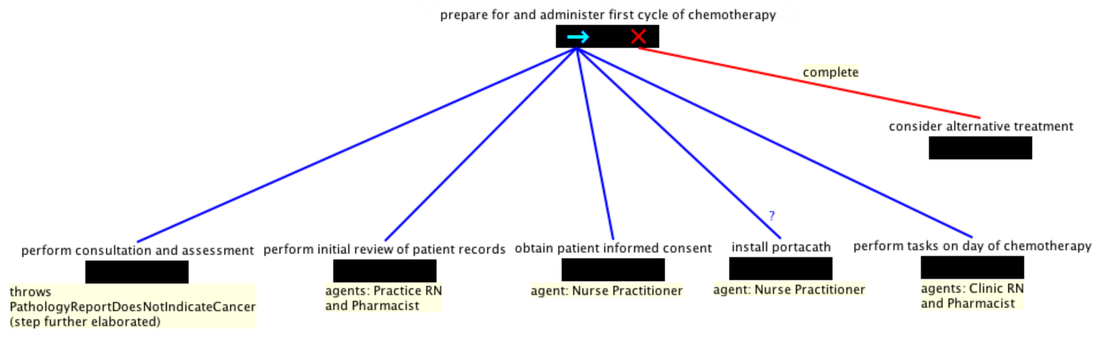

prepare for and administer first cycle of chemotherapy
complete
consider alternative treatment
?
perform consultation and assessment
throws PathologyReportDoesNotIndicateCancer (step further elaborated)
perform initial review of patient records
agents: Practice RN and Pharmacist
obtain patient informed consent
agent: Nurse Practitioner
install portacath
agent: Nurse Practitioner
perform tasks on day of chemotherapy
agents: Clinic RN and Pharmacist

Fig. 1. An example chemotherapy process.

The hyperlinked, natural-language description generated by the Little-JIL Narrator described here contains information about resources, artifacts and agents, similar to the EPGs generated by the tools discussed above, but also includes a detailed description of the control flow. The Little-JIL language provides complex control flow constructs, such as support for exceptional behavior, concurrency, synchronization, and recursion, thereby making the generation of the narrative more useful, but more challenging. Finally, the full description is automatically generated from the process model, eliminating the issues related to manual creation and maintenance mentioned above.

3 Example

This section presents a part of a Little-JIL process model for a chemotherapy process, and then shows and explains the natural-language description generated from this process model using the Narrator. Chemotherapy is a very high-risk medical procedure since incorrect administration of chemotherapy medications can be fatal for patients or cause them irreversible harm. As a result, complex and sophisticated processes are in place to ensure that proper checks are done and that the risk of harming patients is minimized.

The main building block of a Little-JIL process model is the step, which corresponds to an activity performed by human or automated agents and is iconically represented by a black rectangle. Little-JIL process models contain a hierarchical decomposition of steps.

Prepare for and administer first cycle of chemotherapy is a *sequential step* (indicated by the arrow in the step rectangle), which means that its substeps need to be performed in left to right order. The substep *install portacath* (a portacath is a device for intravenous access) is an optional step (indicated by the question mark above the step). This means that it is up to the step's agent to decide whether to perform that step. Since not all chemotherapy regimens require the installation of a portacath, the agent may decide not to execute this step for some patients.

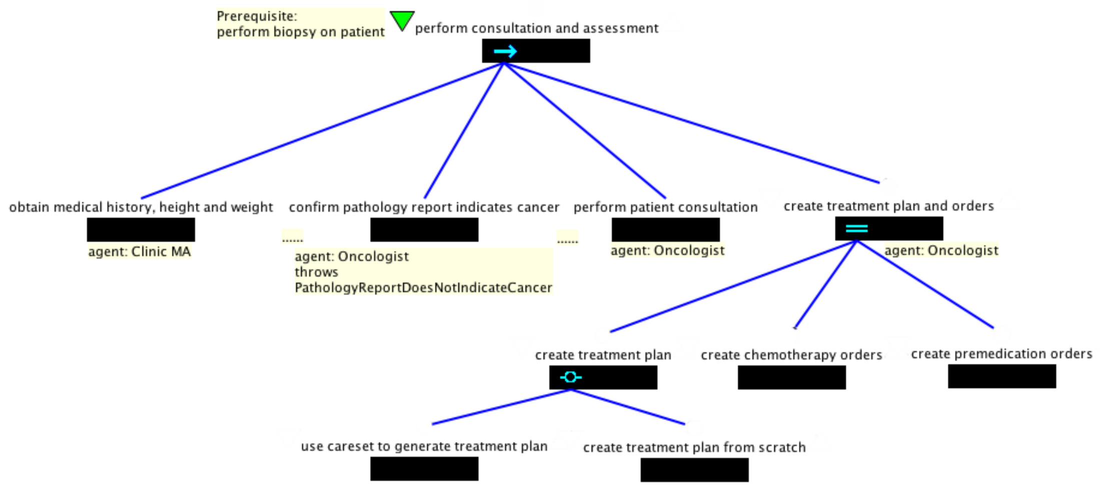

Prerequisite:
perform biopsy on patient
perform consultation and assessment
obtain medical history, height and weight
agent: Clinic MA
confirm pathology report indicates cancer
agent: Oncologist
throws
PathologyReportDoesNotIndicateCancer
perform patient consultation
agent: Oncologist
create treatment plan and orders
agent: Oncologist
create treatment plan
create chemotherapy orders
create premedication orders
use careset to generate treatment plan
create treatment plan from scratch

