Generating Natural-language Process Descriptions from Formal Process Definitions

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

Tiffany Y. Chao*
Dept. of Computer Science
University of Massachusetts
Amherst, MA 01003
tiffany.y.chao@boeing.com

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

ABSTRACT

Process models are often used to support the understanding and analysis of complex systems. The accuracy of such process models usually requires that domain experts carefully review, evaluate, correct, and propose improvements to these models. Domain experts, however, are often not experts in process modeling and may not even have any programming experience. Consequently, domain experts 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 natural language process descriptions in existing electronic process guides, these process descriptions are generated completely automatically and can describe complex process features, such as exception handling, concurrency, and non-determinisitc choice. The generated process descriptions have been well-received by domain experts from several different fields, and they have also proven useful to process programmers.

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 human agents, 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, human agents 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 human agents play. To develop accurate models of a human-intensive process, it is usually important that the domain experts carefully review, evaluate, correct, and propose improvements to these models. Domain experts, however, are often not experts in process modeling and may not have any programming experience. Consequently, domain experts 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 (e.g., exception handling) or data flow (e.g., call by value).

To help domain experts understand complex processes, we have developed the Little-JIL Narrator. The Narrator takes a Little-JIL [6] process definition (i.e., the 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, hyperlinked, 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 extremely 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. Moreover, it 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 definition from which it is derived. 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 definition based on a real-world medical process that we have elicited from domain experts and then provides and explains the generated narrative. Sec-

*Tiffany Chao now works at The Boeing Company, Seattle, WA 98124.

ICSSP'10, May 21-22, 2011, Honolulu, Hawaii, USA.

tion 4 then provides an overview of the Narrator design. Section 5 describes our experience with the Narrator and some issues that were encountered. The final section concludes with a discussion of future research directions.

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 an input to a 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 often their hierarchical decomposition), artifacts and resources, roles (or agents), 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 also 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 associated with a given activity to decide how much control flow information to include.

For complex system processes, such as medical processes, precise activity ordering or control flow information can be very important. Medical guidelines, for instance, can define strict constraints on the order of performing certain activities. Also, medical processes exhibit a high degree of concurrency (as different medical professionals can work on the same process in parallel) and non-deterministic choices made by agents at runtime. Thus, understanding and reasoning about such complex processes by various stakeholders necessitates the inclusion of such control flow information in the corresponding natural-language description. Relying on the process engineer to manually include such control flow information is not satisfactory. A manual approach is error-prone, time-consuming, and does not guarantee completeness of the included control-flow information. Furthermore, a manual approach makes maintaining the natural-language description and ensuring that it stays consistent with the associated process model difficult.

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.

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 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 Little-JIL process definition for a simplified 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.

In our model, the chemotherapy process starts with a patient consultation where an oncologist examines the patient's biopsy results, makes a diagnosis, decides whether chemotherapy is needed, and accordingly creates a treatment plan and medication orders. To increase safety, a sequence of redundancy checks usually follows. A nurse and a pharmacist perform a set of verifications ranging from ensuring that the ordered medication are appropriate for the patient's diagnosis (according to best practices from the medical literature) to checking the recentness of the patient's height and weight data on which chemotherapy doses are based. A nurse practitioner needs to conduct a teaching session with the patient to explain the chemotherapy medications and their side effects, to obtain the patient's informed consent for chemotherapy administration, and op-

tionally, depending on prescribed medications, to set up the patient's intravenous access. Once all these preparatory tasks are performed, the patient can be administered chemotherapy. On the day of chemotherapy administration, a clinic nurse performs final checks to make sure that the patient's condition allows the treatment, collaborates with the pharmacist who prepares the chemotherapy medication, and finally administers the medications to the patient.

The chemotherapy process, which often spans several days or even weeks, exhibits several kinds of complexities. For instance, the chemotherapy process involves a variety of process performers, or agents, who need to carefully and effectively coordinate their actions to perform the process safely and efficiently. Exceptional scenarios can significantly complicate the process. For example, a doctor may prescribe a medication that is inappropriate for the diagnosis or a medical assistant might record incorrect patient weight, which can trickle down the process and eventually affect the doses of the administered chemotherapy medications. Such problems need to be detected and then dealt with carefully.

