UWoN (User Workflow Notation): A Visualization Tool for User Workflow Analysis and Modeling for Developing Healthcare IT

Xianjun Sam Zheng[1,4], Stefan Christov[2], Yu Sun[3], Xiping Song[1]
[1]Siemens Corporate Research, [2]University of Massachusetts Amherst,
[3]University of Alabama at Birmingham, [4]Tsinghua University

Successful user interface (UI) design for healthcare IT systems requires a solid understanding of the users' workflow so that designers can create a solution that delivers the "right information to the right user at the right time". There isn't, however, an effective tool during the requirement elicitation phase to document user workflow in a systematic, formal, and intuitive way. Here, we propose UWoN (User Workflow Notation), which is heavily influenced by formal process modeling languages. In particular, UWoN builds on the rich and rigorous semantics of the Little-JIL process definition language, but aims to make the visual syntax more intuitive to UI designers, who often lack experience with engineering notations. We introduce the idea of representing complex workflows with the "icicle" visualization concept and compare it to the traditional hierarchical tree visualization. We illustrate and discuss one example of applying the UWoN to model and analyze user workflow for a healthcare intervention application.

INTRODUCTION

Healthcare IT systems are becoming increasingly complex – providing more functionality, recording and presenting more data, and involving more users with different experience and specialty. Given this increasing amount of complexity, when the user interface (UI) is poorly designed users are often overwhelmed and struggle to find the right functionalities and data for potentially time-sensitive tasks (Sun & Gregory, 2007). Indeed, the complexity and poor design of health information systems can lead to medical errors which in turn can endanger patients' lives as reported by the Institute of Medicine (Kohn, Corrigan, & Donaldson, 1999).

Successful UI design requires a solid understanding of the users' workflow so that designers can create a solution that delivers the "right information to the right user at the right time". Compared to the functionality-driven approach, workflow-driven design can be an effective way to manage the complexity of healthcare IT systems. Take medical interventions for an example, many clinical procedures in today's interventional environment (or healthcare in general) are highly standardized with protocols and guidelines: the similar tasks, sub-tasks, steps, & sequences need to be fulfilled to achieve the desired goals. However, there isn't an effective tool during the requirement elicitation phase to document the user workflow in a systematic, formal, and intuitive way.

On the one hand, UI specialists or designers often use various flow charts and custom diagrams to depict user workflow visually, allowing them to manually analyze the workflow model to discover patterns that can potentially inform design decisions. This practice, however, is not formal: it generates ambiguity during communication (different people would use different notations which often do not have well-defined semantics) and does not support quantitative analysis of the model. Automated analysis will become critical once the user workflow model grows large and complex. On the other hand, several existing engineering modeling methods, such as Little-JIL (Cass, Lerner, Sutton, McCall, Wise, & Osterweil, 2000) or CTTE (Mori, Paterno, & Santoro, 2002), can be applied to model and analyze user workflow. These methods are rigorous and formal. However, their visual representations are not so intuitive. It is difficult to visually examine the complex user workflow model to explore patterns.

Here, we propose UWoN (User Workflow Notation), which is heavily influenced by formal process modeling languages. In particular, UWoN builds on the rich and rigorous semantics of the Little-JIL process definition language, but aims to make the visual syntax more intuitive to UI designers, who are often not experienced with engineering notations. We introduce the idea of representing complex workflows with the "icicle" visualization concept and compare it to the traditional hierarchical tree visualization.

Furthermore, the formal semantics of UWoN allow for quantitative analysis of workflow models that can potentially inform the UI design. UI design usually involves some kind of manual analysis of the user workflow, and this analysis can be combined with the experience of the designer(s) to create a user interface solution.

RELATED WORK

There are a number of approaches and notations for representing user workflow and some of them have been used to inform the design of interactive systems. For instance, ConcurTaskTrees (CTT) (Mori, Paterno, & Santoro, 2002) represents the user workflow as a hierarchical decomposition of tasks and have semantic constructs to express various kinds of control flow and data flow (see Figure 1 below for an example). CTT has formal semantics and its models have been used for simulation as well as to support model checking. CTT has visual syntax where tasks are represented with nodes in a task

decomposition tree and control flow is represented with edges between different tasks. In CTT, however, there is some degree of coupling between control flow and data flow (e.g. one of the control flow operators is "concurrency with information exchange") and this results in a large number of control flow operators that need to be visualized as illustrated in Figure1. The authors of CTT indeed report that users have found the visual syntax of CTT control flow operators difficult to understand (Mori, Paterno, & Santoro, 2002).