Fig. 2. Elaboration of *perform consultation and assessment.*

The substep *perform consultation and assessment* is further elaborated in Figure 2. This step has a prerequisite (indicated by the filled triangle to the left of the step rectangle) to represent the fact that before the patient consultation and assessment can start, the patient biopsy must have been performed.

The substep *create treatment plan and orders* is a *parallel step* (indicated by the equal sign on the step rectangle), representing the fact that the oncologist can choose to create the treatment plan, the chemotherapy orders, and the premedication orders in any order, including simultaneously. Furthermore, the oncologist can choose between two alternative options for how to create the treatment plan—use a standardized treatment plan (a careset) or create the treatment plan from scratch, expressed by making *create treatment plan* a choice step (represented by a circle over a line in the step rectangle).

While performing the step *confirm pathology report indicates cancer* in Figure 2, the oncologist may discover that the pathology report does not indicate cancer. This unusual circumstance is modeled by using Little-JIL's exception handling mechanisms. The step *confirm pathology report indicates cancer* in Figure 2 throws the exception *PathologyReportDoesNotIndicateCancer*. The exception propagates up the step tree until a matching exception handler is found. An exception handler is itself a regular step and, as any step, it can be decomposed to any desired level of detail. Once the handler step *consider alternative treatment* (in Figure 1) is performed, Little-JIL's continuation action semantics are used to indicate how the process should continue. In this example, the continuation action is *complete* (indicated by the label on the edge connecting the handler and its parent step), which means that the parent step *prepare for and administer first cycle of chemotherapy* is considered completed.

In Little-JIL, each step is performed by humans or automated agents. For example, the step *perform initial review of patient records* in Figure 1 is performed by a Practice Registered Nurse (RN) and a Pharmacist. To reduce visual clutter, we did not annotate every step in Figures 1 and 2 with agents.

The rich semantics of Little-JIL allow the model of the chemotherapy process to capture many aspects of the real-world process concisely. While these features of Little-JIL and other similar process notations make process models attractive to system engineers and other people trained in these notations, people without such training may prefer textual rather than a diagrammatical representations. Figure 3 shows the automatically generated textual narrative created by the Narrator from the process model shown in Figures 1 and 2.

The narrative consists of two main parts: a table of contents on the left and the descriptive part on the right. The table of contents lists the names of the steps from the process model and uses the same icons used in the diagramatic representation to represent the step kinds (e.g., sequential, parallel, choice step). The parent/child relationship from the Little-JIL process model tree is captured by the indentation in the table of contents. The table of contents also shows the exception handlers. For example, the step *consider alternative treatment* is an exception handler, indicated by the icon on the left, which shows the continuation action (in this case *complete*) that needs to be taken after the handling of an exception is done. Each step in the table of contents is also a hyperlink and clicking on it will bring up a more detailed description of that step in the descriptive part of the narrative.

The descriptive part of the narrative (the right part of Figure 3) contains a section for each step in the process model. This step section consists of several subsections that present various attributes of the given step, such as name, pre/post requisites, substep sequencing information, exceptions, and required resources. For instance, the step section for *Perform Consultation And Assessment* contains a subsection about the step's prerequisite, a subsection about the step's substeps and the order in which they have to be executed, and a subsection about an exception that the step throws and how that exception is handled.

Unlike the diagrammatic representation of the process model in Figures 1 and 2 where familiarity with the notation semantics is assumed, the descriptive part provides sentences to explain the process. For example, the step section for *Perform Consultation And Assessment* in Figure 3 explains what it means for the step to be a sequential step, namely that its substeps need to be completed in the listed order.

The descriptive part of the narrative also uses hyperlinks to facilitate navigation. When the substep *perform tasks on day of chemotherapy* (in the step section for *Prepare For And Administer First Cycle of Chemotherapy*) is clicked, for example, its step section will be displayed and the user will see the detailed information associated with that step. Another facility to help with navigation is an alphabetized index of step names (shown in the bottom right part of Figure 3), where each step is represented as a hyperlink that can be clicked to display the description for that step. The index can be opened at any time by