For this example, we present a part of the definition of the chemotherapy process described above in the Little-JIL process modeling language. Little-JIL [6] is a process modeling language with rich semantics that are useful in capturing the complexities of real-world processes such as the chemotherapy process. Little-JIL's semantics are also formally defined, which makes process definitions amenable to various kinds of automated analysis (e.g., [7, 8, 18, 21]).

The main building block of a Little-JIL process definition 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 definitions contain a hierarchical decomposition of steps, where steps at a higher level of abstraction are decomposed into substeps at a lower level of abstraction and the decomposition can continue to any desired level of detail. For example, the high-level step *prepare for and administer first cycle of chemotherapy* in Figure 1 is decomposed into five substeps—*perform consultation and assessment*, *perform initial review of patient records*, and so on.

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 regimes require the installation of a portacath, the agent may decide not to execute this step for some patients.

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. *perform consultation and assessment* is also a sequential step and thus its substeps need to be performed in left to right order.

The substep *create treatment plan and orders* is a *parallel step* (indicated by the equal sign on the step rectangle), which means that its substeps can be executed in any order, including simultaneously. This represents the fact that the oncologist can choose to create the treatment plan, the chemotherapy orders, and the premedication orders in any order. Furthermore, the oncologist can choose between two alternative options for how to create the treatment plan—using a standardized treatment plan template (a careset) or creating the treatment plan from scratch. The process definition in Figure 2 captures this fact by making *create treatment plan* a choice step. A *choice step* (represented by a circle over a line in the step rectangle) means that the agent can complete the step by choosing and completing one of the substeps successfully.

While performing the step *confirm pathology report indicates cancer* in Figure 2, the oncologist may discover that the pathology report in fact does not indicate cancer. When this unusual circumstance arises, it needs to be properly addressed and requires a deviation from the normal process flow. This 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. In this example, the exception *PathologyReportDoesNotIndicateCancer* is handled by the handler *consider alternative treatment* attached to the "X" in the step rectangle of the step *prepare for and administer first cycle of chemotherapy* in Figure 1. Once the handler step *consider alternative treatment* 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 successfully completed.

In Little-JIL, each step is performed by humans or automated agents, where an agent is considered to be a resource that must be obtained before the step can be commenced. For example, the step *perform initial review of patient records* in Figure 1 is performed by a Practice Registered Nurse (RN) and a Pharmacist, and the steps *obtain patient informed consent* and *install portacath* are performed by a Nurse Practitioner. 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 definition 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 definitions attractive to system engineers and other people trained in these notations, non-technical people may prefer textual rather than a graphical visual representations. Figure 3 shows the automatically generated textual narrative created by the Narrator from the process definition 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 definition and uses the same icons used in the diagramatic representation to represent the step kinds (e.g., sequential, parallel, leaf step). The parent/child relationship from the Little-JIL process definition tree is captured by the indentation in the table of contents. For example, the steps at one level of indentation under *prepare for and administer first*

