A Visualization of Medical Therapy Plans compared to Gantt and PERT Charts

Robert Kosara Silvia Miksch

Insitute of Software Technology (IFS), Vienna University of Technology
Favoritenstraße 9–11/E 188, A-1040 Vienna, Austria, Europe
http://www.ifs.tuwien.ac.at/asgaard/
E-mail: {rkosara, silvia}@ifs.tuwien.ac.at

Abstract

Medical therapy planning shares a number of properties of project management. It is, however, different in a few very important aspects — most notably, the more complex notion of time it requires.

Asbru is a language that can represent medical therapy plans in terms of time-oriented, skeletal plans. But due to its formal nature, it cannot be used directly by physicians. Therefore, we developed a visualization and user interface to deal with plans defined in Asbru, which can deal with most of its complexity.

We present this interface (called AsbruView), and discuss its features and advantages over the two representations typically used in project management: Gantt and PERT charts.

Areas of Research: Information Visualization, Time in problem solving

1. Introduction

Medical therapy planning plays an increasingly important role in the everyday work of physicians. It is based on the idea that a physician does not have to invent a new treatment for each patient, but rather can make use of the knowledge gathered by others (and him-/herself) by following a clinical guideline or protocol. Only patients that do not fulfill the conditions for applying such a standard treatment must be planned for separately.

Treatment planning is different from project management in several respects. The duration of a treatment is not known beforehand, and does not play a big role in the assessment of whether or not it was successful. A treatment plan can be aborted at any time if it turns out to be ineffective, and another plan can be used instead. Often, several different treatments or drugs have to be tried to find the best for a given patient.

There are a number of ways to specify clinical protocols (e.g., flow-charts, decision tables, etc.), but they lack many features needed in this domain (for a more thorough discussion see [5, 6, 7]).

Asbru is a language for specifying clinical protocols as time-oriented, skeletal plans. In the Asgaard Project, we develop methods to support the design and execution of plans defined in Asbru[1].

In Asbru, it is possible to specify actions as well as conditions based on durations: a condition might only become true if a parameter is above a certain value for a certain amount of time, or a plan (we call the parts of a protocol a plan) might be restricted to start within a certain interval, end within another given interval, and also be confined to a certain minimum and maximum duration.

Asbru itself, however, is a language with a LISP-like syntax that is impossible to use by physicians (see Figure 1). So we needed a visualization and user interface to enable physicians to work with our tools and methods. This Visualization is called AsbruView. It is based on utilizing metaphors to make Asbru's concepts easier to understand and use.

This paper is organized as follows: In section 2, we give a short introduction to Asbru. AsbruView is described in section 3. A comparison of AsbruView and Gantt and PERT charts is given in section 4. The results of a usability study of our AsbruView prototype are shortly discussed in section 5.

2. Asbru Basics

A few of Asbru's basic concepts will be presented here briefly. For a more in-depth discussion, please see [6, 7].

[1] Many of the names in the Asgaard project are based on names from Norse mythology. Asbru (also known as Bifrost), for example, is the rainbow bridge that leads to Asgaard, the home of the gods.

Asbru allows the definition of plans that use other plans (similar to function calls in programming languages), which are called sub-plans. Sub-plans are also defined in Asbru, and can be used in many different plans. If a plan has a sub-plan that is not defined elsewhere, that sub-plan is considered an action (to be performed by medical staff). This hierarchical decomposition is an important concept for making knowledge reusable, and also for structuring that knowledge.

A plan's execution is controlled by a number of conditions. It is important to note that all conditions are defined for certain time spans, unlike conditions in programming languages (e.g., a condition becomes true when a patient's glucose level has been too high for three consecutive days).

Asbru also contains powerful means of data abstraction, so that conditions can be triggered not only by values, but also by their rate of change, etc.

In addition to sub-plans and conditions, each Asbru plan contains preferences, intentions, and effects. These components are important for critiquing, etc., but are beyond the scope of this paper.

Every plan has a type that specifies the order in which its sub-plans must be performed, and whether all sub-plans must complete successfully in order for the whole plan to be successful. If not all sub-plans of a plan have to be executed, the *continuation condition* lists the compulsory plans. A plan can contain a specification of the time a sub-plan may take (using time annotations, see below). Asbru's plan types are listed in Table 1.