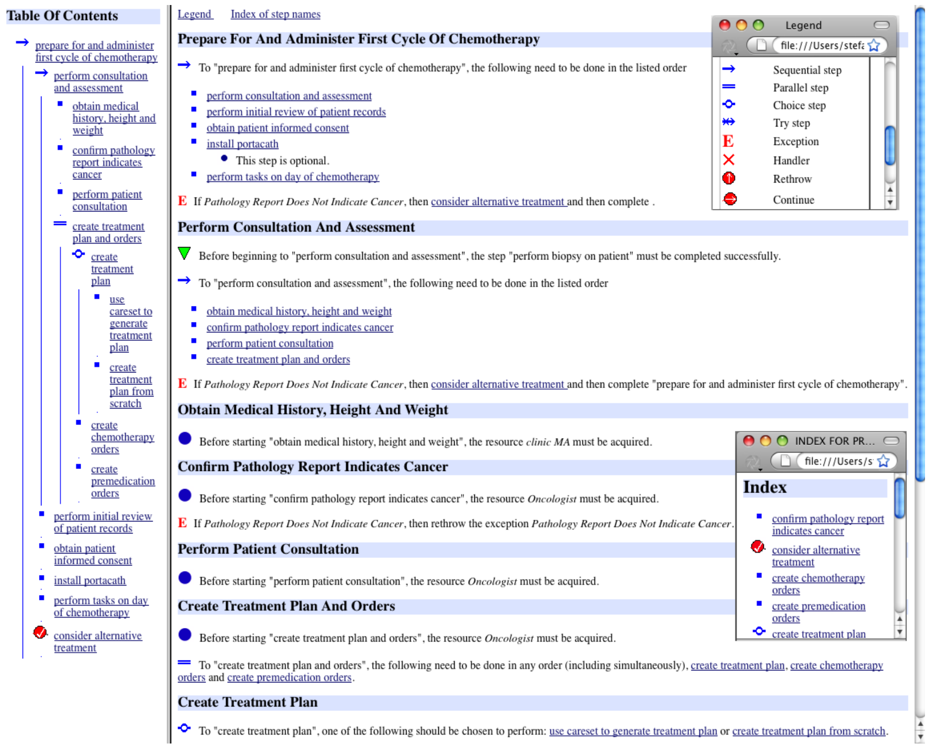


Table Of Contents

- prepare for and administer first cycle of chemotherapy
  - perform consultation and assessment
    - obtain medical history, height and weight
    - confirm pathology report indicates cancer
    - perform patient consultation
    - create treatment plan and orders
      - create treatment plan
        - use careset to generate treatment plan
        - create treatment plan from scratch
      - create chemotherapy orders
      - create premedication orders
  - perform initial review of patient records
  - obtain patient informed consent
  - install portacath
  - perform tasks on day of chemotherapy
  - consider alternative treatment

Legend Index of step names

Prepare For And Administer First Cycle Of Chemotherapy

To "prepare for and administer first cycle of chemotherapy", the following need to be done in the listed order

- perform consultation and assessment
- perform initial review of patient records
- obtain patient informed consent
- install portacath
  - This step is optional.
- perform tasks on day of chemotherapy

E If *Pathology Report Does Not Indicate Cancer*, then consider alternative treatment and then complete .

Perform Consultation And Assessment

Before beginning to "perform consultation and assessment", the step "perform biopsy on patient" must be completed successfully.

To "perform consultation and assessment", the following need to be done in the listed order

- obtain medical history, height and weight
- confirm pathology report indicates cancer
- perform patient consultation
- create treatment plan and orders

E If *Pathology Report Does Not Indicate Cancer*, then consider alternative treatment and then complete "prepare for and administer first cycle of chemotherapy".

Obtain Medical History, Height And Weight

Before starting "obtain medical history, height and weight", the resource *clinic MA* must be acquired.

Confirm Pathology Report Indicates Cancer

Before starting "confirm pathology report indicates cancer", the resource *Oncologist* must be acquired.

E If *Pathology Report Does Not Indicate Cancer*, then rethrow the exception *Pathology Report Does Not Indicate Cancer*.

Perform Patient Consultation

Before starting "perform patient consultation", the resource *Oncologist* must be acquired.

Create Treatment Plan And Orders

Before starting "create treatment plan and orders", the resource *Oncologist* must be acquired.

To "create treatment plan and orders", the following need to be done in any order (including simultaneously), create treatment plan, create chemotherapy orders and create premedication orders.

Create Treatment Plan

To "create treatment plan", one of the following should be chosen to perform: use careset to generate treatment plan or create treatment plan from scratch.

Legend

file:///Users/stefa

→	Sequential step
═	Parallel step
⌖	Choice step
↠	Try step
E	Exception
✕	Handler
●	Rethrow
⊖	Continue

INDEX FOR PR...

file:///Users/s

Index

- confirm pathology report indicates cancer
- consider alternative treatment
- create chemotherapy orders
- create premedication orders
- create treatment plan

Fig. 3. The automatically generated narrative for process model in Fig. 1 and 2.

clicking on *Index of step names* on the top of the main part of the narrative and closed when not needed.

The narrative uses the same icons as the diagrammatic view of the process model. Although these icons are not necessary to understand the narrative view, they might be helpful for users who would like to work with both views at the same time. They also provide some visual grouping of sentences based on the icon the sentences are associated with. The meaning of the icons can be seen in a legend (shown in the top right part of Figre 3), which can be opened and closed the same way as the index of step names.