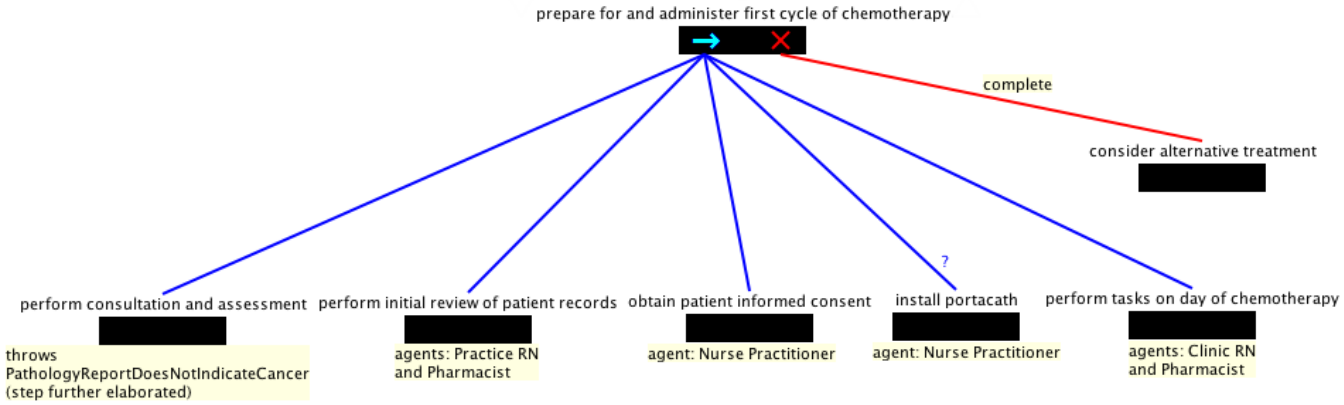

Figure 1: An example chemotherapy process.

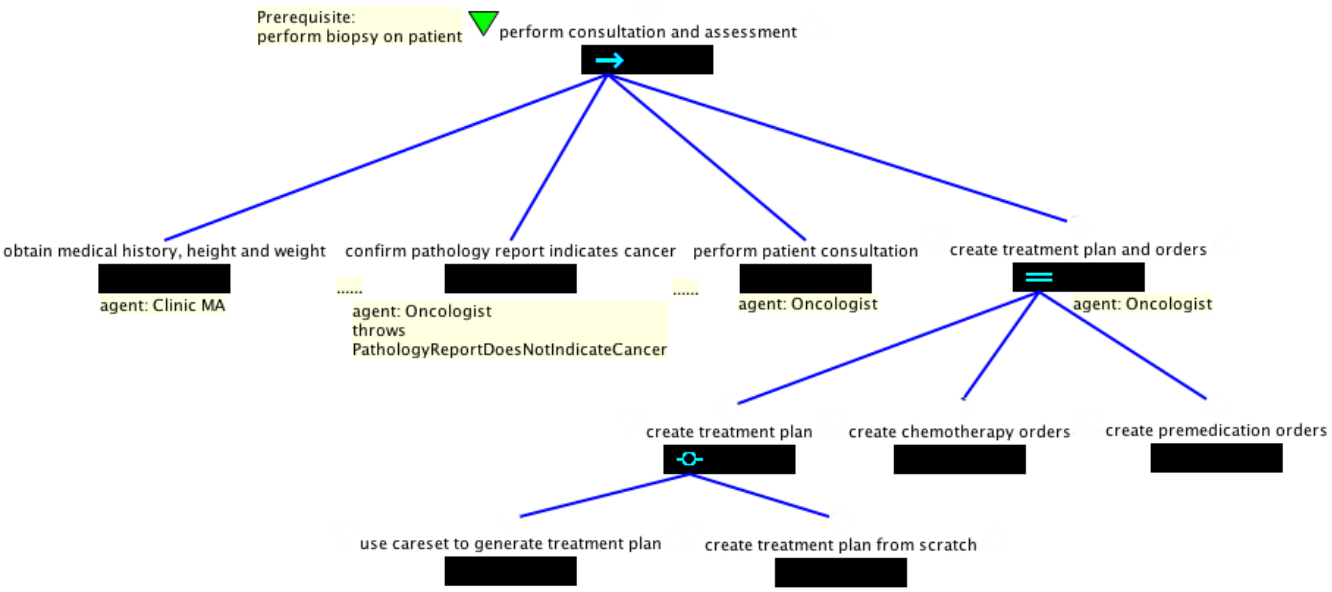

Figure 2: Elaboration of *perform consultation and assessment.*

cycle of chemotherapy, namely *perform consultation and assessment*, *perform initial review of patient records*, and so on, are the substeps of *prepare for and administer first cycle of chemotherapy.*

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 definition. 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 definition in Figures 1 and 2 where familiarity with the notation semantics is assumed, the descriptive part presents carefully selected sentences that explain the process in natural language . For example, the step section for *Perform Consultation And Assessment* in Figure 3 explains what it means for the step to have a prerequisite, namely that the prerequisite step *perform biopsy on patient* needs to be completed before the step can be started. Similarly, it 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 ex-

Figure 3: The automatically generated narrative for the process definition in Figures 1 and 2.

ample, 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 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 diagramatic view of the process definition. 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

This section presents the architecture of the Little-JIL Narrator and explains the constituting components.