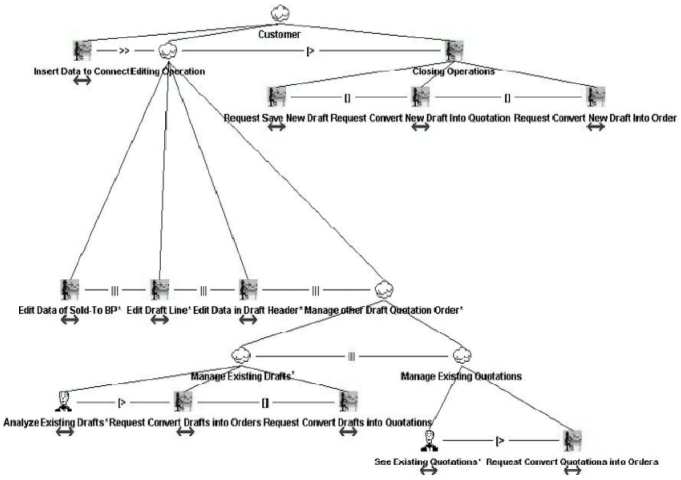

Fig 1. An example of user workflow model in CTT.

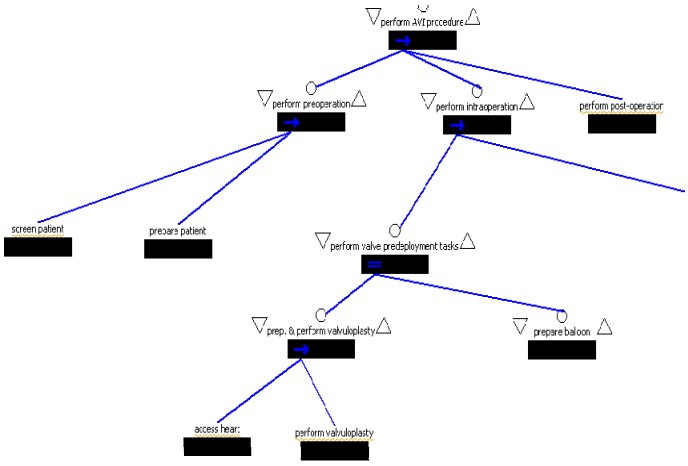

Fig 2. An example of user workflow model in Little-JIL.

Another notation that has been successfully used to model workflow is the Little-JIL process definition language (Wise, et al, 2000) on which UWoN currently bases most of its semantics. In Little-JIL, the workflow is also represented as a hierarchical decomposition of tasks (see Figure 2). The language has rich semantics that can represent various aspects of control flow (e.g. sequencing, parallelism, synchronization, exception handling). In addition, Little-JIL has semantics to represent the use of artifacts and resources by the different activities in a Little-JIL process model. Little-JIL's semantics are formally defined, which has made Little-JIL process definitions amenable to various kinds of automated analysis, such as model checking, fault tree analysis, and discrete-event simulation.