4 Design Approach

Figure 4 shows the high-level architecture of the Narrator. The *Little-JIL Process Model*, the *Phrasing Templates*, and the *Customization Rules* are used by the *Narration Weaver* to produce the *Narrative Content*, which contains just the content and the structure without any formatting of the natural-language document to be generated. The *Formatting Weaver* then combines the *Narrative*

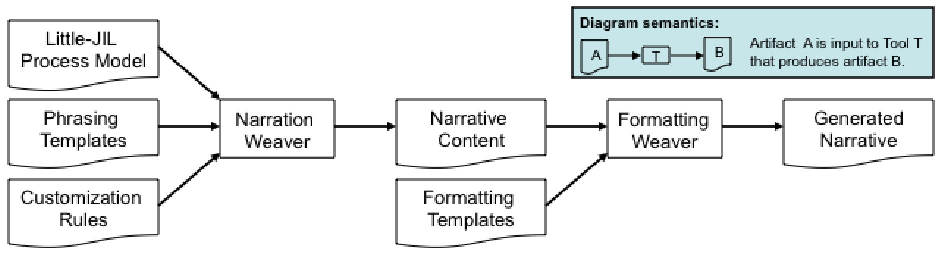


Fig. 4. Architecture of the Little-JIL Narrator.

Content together with the *Formatting Templates* to produce the final *Generated Narrative*.

Phrasing Templates. The Phrasing Templates are parameterized, natural-language phrases that correspond to the different semantic features of the Little-JIL process language (e.g., what it means for a step to be sequential), where the parameters represent information that is specific to a given process model. Figure 5 shows three example phrasing templates. The first phrasing template is used with sequential process steps to generate a sentence explaining the order of execution of their substeps.

The second phrasing template in Figure 5 is used to generate a sentence that explains what it means for a step to have a prerequisite. For instance, applying this phrasing template to the step *perform consultation and assessment* (in Figure 2) and its prerequisite *perform biopsy on patient* results in the sentence *Before beginning "perform consultation and assessment", the step "perform biopsy on patient" must be completed successfully.* This exact sentence can be seen in the step section for *Perform Consultation And Assessment* in Figure 3.

The third phrasing template in Figure 5 is used to generate a sentence explaining what it means for a step to throw an exception and how it is handled. For instance, applying this phrasing template to the step *perform consultation and assessment*, which can throw the exception *PathologyReportDoesNotIndicateCancer* (as shown in Figure 1), results in the sentence *If Pathology Report Does Not Indicate Cancer*, then consider alternative treatment and complete "prepare for and administer first cycle of chemotherapy." This sentence briefly captures what the exceptional event is, how it is dealt with (i.e., by executing the exception handler *consider alternative treatment*) and how normal process execution is resumed after the exception has been handled.

To [stepName], the following need to be done in the listed order [substepsList].
Before beginning to [stepName], the [activityName] [prerequisite] must be completed successfully.
If [exception], then [handler] and then complete [parentStepName].

Fig. 5. Example phrasing templates.

Customization Rules. The Customization Rules in the Narrator architecture in Figure 4 represent a set of user preferences to customize the content and the structure of the generated natural-language narrative. The Narrator supports the use of synonyms. For example, different words can be used to refer to a process activity. The parameter *[activityName]* in the second phrasing template in Figure 5 is a placeholder for such a synonym. The word "step" was used in place of *[activityName]* when this phrasing template was instantiated to describe the prerequisite of *Perform Consultation and Assessment* in Figure 3.

Another kind of customization supported by the Little-JIL Narrator deals with the ability to hide or show certain kinds of process information. For example, the user can select to hide or show sentences that present information about the resources in a process model. Before the narrative shown in Figure 3 was generated, the option to show resources was selected. Thus, resource information, such as the human agents responsible for executing the process steps, is included in the narrative. The user can choose to hide this information or, alternatively, the user can choose to include additional information about the resources and the artifacts used in the process. Choosing to include such additional information results in the narrative containing sentences such as *Successful completion of the step "perform consultation and assessment" should yield the chemo orders.* The Narrator also provides the flexibility to choose what kinds of process steps to associate certain information with. For example, the user can choose to show resource information only for leaf steps but not intermediate steps. This is sometimes useful as intermediate steps in Little-JIL are often used for coordination purposes, whereas the actual work performed by agents is modeled by leaf steps.

The Little-JIL Narrator also provides facilities to customize the sentence structure of the generated narrative. For instance, the user can define when the substeps of a step should be enumerated as a comma-separated phrase or when they should be shown as a list.

Formatting Templates. The Narrative Content artifact produced by the Narration Weaver (as shown in Figure 4) contains the raw content of the generated narrative, but does not have any formatting information. It is the job of the Formatting Templates to define the presentation style of the generated narrative. This design essentially follows the well-established recommendations from the web application domain to separate content from presentation.