Figure 4 shows the high-level architecture of the Narrator. The *Little-JIL Process Definition*, the *Phrasing Templates*, and the *Customization Rules* are used as inputs 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 that is to be generated. The *Formatting Weaver* then combines the *Narrative Content* together with the *Formatting Templates* to produce the final *Generated Narrative*. The Little-JIL Process Definition artifact represents any process definition created in the Little-JIL language (like the one described in section 3).

Phrasing Templates. The Phrasing Templates are parameterized, natural-language phrases that correspond to the different semantic features of the Little-JIL process language. A phrasing template expresses in natural language

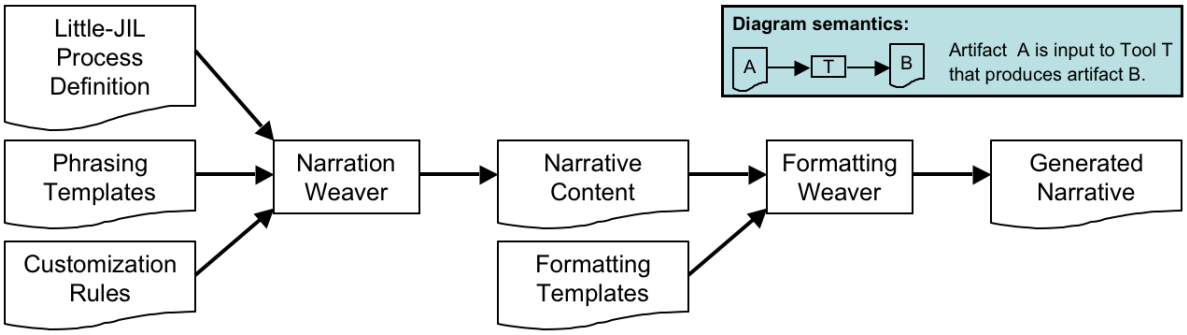

Figure 4: Architecture of the Little-JIL Narrator.

the meaning of a particular semantic feature (e.g., what it means for a step to be sequential), where the parameters represent information that is specific to a given process definition.

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. That is, if a process step "do activity A" is sequential, that phrasing template can be used to generate a sentence saying that to do activity A, its substeps need to be done in the listed sequential order (<*substepsList*> will be substituted with a list of the substeps of activity A). The text snippet starting with *To "prepare for and administer...* and ending with the bulleted substep list in the step section for *Prepare For and Administer First Cycle of Chemotherapy* in Figure 3 was produced using this phrasing template to describe the substep ordering semantics of the root step in Figure 1.

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.

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. The user could choose, other synonyms instead, such as "activity" or "task."

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>.

Figure 5: Example phrasing templates.

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 definition. Before the narrative shown in Figure 3 was generated, the option to show resources was selected. Thus, resource information, such as the human resources (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 process but not intermediate process 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. Consider the substeps of *Prepare For And Administer First Cycle of Chemotherapy* in Figure 3. In this example, they are shown as a list following the phrase *To "prepare for ...", the following need to be done in the listed order.* The user could alternatively choose the following comma-separated version for that sentence:

To "prepare for and administer first cycle of chemotherapy", first perform consultation and assessment, then perform initial review of patient records, then obtain patient informed consent, then install portacath (optionally), and finally perform tasks on the day of chemotherapy.

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. DISCUSSION

This section discusses our experiences applying the Little-JIL Narrator to definitions of real-world processes and some of the issues that arose.

5.1 Experience

We have used the Narrator to generate natural-language descriptions of several large Little-JIL definitions 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 definition. These process definitions 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 definition had 467 Little-JIL steps (207 of which were related to the handling of 59 exceptional situations); the blood transfusion process definition had 248 Little-JIL steps (37 of which were related to the handling of 15 exceptional situations); and the ODR process definition had 209 Little-JIL steps (108 of which were related to the handling of 19 exceptional situations).