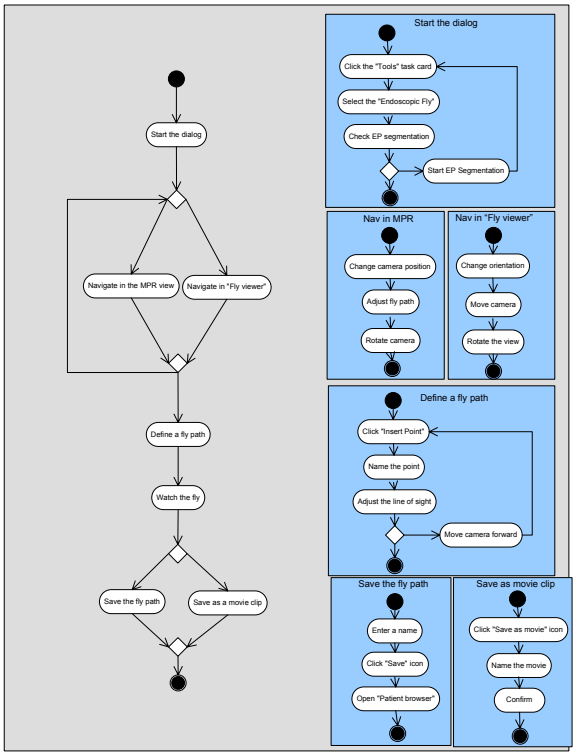

Fig 3. An example of user workflow model in UML, which shows a simple workflow of medical intervention with two hierarchical levels of tasks (the first level tasks are shown in the gray box, the second level ones are shown in the light blue boxes).

Little-JIL has visual syntax where, similar to CTT, the tasks are nodes in a task decomposition tree. Control flow is decoupled from data flow in Little-JIL, which results in fewer control flow operators to visualize compared to CTT. In addition, control flow operators are visualized by a badge on the nodes representing tasks, which further decreases the amount of visual clutter compared to CTT, where control flow operators are represented by edges between nodes (see Figure 2 and Figure 6). Even with this improved visualization, however, control flow is not always intuitive to understand in a Little-JIL task tree. The tree notation also suffers from scalability issues in the sense that the size of a tree diagram grows very quickly with the number and level of tasks included in that diagram.

UML Activity Diagrams have also been used to capture user workflow (OMG, 2012). Activity diagrams have semantics to support aspects of control flow such as sequencing, parallelism, synchronization and, to some extent, exception handling. Unlike CTT and Little-JIL's visual syntax, the visual syntax of Activity Diagrams is based on a flowchart-like notation rather than hierarchical task decomposition. This makes the control flow somewhat easier to understand, but Activity Diagrams still become

cluttered quickly even for relatively small workflows. Figure 3 shows an example of a relatively simple workflow of medical intervention, which only contains two levels of tasks. In addition, it is difficult to get an overview of the hierarchical decomposition of tasks. Understanding and visualizing the hierarchical task decomposition is important in UI design as it provides insight into how UI elements can be grouped together, in addition to what the flow of control between or sequence of use of these UI elements should be.

Use cases and scenarios written in natural language have also been used to document user workflow (Alexander, & Maiden, 2004). Quite often, however, they are inadequate to serve as a good basis for UI design. The natural language could often be a source of ambiguity and hierarchical decomposition and various aspects of control flow (e.g. parallelism and exception handling) are hard to see and understand. In addition, it is often very hard for UI designers to aggregate various use cases together to decide how to build a UI that covers all the given use cases.

USER WORKFLOW NOTATION

The User Workflow Notation (UWoN) is intended to support modeling of individual user tasks (including interaction with computer systems) as well as the overall workflow of complex software systems. UWoN is being developed with three main goals in mind: rich semantics, formal semantics, and intuitive syntax. To achieve that, UWoN leverages work from the process/workflow modeling domain as well as from the information visualization domain.

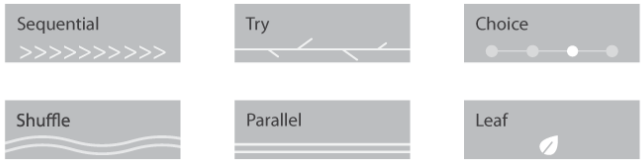

Fig 4. Visual syntax for various control flow in UWoN.

Syntax and Semantics

As in Little-JIL, the core of a UWoN workflow model is the step. A step corresponds to an activity performed by a human or a machine, and it can be hierarchically decomposed into substeps up to any level of detail. A step determines the order of execution (i.e. control flow) of its substeps. A sequential step (see Figure 4 for an example), for example, means that its substeps need to be executed in left-to-right order. In addition to the five Little-JIL step kinds, UWoN also supports the shuffle step. A shuffle step (see Figure 4) means that all of its substeps need to be executed, but they can be executed in any non-overlapping order. Shuffle can be expressed with a combination of choice and sequential steps, but this can make the workflow model unnecessarily complex, especially when the shuffle step has a large number of substeps. Since one of the goals of UWoN is intuitive syntax and since, based on our experience, shuffle steps occur often in workflows, we decided to provide direct support for shuffle steps in UWoN's syntax.