For Figure 3, for example, the Formatting Templates were responsible for defining text font, text color, text size, text style (e.g., bold vs. non-bold), spacing information and background color for the table of contents, the main part of the narrative, the index and the legend. The Formatting Templates were also responsible for associating images (e.g, arrow, filled circle, check mark, etc.) with the different sections of the narrative and for the visualization (indentation and vertical lines that help keeping track of the hierarchical decomposition) of the table of contents. The set of Formatting Templates used by the Formatting Weaver in this example resulted in an HTML-based generated narrative. A different set of Formatting Templates could be used to produce a plain text (not hyperlinked) narrative or a narrative in some other document format.

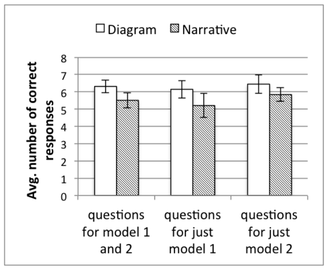
In terms of implementation, the Phrasing Templates, the Customization Rules, and the Narrative Content artifacts are currently XML documents following a schema we defined. The Narration Weaver is a Java system. The formatting templates are expressed as XSLT [3] templates and the Formatting Weaver is therefore an XSLT processor.

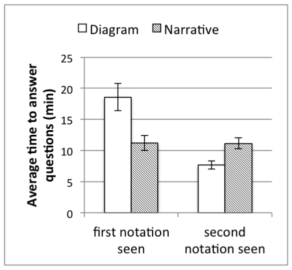
5 Evaluation and Discussion

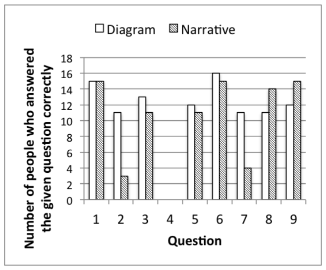
5.1 User study

Design. We conducted a user study to compare the understandablility of the generated narrative and the Little-JIL diagrammatic representation. We created two Little-JIL diagrams and their corresponding narratives. The two models were of equal complexity, in terms of their size and the language features they employed. In fact, one model was created by essentially "reshuffling" the steps in the other one. For each process model, we also created a questionnaire consisting of nine questions testing the understanding of the process. The two questionnaires were of equal complexity since they were almost identical except for variations in step names. We used colors for step names (e.g., *perform blue*) as opposed to names of real activities (e.g., *drive to work*) to not bias the results based on a subject's experience with a domain. We also included a final questionnaire with 5 open-ended questions asking for general feedback about the two process representations. The process descriptions and questionnaires are available at `www.cs.umass.edu/~christov/processNotationsStudy.html` .

Half of the subjects were presented with the narrative representation of process model 1 followed by the diagrammatic representation of process model 2; the other half of the subjects were presented with the diagrammatic representation of process model 1 followed by the narrative representation of process model 2. We presented the two process representations in different orders to the two groups of subjects, because even though the understanding of process model 1 should not have helped with understanding process model 2, the subjects might have improved their understanding of fundamental process concepts (such as various kinds of control flow, exception handling and artifact use). The subjects were assigned randomly to one of these two groups. Subjects were given a 10-15 minutes training in their first process model representations (the narrative or the diagrammatic representation). Then, the subjects were presented with a process model in that representation and with the questionnaire for that process model. After the subjects answered the questionnaire (there was no time limit imposed), that subjects were given a 10-15 minutes training in the other process model representation. Then, the subjects were presented with the process model for their second process represented in their second representation and with the questionnaire for that process model. After the subjects answered the second questionnaire (again, no time limit was imposed), the subjects were presented with the final questionnaire. 16 subjects participated in the study: 6







a b c

Table 1. Study results.

graduate and 9 undergraduate computer science students and one non-computer science undergraduate student. We excluded students who had experience with the Little-JIL diagrammatic or narrative representations.

Results. On average, the number of correct responses per subject taking into account the questions for both models was slightly higher when the subjects were using the diagrams—6.3 vs. 5.5, $p = 0.037$ (Table 1a). This table also shows a small increase in the average number of correctly answered questions for process model 2 for both representations. Since process model 2 was presented after process model 1, this might suggest that whichever representation was introduced to the subjects first had an influence on facilitating the understanding of the second representation.

The subjects took longer time on average to answer the questions about process model 1 when they were using the diagrams—18.57 minutes vs. 11.22 minutes with the narrative, $p = 0.007$ (Table 1b). This might suggest that the narrative is easier to learn and use, if a person does not have familiarity with a process modeling notation and/or process modeling concepts.

The time to answer the second set of equally difficult questions about process model 2 using the diagrams, however, decreased by more than half after the subjects had used the narrative to answer the questions about process model 1 ($p = 0.0001$). This might suggest that using the narrative to introduce process modeling concepts, and specifically concepts captured by the Little-JIL language, facilitates subsequent understanding of the Little-JIL diagrams.