One of the central ideas behind Asbru is that of the Time Annotation. A Time Annotation consists of seven parts, any subset of which (except the reference point) can be left undefined (in this list, *action* also means other types of events, like conditions, intentions, etc.):

Reference Point. This is the point that all the other points in time are defined relative to. It can be an abstract point in time (e.g., *conception*).

Earliest Starting Shift (ESS). The smallest offset from the reference point when the action can start.

Latest Starting Shift (LSS). The latest point in time when the action must start.

Earliest Finishing Shift (EFS). The earliest point in time when the action can end.

Latest Finishing Shift (LFS). The greatest offset from the reference point when the action must end.

Minimum Duration (MinDu). The minimum amount of time the action or condition must last. This is not necessarily identical with the interval between LSS and EFS. It is bounded, however, by this difference (it can not be shorter) and the maximum duration.

Maximum Duration (MaxDu). The maximum duration that the condition or action may last. It is bounded by the difference between LFS and ESS, and the minimum duration.

3. Two Views

We started out with one visualization to capture all aspects of Asbru [5]. This proved to be unusable however, so we decided to provide two views: one that just contains topological information (and that shows which conditions are already defined), and one that gives detailed information about the temporal extents of plans, conditions, etc. [3, 4]

The two views AsbruView now consists of (and any that will be added eventually; for example one based on [11], which we are currently working on) show the same data — but from different "angles".

To make this easier to understand, plans are drawn using the same color in all views. It is also possible to show more or less detail by showing or hiding a plan's sub-plans. This is also synchronized, so that the same amount of information is visible in both views.

A separate control bar (on the right hand side, see Figure 4) is used to perform actions that affect the underlying data that is shown in both views, like creating or deleting a plan, or changing its name.

3.1. Topological View

Topological View is the more "interactive" view, because not only are plans shown here (with the plans they contain, i.e. the whole hierarchy of plans), they can also be moved. If a plan is moved by the user, it can change its position within its current super-plan, or be moved to different levels within the hierarchy. A simple but effective method was used to allow three-dimensional navigation on the two-dimensional screen [3].

Plan Layouts

AsbruView's Topological View stems from the remains of the original design ideas [5]. We replaced the "exact" time axis by an axis that does not have a scale on it, but that is merely meant as a "direction".

Thus, it is possible to express relations between plans, like "*A happens after B*" or "*A starts at the same time as B*" (see Table 1) by simply laying the corresponding objects out along (or parallel to) the time axis.

The central metaphor in this view is that of a running track (Figure 2). A plan is represented by such a track, which the physician is considered to run along while treating the patient.

```
(PLAN I-RDS-Therapy
  ...
  (DO-ALL-SEQUENTIALLY
    (initial-phase)
    (one-of-controlled-ventilation)
    (weaning)
    (one-of-cpap-extubation)
  )
)

(PLAN one-of-controlled-ventilation
  ...
  (DO-SOME-ANY-ORDER
    (controlled-ventilation)
    (permissive-hypercapnia)
    (crisis-management)
    CONTINUATION-CONDITION controlled-ventilation
  )
)

(PLAN controlled-ventilation
  (PREFERENCES (SELECT-METHOD BEST-FIT))
  (INTENTION:INTERMEDIATE-STATE (MAINTAIN STATE(BG) NORMAL controlled-ventilation *))
  (INTENTION:INTERMEDIATE-ACTION (MAINTAIN STATE(RESPIRATOR-SETTING) LOW controlled-ventilation *))
  (SETUP-PRECONDITIONS (PIP (<= 30) I-RDS *now*)
    (BG available I-RDS [[_, _], [_, _], [1 MIN,_] (ACTIVATED initial-phase-1#)]))
  (ACTIVATED-CONDITIONS AUTOMATIC)
  (ABORT-CONDITIONS ACTIVATED
    (OR (PIP (> 30) controlled-ventilation [[_, _], [_, _], [30 SEC, _], *self*])
      (RATE(BG) TOO-STEEP controlled-ventilation [[_, _], [_, _], [30 SEC,_], *self*])))
  (SAMPLING-FREQUENCY 10 SEC)
  (COMPLETE-CONDITIONS
    (FiO2 (<= 50) controlled-ventilation [[_, _], [_, _], [180 MIN, _], *self*])
    (PIP (<= 23) controlled-ventilation  [[_, _], [_, _], [180 MIN, _], *self*])
    (f (<= 60) controlled-ventilation [[_, _], [_, _], [180 MIN, _], *self*])
    (state(patient) (NOT DYSPNEIC) controlled-ventilation [[_, _], [_, _], [180 MIN, _], *self*])
    (STATE(BG) (OR NORMAL ABOVE-NORMAL) controlled-ventilation
       [[_, _], [_, _], [180 MIN,_], *self*])
    (SAMPLING-FREQUENCY 10 MIN))

  (DO-ALL-SEQUENTIALLY
    (one-of-increase-decrease-ventilation)
    (observing))
)
```