Visualization

UWoN currently visualizes a workflow as a hierarchical decomposition of tasks, but uses the icicle visualization concept instead of a standard tree. In the icicle visualization, each "row" of steps corresponds to a level from the workflow tree. The root task (which is also the highest level task) is shown on the top as a rectangle spanning the width of the tree. The tasks directly under the root are the subtasks that the root task is decomposed into. The same rules apply for the decomposition of the root's subtasks, their subtasks and so on.

Similar to most hierarchical task decomposition workflow visualizations, representing control flow is the main challenge with the icicle visualization as well. Currently, UWoN addresses this challenge by placing a control flow visual pattern at the bottom of each rectangle representing a task. Figure 4 shows the visual patterns for the six kinds of control flow supported in UWoN. The main motivation behind these visual patterns was to make them intuitively correspond to the control flow semantics they represent and also make them scale on the horizontal dimension as icicle visualizations (and any tree-like visualization) can easily become quite wide.

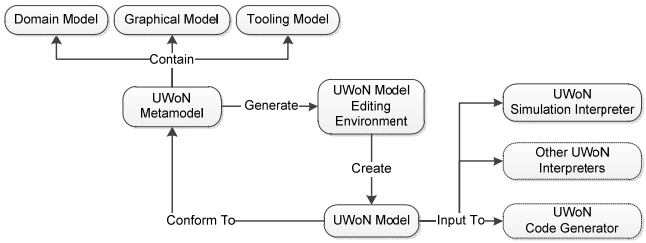

Fig 5. Overview of the extensible UWoN tool architecture

Architecture/ Tool Support

In order to achieve the design goals of UWoN (i.e., intuitive syntax and rich and formal semantics), the tool support shall be capable of precisely defining the syntax and semantics, as well as enabling the intuitive visualization and a user-centric editing environment. Additionally, UWoN shall also support functionalities for quantitative analysis, such as workflow simulation, workflow metrics calculation, and code generation. Therefore, the tool support for UWoN should provide an extensible framework that can integrate different functional components with the basic editing environment.

Domain-Specific Modeling (DSM) (Lédeczi et al., 2001) is applied to facilitate the tool implementation for UWoN. DSM is a Model-driven Engineering (Schmidt, 2006) methodology that uses a Domain-Specific Modeling Language (DSML) (Gray et al, 2007) to declaratively define a system using specific domain concepts, directly compute and analyze the domain through model interpreters, and automatically generate the desired software artifacts by model transformation engines and generators. Considering the workflow notation as an application domain, the formal specification of the metamodel for this domain need to defined first. The metamodel (Atkinson, & Kuhne, 2003) is

used to specify the entities, associations and constraints for the workflow domain, which can be used to generate a modeling environment, enabling users to build concrete models and define the workflow. The models conform to the definition of the metamodel and can be used to compute and generate of other software artifacts.

Figure 5 shows the high-level architecture of the UWoN tool support. We used the Graphical Modeling Framework (GMF, 2010) , a powerful DSM tool in Eclipse, to support the implementation. The metamodel for GMF consists of three components: 1) the abstract syntax of the notation that is captured in the domain model 2) the concrete syntax (i.e., the intuitive visualization) that is specified in the graphical model 3) the tooling model which defines the features of the editing environment (e.g., palette, creation buttons, actions). This metamodel is applied for automatic creation the UWoN modeling editor, which will be used by end-users to create concrete workflow models. The separation of domain model, graphical model and tooling model supports the extensible framework, so that any changes to the visualization will not affect the notation definition or the editing environment, and any alteration of the domain model or tooling model will not cause the changes of other two.

The rich and formal semantics of UWoN is implemented through the UWoN simulation interpreter. The semantics of each workflow step is defined by a finite-state machine, and the simulation interpreter takes an UWoN model as the input and executes the workflow by simulating the working process. The interpreter controls the whole execution process, asks users input, and present the simulation process. Through the simulation, users can gain a better understanding about the specific workflow.