Table 1c seems to indicate that the diagrams made questions 2, 3 and 7 easier to answer, whereas the narrative made questions 8 and 9 easier to answer[1]. Question 2 was about prerequisites, 3 about parallel execution and 7 about the number of times a step can be executed. Questions 8 and 9 were about artifacts. This might suggest the merit of both representations in terms of facilitating the understanding of different kinds of process information. The difference between

[1] Question 4 was excluded from our analysis because it covered the difficult concept of non-deterministic choice and the training we provided might not have been sufficient to answer this question. It was answered correctly by only one person when they were using the narrative.

the number of correct responses for questions 2 and 7 was statistically significant ($p = 0.002$ and $p = 0.004$) and for question 9 ($p = 0.083$) it was between the significance levels of $p = 0.1$ and $p = 0.05$.

6 subjects (37.5%) found the narrative easier to understand, the other 10 preferred the diagrams. The answers to the open-ended, qualitative questions also indicated that the order of step execution was easier to understand with the narrative, whereas the number of times a step needs to be performed was easier to understand with the diagrams. Several people mentioned that the narrative was easier to understand at first with little training, which seems to be supported by the results in Table 1b.

Threats to validity. All of the study subjects, except one, were computer science students and indicated that they have programming experience. We plan to repeat the user study with non-computer scientists to see if familiarity with programming might have influenced the results (e.g., familiarity with programming might have made the diagrams easier to understand).

Immediately before the subjects were given the questionnaires, the subjects received focused training about how each of the two process representations expresses process information that the subsequent questionnaires asked about. In a real-world situation, the amount of time between training and a subsequent use of one of the process representations might be much longer. Furthermore, the training might not necessarily cover the representation of some kinds of information (e.g., exception handling) that needs to be understood later on. Thus, users of the process representations might forget or might have never been trained how certain kinds of information are represented in a given notation. We expect that in situations like this, the narrative might be easier to understand since it explicitly explains process information in natural language and is thus more self-explanatory.

Due to time constraints on the study sessions, the two process models used were relatively small (19 steps each). Realistic process models, especially models of complex human-intensive processes, however, could be larger. It is not clear whether the results obtained in the study will apply to such larger models.

5.2 Experience

We have used the Narrator to generate natural-language descriptions of several large Little-JIL models of real-world processes from the medical and negotiations domains [8, 9, 15]. In particular, we have applied the tool to a chemotherapy, a blood transfusion, and an online dispute resolution (ODR) process model. These process models were developed in collaboration with domain experts as part of case studies evaluating the application of process modeling and formal analysis technology in support of continuous process improvement. The chemotherapy process model had 467 Little-JIL steps (207 of which were related to the handling of 59 exceptional situations); the blood transfusion process model had 248 Little-JIL steps (37 of which were related to the handling of 15 exceptional situations); and the ODR process model had 209 Little-JIL steps (108 of which were related to the handling of 19 exceptional situations).

Even though the Little-JIL Narrator is an early prototype, our experiences with it have been positive and promising. The medical professionals from whom we elicited the medical processes expressed satisfaction with the generated narrative. They liked the fact that the process was described in natural language that they could easily understand and, at the same time, the description was precise yet contained a significant level of detail. Furthermore, since the narrative is automatically generated from a Little-JIL process model, we were able to show the domain experts the most up-to-date natural-language description and discuss the latest process changes and additions with them. The ODR domain expert, however, preferred to look at the diagrammatic depiction of the Little-JIL ODR process model and have a process programmer explain the semantics of the Little-JIL iconography. This experience suggests that although a generated natural-language description of a process model is beneficial to certain domain experts, it is certainly not a replacement for the diagrammatic view and the two representations could complement each other in an EPG.

Besides being beneficial to domain experts, the generated narrative turned out to be beneficial to Little-JIL process programmers as well. Being able to look at the natural-language narrative helped reveal errors in the process model that were not immediately noticeable in the diagrammatic representation. For example, artifact flow is not easily visible in the diagrammatic representation and inspecting the generated narrative helped pinpoint omissions or unnecessary additions of artifacts. The ability to automatically and quickly (it takes about a second for a process model with several hundred steps) generate a narrative makes the Little-JIL Narrator a useful tool for debugging process models.

Perhaps one obvious disadvantage of the generated Narrative is its size. The natural-language descriptions of the non-trivial process models we worked with tended to be quite long and verbose. Unfortunately this is inevitable when trying to express precisely and completely in natural language all the information captured in a process model created in a semantically rich process language. The customization rules that the Little-JIL narrator supports were useful in addressing this issue, by allowing the user to selectively hide or show information and thus focus interest on selected aspects of the process model (e.g., control flow only, but no artifact flow).

The separation of concerns supported by the design of the Narrator allowed for easy modifications to the tool. In particular, it was easy to experiment with and change in the description of the semantics of the language, as only the pertinent phrasing templates had to be changed. Similarly, it was easy to modify the look-and-feel of the generated HTML document based on user feedback because only the formatting templates had to be changed.