Even though it is an early prototype, our experiences with the Little-JIL Narrator have been very positive and promising. The medicial 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 definition, 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 definition 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 definition 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 process programmers as well. Little-JIL process programmers use the Little-JIL visual editor to create and edit process definitions. Being able to look at the natural-language narrative helped reveal errors in the process definition that were not immediately noticeable in the diagrammatic representation. For example, artifact flow is not easily visible in the diagramatic representation and inspecting the generated narrative helped pinpoint omissions from or unnecessary additions of artifacts to certain process steps. Similarly, the diagrammatic specification of the number of times a step needs to be executed (e.g., an asterisk for 0 or more times) could be easily overlooked by a Little-JIL process programmer. Seeing a phrase, such as "this step should be done 0 or more times" in the generated narrative, however, makes this information easier to spot and correct, when the given step should not be executed that many number of times. The ability to automatically and quickly (it takes about a second for a process definition with several hundred steps) generate a narrative makes the Little-JIL Narrator a useful tool for debugging process definitions.

Perhaps one obvious disadvantage of the generated Narrative is its size. The natural-language descriptions of the non-trivial process definitions 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 definition created in a semantically rich process language, such as Little-JIL. 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 definition (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 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.

5.2 Issues

Some interesting issues arose during the design, implementation, and use of the Little-JIL Narrator.

To generate text automatically that reads naturally to humans requires that the process definitions be written ac-

cording to some guidelines or conventions. To be able to plug step names from the Little-JIL process definition directly into the phrasing templates, for example, we assume that step names start with a verb. For instance, the parameter <*stepName*> in the first two phrasing templates in Figure 5 need to be substituted by a step name starting with a verb, otherwise the generated sentence will not read naturally. Our experience with the Little-JIL Narrator, and with process definitions in general, indicates that this assumption about step names is reasonable and not very limiting. Since process steps correspond to activities, most of the time step names are phrases that indeed start with a verb. Establishing a convention for naming process steps in such a way did not seem to be hard to follow or impose any artificial constraints when we were developing definitions for several large processes from the medical and negotiations domains.

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 to be included in the generated narrative. For example, Little-JIL's choice step represents non-deterministic choice. 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. Generating a natural-language sentence that explicitly describes what happens when a substep of a choice step fails is impossible, however, because statically it is not known which substep will be chosen and hence which substeps will remain as alternatives if the first chosen step fails. We dealt with this issue by trading off the precision of the generated natural-language text for its correctness. Thus, when generating the natural language for the situation when a substep of a choice step fails, 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. Thus, it is not possible to determine statically if an artifact gets used or modified in a step, which again necessitates trading off precision for correctness of the generated natural-language description. Moreover, simply mentioning in the generated narrative that an artifact passes through a step is not satisfactory because it is not very informative and, more importantly, because such information is too tightly coupled with the artifact flow technicalities of the process language, which might not be of interest to domain experts. 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 of artifacts is to have the creator of the process definition explicitly provide information when an artifact is used and/or modified and when it simply passes through a step. Such annotations would also improve the accuracy of some of the analysis approaches that we also apply.

Another challenge is representing the rich exception handling semantics of Little-JIL in natural language. 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 handle 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

The Little-JIL Narrator automatically generates a hyperlinked natural-language description based on a formal process definition specified in a semantically rich process modeling language. The generated narrative describes process constructs such as resources, artifacts, activities, and agents. It also provides detailed and precise information about the control flow of process activities, including exception handling, concurrency and non-deterministic choice. The customization rules supported by the Little-JIL Narrator allow the user to tailor the generated narrative by selectively filtering out information, such as resource and artifact descriptions. The flexible template-based design of the Little-JIL Narrator allows modifications of the phrasing used to describe the formal semantics of the process language as well as modifications of the presentation style of the generated narrative to be easily made.

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 unfamiliar with all the terminology and who use the generated narrative as a process guide. It could also help process programmers who are not necessarily 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 increasing usability, there is some evidence suggesting that 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 by an individual with that specific process role.

The Little-JIL Narrator could also benefit from a better user interface for editing the phrasing templates and the customization rules. Currently, these are expressed in XML format and, despite the fact that the XML schema used is very simple, having a graphical user interface for editing the templates and the rules would make the tool more accessible to non-technical users.