The simulation interpreter is implemented as a plug-in to the modeling environment. Similarly, other functional components can be developed in the same way. The UWoN modeling environment provides a set of APIs to access and control the models, so different plug-ins can be readily integrated with the UWoN editing environment without interfering other existing components. Furthermore, a number of model transformation and code generation tools are available in Eclipse, which offers the potential to generate UI code from the UWoN models automatically.

AN EXAMPLE OF MEDICAL INTERVENTION WORKFLOW

We have applied the UWoN method to visualize and model the user workflow in several projects of developing software applications for image-guided medical intervention.

With the rapid advancement of medical imaging technology, more and more image-guided applications and tools are introduced and developed to support medical interventional procedures. There are many benefits of image-guided intervention, including more detailed planning prior to the operation, more accurate 2D/3D visual guidance during the procedure, and better outcome assessment after the procedure. As a result, the procedure is less invasive, takes shorter time, and patients can recover faster compared to the conventional intervention approach. However, the settings of clinical intervention are complex, typically involving multiple users (patient, physicians, radiologists, nurses, and technicians), multiple systems (imaging, anesthesia, vital sign, etc.), various devices (needle, syringes, catheters, guidewires, etc.), and different imaging modalities (x-ray fluoro, CT, MR, ultra-sound etc.). And yet even though different users perform different tasks by using different devices, tools, or data, they are working together to achieve a common goal, i.e., a successful medical intervention for the patient. It is critical to systematically analyze and model users' workflow so that designer can have a good understanding of the different user tasks, sub-tasks, steps, and their relationships before designing any new UI solutions.

UWoN for Aorta Valve Implementation

Here, we reported our experience in applying the UWoN method in one specific project, aortic valve implantation (AVI), which is a cardiovascular procedure and involves introducing and placing an artificial valve into the aorta of a patient's heart. The knowledge of the AVI procedure was acquired from previously conducted ethnography studies in several hospitals. This knowledge was documented and later validated by the different domain users (e.g., interventional cardiologists, nurses, and technicians). Based on the collected information, we then modeled and visualized the user workflow for AVI.

Figures 6 and 7 show the same part of the AVI workflow modeled and visualized in the Little-JIL standard tree notation and in the UWoN icicle notation respectively. In general, the high-level AVI workflow can be decomposed into three main phases: Perform Preoperation, Perform Intraoperation and Perform Postoperation. Each of these phases is further decomposed to tasks at a lower level of detail. For instance, Perform Preoperation task is further decomposed into two sub-tasks: Screen Patient and Prepare Patient, which are leaf steps. The task of Perform Intraoperation is more complex: it is decomposed into two sub-tasks: Perform Valve Predeployment and Deploy Value, which are further decomposed into corresponding lower level leaf steps.

As discussed earlier, UWoN inherits the syntax and semantics from Little-JIL but aims to improve the visual syntax by making it more intuitive to UI designers. This is done by incorporating several key design considerations. First one is to design more intuitive control flow icons. The new visual syntax was created by a group of professional visual designers through several design and feedback iterations. Specifically, each type of control flow icon is now designed as a visual texture pattern rather than as a single icon object (as in the Little-JIL), and the control flow indicator is placed at the bottom of each rectangle shape, which represents a task step. This design consideration makes each task step look and also behave like a physical

"brick" or "tile", which has a name (i.e., task name) and an attribute (i.e., control flow type). Different "bricks" or "tiles" can be easily stacked up together either horizontally or vertically. Once there are a group of "bricks", one can start seeing global patterns because the local texture pattern for each "brick" will emerge as a global texture pattern for many "bricks". In fact, the more "bricks" with the same control flow type are stacked up together, the stronger the global control flow pattern will emerge. For instance, since the AVI workflow has many sequential task steps, Figure 7 shows a stronger impression of global pattern for sequencing control flow as compared to Figure 6.