5.3 Issues

To generate text automatically that reads naturally to humans requires that the process models be written according to some guidelines or conventions. To be able to plug step names from the Little-JIL process model directly into the phrasing templates, for example, we assume that step names start with a verb.

Our experience with the Little-JIL Narrator, and with process models in general, indicates that this assumption about step names is reasonable and not very limiting.

Another difficulty that we encountered was related to the combination of the semantic richness of Little-JIL and the need to *statically* determine certain kinds of information. For example, to perform a choice step, any of its substeps can be chosen, and if the chosen substep is completed successfully, then the parent choice step is completed successfully as well. If the selected chosen step fails, however, the agent can choose to perform any of the remaining substeps of the choice step. Thus, when generating the natural language for this situation, we simply do not specify exactly which of the remaining substeps needs to be performed next but indicate the set of choices.

Generating natural language that accurately describes artifact flow also proved to be challenging. Since Little-JIL is a scoped process modeling language, some artifacts pass through certain steps just to reach steps in different scopes, but such artifacts are not necessarily used or modified in the steps they pass through. For such artifacts, the current phrasing is that they *may* get used or modified. One possible approach for improving the precision of the generated description is to have the creator of the process model explicitly provide information about when an artifact is used and/or modified. Such annotations would also improve the accuracy of some of the analysis approaches that we also apply [7, 8, 18, 21].

Another challenge is representing the rich exception handling semantics of Little-JIL. As in many scoped (process) programming languages, when an exception is thrown, the handler may not be located in the immediate enclosing scope. Thus, there needs to be a search for the matching handler. When describing exception handling, the natural-language narrative needs to specify the appropriate handler as well as how the process returns to normal flow once an exception has been handled. This is complicated by the several exception-handling continuation semantics of Little-JIL, by the fact that the continuation action depends on the context of the handler (i.e., the kind of step, such as parallel or sequential step, to which it is attached), and by the fact that an exception handler can throw an exception itself. These complications lead to a large number of phrasing templates to describe exceptional flow (over 100) and to challenges in determining which text to generate statically. While we have created most of the necessary phrasing templates to describe Little-JIL's exception handling semantics, for the sake of simplicity, the current implementation of the Narrator supports the generation of natural language for just the most commonly used exception handling constructs.

6 Conclusion and Future Work

There are several directions for potential improvement of the Little-JIL Narrator approach. Linking the generated narrative to a glossary of terms, or ontology, seems to be a useful direction. Such a capability is already supported by some

EPG tools, such as Spearmint. A glossary may help novice process performers who are not familiar with all the domain terminology.

Another area of improvement is including additional customization rules. Although the customization rules currently supported by the Narrator are very helpful for selectively filtering content and displaying information of interest, more extensive tailoring could further improve the usability of the process descriptions [19]. For example, being able to present only the part of a process description related to a specific role would focus the description and may make it easier to navigate and understand that specific process role.

The index generated by the Narrator could also be improved. It currently contains only step names, but it may be useful to have indexes that include resource names, exceptions, and roles.

Our initial experience with the Little-JIL Narrator has been promising. We have applied the Narrator to several large models of real-world processes from the medical and negotiations domains; the generated narrative has been well-received by domain experts and it seems to be useful to process programmers for discovering errors in a process model. A user study in which the narrative was compared to the Little-JIL diagrammatic representation, indicates that the narrative provides similar level of process understanding as the diagrams and it seems to make certain kinds of process information (i.e., artifacts) easier to understand. Study subjects took less time to learn and use the narrative when it was introduced as a first process notation to them and, furthermore, introducing the narrative before the diagrams was associated with a significant decrease of the time needed to learn and start using the diagrammatic representation. A large fraction of the study subjects (37.5%) indicated that the narrative is easier to understand than the diagrammatic notation.

The user study results and our experience indicate that the narrative is not a substitute for a diagrammatic process representation, but the narrative seems to provide added value in terms of understanding process models. Thus, we believe the narrative could complement other representations in making process models accessible to various stakeholders with different background and training.

7 Acknowledgments

This material is based upon work supported by the National Science Foundation under Awards CCF-0820198, CCF-0905530 and IIS-0705772, and by a Gift from the Baystate Medical Center, Rays of Hope Foundation. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the NSF.

The authors gratefully acknowledge the contributions of Eric Raboin, who developed the first version of the Little-JIL Narrator and whose work contributed to the look-and-feel of the tool, and of Meagan Day, who made contributions to the phrasing templates. The authors also gratefully acknowledge George Avrunin, Heather Conboy, Elizabeth Henneman, Lee Osterweil, and Sandy Wise for the discussions and feedback on the Little-JIL Narrator.