The index generated by the Narrator could also be im-

proved. 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 definitions of real-world processes from the medical and negotiations domains; the generated narrative has been well-received by domain experts. The generated narrative also seems to be useful to process programmers for discovering errors in a process definition.

The main focus of the Little-JIL Narrator is natural language generation and it aims to support stakeholders who are uncomfortable with formal notations or stakeholders who are technically savvy but can gain additional insight into the process (or its model) from its natural-language description. Such a generated narrative could be a useful addition to existing electronic process guides as it complements other components of these systems, such as a graphical process representation. Unlike natural language descriptions in existing EPGs, the narrative is generated completely automatically and supports the description of complex process features, such as exception handling, concurrency and nondeterminisitc choice. We believe that the Little-JIL Narrator approach is an important step towards automatically generating natural-language descriptions of non-trivial process models specified in semantically-rich process languages.

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.

8. REFERENCES

[1] Adonis, www.boc-group.com/at.

[2] Aris, www.ids-scheer.de.

[3] XSL transformations (XSLT) version 2.0, www.w3.org/tr/xslt20.

[4] U. Becker-Kornstaedt, D. Hamann, R. Kempkens, P. Rösch, M. Verlage, R. Webby, and J. Zettel. 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*, CAiSE '99, pages 119–133. Springer-Verlag, 1999.

[5] U. Becker-Kornstaedt and M. Verlage. The V-modell guide: Experience with a web-based approach for process support. In *Proceedings of Software Technology and Engineering Practice (STEP)*, pages 161–168. IEEE Computer Society Press, 1999.

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

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

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

[9] S. Christov, B. Chen, G. S. Avrunin, L. A. Clarke, L. J. Osterweil, D. Brown, L. Cassells, and W. Mertens. 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] L. A. Clarke, L. J. Osterweil, and G. S. Avrunin. Supporting human-intensive systems. In *Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research*, FoSER '10, pages 87–92. ACM, 2010.

[11] C. Damas, B. Lambeau, F. Roucoux, and A. van Lamsweerde. Analyzing critical process models through behavior model synthesis. In *ICSE '09: Proceedings of the 2009 IEEE 31st International Conference on Software Engineering*, pages 441–451. IEEE Computer Society, 2009.

[12] P. Haumer. Increasing development increasing development knowledge with EPFC. *Eclipse Review*, pages 26–33, 2006.

[13] M. Kellner, U. Becker-Kornstaedt, W. Riddle, J. Tomal, and M. Verlage. Process guides: Effective guidance for process participants. In *Proeceedings of the International Conference on the Software Process*, pages 11–25, 1998.

[14] J. D. Lee and K. A. See. Trust in automation: designing for appropriate reliance. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 46:50–80, 2004.

[15] L. J. Osterweil and L. A. Clarke. Supporting negotiation and dispute resolution with computing and communication technologies. In *Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research*, FoSER '10, pages 269–272. ACM, 2010.

[16] R. Parasuraman and V. Riley. Humans and automation: use, misuse, disuse, abuse. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 39(2):230–253, June 1997.

[17] M. Phongpaibul, S. Koolmanojwong, A. Lam, and B. Boehm. Comparative experiences with electronic process guide generator tools. In *Proceedings of the International Conference on Software Process*, ICSP'07, pages 61–72. Springer-Verlag, 2007.

[18] M. Raunak, L. Osterweil, A. Wise, L. Clarke, and P. Henneman. 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*, pages 73–83. IEEE Computer Society, 2009.

[19] L. Scott, L. Carvalho, R. Jeffery, and U. Becker-Kornstaedt. Understanding the use of an electronic process guide. *Information and Software Technology*, 44:601–616, 2002.

[20] B. I. Simidchieva, M. S. Marzilli, L. A. Clarke, and L. J. Osterweil. Specifying and verifying requirements for election processes. In *Proceedings of the 2008 International Conference on Digital Government Research*, dg.o '08, pages 63–72. Digital Government Society of North America, 2008.

[21] D. Wang, J. Pan, G. S. Avrunin, L. A. Clarke, and B. Chen. 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)*, pages 765–770, 2010.