Moreover, the composition of task steps as icicle visualization not only preserves the hierarchical information, but also eliminates the edge connections. This design consideration makes the whole user workflow visualization more compact and organized as compared to the standard tree visualization. As Figure 6 and 7 illustrate, while a standard tree nicely captures the hierarchical decomposition of the AVI workflow, it does not seem to scale as well as the icicle representation – with the icicle representation the same amount of information fits on smaller space. This is again due to the fact that edges are not included in the icicle representation as the parent-child relationship is expressed by the vertical alignment of the rectangles representing tasks.

In general, when the user workflow is modeled and visualized in UWoN, UI designers can gain a holistic and intuitive view of all the tasks and their relationships. This visualization is to support UI designers' exploring and discovering of user workflow patterns, which then can be used to inform their design decisions. Take Figure 7 for an example, one can quickly see that the global control flow for AVI procedure is sequential. Therefore, some kind of wizard design may be appropriate to support the user workflow. Of course, like the standard tree presentation, the icicle visualization also permits to inspect the details of each task. For instance, it is also easy to see some sub-task, such as Perform Valve Predeployment, has some parallel activities. Therefore, providing some communication indication in the UI can be useful to support the coordination of these parallel activities.

Last, because the notation of UWoN is formal, like other formal modeling tools, such as CTT and Little-JIL, it can benefit from various kinds of quantitative analysis, and the results can be useful by UI designers to validate their design.

DISCUSSIONS & FUTURE WORK

We proposed the UWoN tool to support UI designers to capture and document user workflow in a systematic, formal, and visually intuitive way. We briefly described the UWoN design and the architecture. We illustrate the "icicle" visualization using an interventional procedure example, and compare it to the hierarchical tree visualizations used in other workflow modeling tools.

Future research and implementation is needed to make UWoN a practically useful tool. For instance, we would like to incorporate other important aspects of control flow supported by traditional process/workflow modeling languages such as exception handling. We also plan to develop quantitative metrics (e.g., workflow complexity measurement) that are relevant to UI design. Lastly, but not the least, a systematic user evaluation of the UWoN tool would be necessary to determine how UWoN will work for UI designers.

ACKNOWLEDGEMENTS

We thank James Lin for valuable input and discussions. We also thank our designers, Nicola Vittori, Patrik Matos, Aran Mun, & Valeria Donati to help create the visualization. We thank Joe Carpinelli for the support of this research project.

REFERENCES

Cass, A. G. , Lerner, B. S., Sutton Jr., S.M., McCall, E. K., Wise A., and Osterweil, L. J.. Little-JIL/Juliette: a process definition language and interpreter. Proc. 22nd Intl. Conference on Software Engineering, ACM (2000), 754–757.

Kohn, L. T., Corrigan, J. M., and Donaldson, M. S. editors. To Err Is Human: Building a Safer Health System. National Academy Press, Washington, D.C., (1999).

Mori, G., Paterno, F., and Santoro C. CTTE: support for developing and analyzing task models for interactive system design. IEEE Transactions on Software Engineering, (2002), 797–813

OMG. Unified modeling language, superstructure specification, version 2.1.1. http://www.omg.org/spec/UML/2.1.1/Superstructure/PDF/

Sun, J.H. and Gregory, J. Designing for Cognitive Artifacts and Activities in Clinical Care. International Association of Societies of Design Research, (2007)

Schmidt, D. Model-Driven Engineering. IEEE Computer, vol. 39, no. 2, (2006), 25-32.

Lédeczi, A., Bakay, A., Maróti, M., Völgyesi, P., Nordstrom, G., Sprinkle, J., Karsai, G. Composing Domain-Specific Design Environments. IEEE Computer, vol.34, no. 11, (2001), 44-51.

Gray, J., Tolvanen, J., Kelly, J., Gokhale, A., Neema, S., Sprinkle, J. Domain-Specific Modeling. Handbook of Dynamic System Modeling, CRC Press, Chapter 7, (2007), 7-1 through 7-20.

Graphical Modeling Framework, http://www.eclipse.org/modeling/gmf/, 2010.

Atkinson, C., Kuhne, T. Model-Driven Development: A Metamodeling Foundation. IEEE Software, vol. 20, no. 5, (2003), 36-41.