References

1. Adonis, www.boc-group.com/at, www.boc-group.com/at
2. Aris, www.ids-scheer.de, www.ids-scheer.de
3. XSL transformations (XSLT) version 2.0, www.w3.org/tr/xslt20, www.w3.org/TR/xslt20
4. Becker-Kornstaedt, U., Hamann, D., Kempkens, R., Rösch, P., Verlage, M., Webby, R., Zettel, J.: Support for the process engineer: The Spearmint approach to software process definition and process guidance. In: Proceedings of the 11th International Conference on Advanced Information Systems Engineering. pp. 119–133. CAiSE '99, Springer-Verlag (1999), http://portal.acm.org/citation.cfm?id=646087.679892
5. Becker-Kornstaedt, U., Verlage, M.: The V-modell guide: Experience with a web-based approach for process support. In: Proceedings of Software Technology and Engineering Practice (STEP). pp. 161–168. IEEE Computer Society Press (1999)
6. Cass, A.G., Lerner, B.S., Stanley M. Sutton, J., McCall, E.K., Wise, A., Osterweil, L.J.: Little-JIL/Juliette: a process definition language and interpreter. In: ICSE '00: Proceedings of the 22nd International Conference on Software Engineering. pp. 754–757. ACM (2000)
7. Chen, B., Avrunin, G.S., Clarke, L.A., Osterweil, L.J.: Automatic fault tree derivation from Little-JIL process definitions. In: SPW/ProSim. LNCS, vol. 3966, pp. 150–158. Shanghai (May 2006)
8. Chen, B., Avrunin, G.S., Henneman, E.A., Clarke, L.A., Osterweil, L.J., Henneman, P.L.: Analyzing medical processes. In: ICSE '08: Proceedings of the 30th International Conference on Software Engineering. pp. 623–632. ACM (2008)
9. Christov, S., Chen, B., Avrunin, G.S., Clarke, L.A., Osterweil, L.J., Brown, D., Cassells, L., Mertens, W.: Formally defining medical processes. Methods of Information in Medicine. Special Topic on Model-Based Design of Trustworthy Health Information Systems 47(5), 392–398 (2008)
10. Clarke, L.A., Osterweil, L.J., Avrunin, G.S.: Supporting human-intensive systems. In: Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research. pp. 87–92. FoSER '10, ACM (2010), http://doi.acm.org/10.1145/1882362.1882381
11. Damas, C., Lambeau, B., Roucoux, F., van Lamsweerde, A.: Analyzing critical process models through behavior model synthesis. In: ICSE '09: Proceedings of the 2009 IEEE 31st International Conference on Software Engineering. pp. 441–451. IEEE Computer Society (2009)
12. Haumer, P.: Increasing development increasing development knowledge with EPFC. Eclipse Review pp. 26–33 (2006)
13. Kellner, M., Becker-Kornstaedt, U., Riddle, W., Tomal, J., Verlage, M.: Process guides: Effective guidance for process participants. In: Proeceedings of the International Conference on the Software Process. pp. 11–25 (1998)
14. Lee, J.D., See, K.A.: Trust in automation: Designing for appropriate reliance. Human Factors: The Journal of the Human Factors and Ergonomics Society 46, 50–80 (2004)
15. Osterweil, L.J., Clarke, L.A.: Supporting negotiation and dispute resolution with computing and communication technologies. In: Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research. pp. 269–272. FoSER '10, ACM (2010), http://doi.acm.org/10.1145/1882362.1882418

16. Parasuraman, R., Riley, V.: Humans and Automation: Use, Misuse, Disuse, Abuse. Human Factors: The Journal of the Human Factors and Ergonomics Society 39(2), 230–253 (June 1997), `http://dx.doi.org/10.1518/001872097778543886`
17. Phongpaibul, M., Koolmanojwong, S., Lam, A., Boehm, B.: Comparative experiences with electronic process guide generator tools. In: Proceedings of the International Conference on Software Process. pp. 61–72. ICSP'07, Springer-Verlag (2007), `http://portal.acm.org/citation.cfm?id=1763239.1763247`
18. Raunak, M., Osterweil, L., Wise, A., Clarke, L., Henneman, P.: Simulating patient flow through an emergency department using process-driven discrete event simulation. In: SEHC '09: Proceedings of the 2009 ICSE Workshop on Software Engineering in Health Care. pp. 73–83. IEEE Computer Society (2009)
19. Scott, L., Carvalho, L., Jeffery, R., Becker-Kornstaedt, U.: Understanding the use of an electronic process guide. Information and Software Technology 44, 601–616 (2002)
20. Simidchieva, B.I., Marzilli, M.S., Clarke, L.A., Osterweil, L.J.: Specifying and verifying requirements for election processes. In: Proceedings of the 2008 International Conference on Digital Government Research. pp. 63–72. dg.o '08, Digital Government Society of North America (2008), `http://portal.acm.org/citation.cfm?id=1367832.1367846`
21. Wang, D., Pan, J., Avrunin, G.S., Clarke, L.A., Chen, B.: An automatic failure mode and effect analysis technique for processes defined in the little-jil process definition language. In: Proceedings of the International Conference on Software Engineering and Knowledge Engineering (SEKE 2010). pp. 765–770 (2010)