Wise, A., Cass, A. G. , Lerner, B. S., McCall, E. K., Osterweil, L. J., Sutton, S.M. Using Little-JIL to Coordinate Agents in Software Engineering. Automated Software Engineering Conference, (2000), 155-163.

Alexander, I., Maiden, N. Scenarios, Stories, Use Cases. (2004) Wiley.

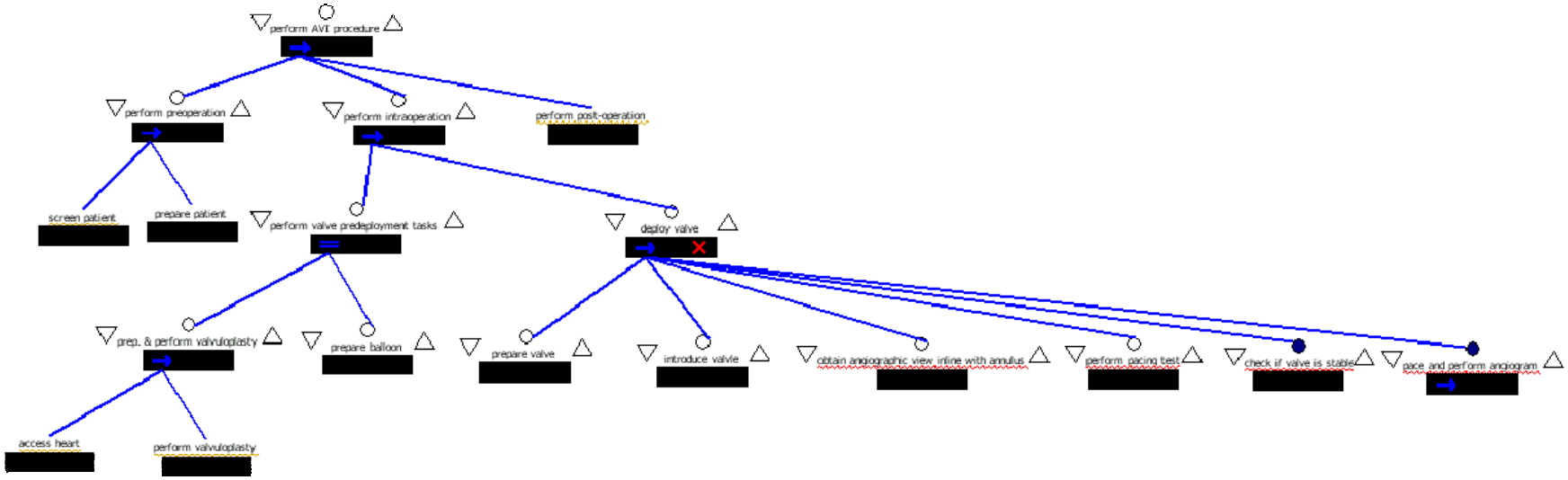

Fig 6. Little-JIL hierarchical tree visualization of the AVI (Aorta Valve Implantation) procedure.

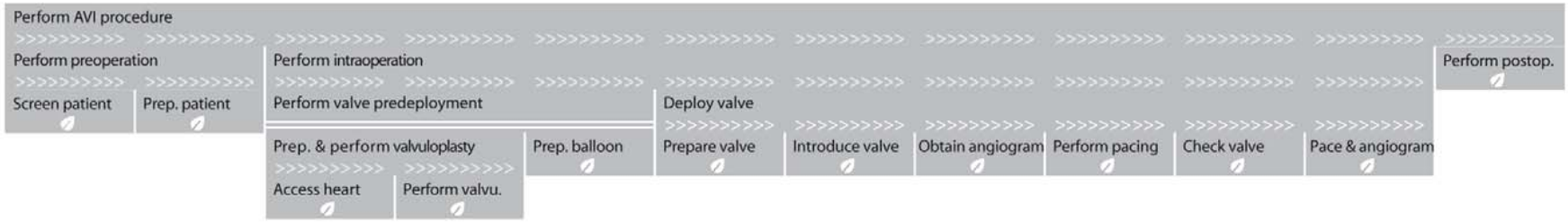

Fig 7. UWoN Icicle visualization of the AVI (Aorta Valve Implantation) procedure.