Figure 1. An example of Asbru code (part of a clinical treatment protocol for Infants' Respiratory Distress Syndrome (I-RDS)).

	All plans must complete to continue	**Some plans must complete to continue**
Start together	*Do-All-Together* Plans (no continuation-condition, all plans must complete)	*Some-Together* Plans (continuation-condition specified as subset of plans)
Execute in any order	*Do-All-Any-Order* Plans (no continuation-condition, all plans must complete)	*Some-Any-Order* Plans (continuation-condition specified as subset of plans)
Execute in total order (sequence)	*Do-All-Sequentially* Plans (no continuation-condition, all plans must complete)	*Some-Sequentially* Plans (continuation-condition specified as subset of plans)

Table 1. Plan Types in Asbru.

Tracks can be stacked on top of each other (hierarchical decomposition), and laid out in different ways (Figure 4). The way these objects are laid out is determined by the type of the plan containing them (Table 1).

Conditions

Another important function of this view is to show which conditions have been defined. To do this, we use metaphors from traffic control, which also make the underlying concepts easier to grasp (Figure 2).

The first traffic sign the runner encounters is a "no entrance with exceptions" sign, which is used for the filter precondition. The traffic sign means that nobody may enter the road this sign is put next to, except people who are listed on a small additional sign — which is what the filter precondition stands for: a condition that must be fulfilled in order to start the plan.

A barrier is used as a metaphor for the setup precondition. If this condition is not fulfilled, the barrier is closed; but it opens as soon as the patient meets the criteria — so in contrast to the filter precondition, this condition can be achieved through therapeutic actions.

A traffic light is used for three more conditions: The red light stands for the abort condition (in this metaphorical world, a red light never changes back to yellow or green).

The yellow light is used for the suspend condition. A yellow light means "Attention!", and this is in a way similar to the meaning of the suspend condition (where an emergency plan may be performed while this plan is suspended). And it is also easy to understand that often a plan will first be suspended, and if the emergency plan does not work, it will ultimately be aborted from the suspended state.

The green light symbolizes the reactivate condition. When after a suspension the criteria for continuing the plan are met again, the green light signals that normal work can continue now.

The finishing flag that the runner must pass in order to win (or at least reach the goal) is used for the complete condition. This finishing flag is usually drawn semi-transparent in order not to obstruct the view to objects behind it.

All condition objects appear gray when the corresponding condition is not defined, and in color when it is (in general, not all conditions of a plan will be specified, because it inherits some of them from the plan using it).

3.2. Temporal View

This view is based on the concept of LifeLines [9, 10], which in turn is based on an old and quite powerful concept called Time Lines [13] — Gantt charts are another application of this concept.

We also make use of the idea of facets (also from [9, 10]) for different aspects of Asbru, like conditions, effects, etc. (i.e., each of these aspects has its own facet that can be opened or closed independently of the others, but that uses the same time scale).

The focus of this view is not so much the topology of plans (which is also visible, however), but the exact temporal dimensions of plans, conditions, etc.

Time Annotation Glyphs

LifeLines are very easy to understand, but are only useful for events whose exact points in time are known (i.e., past events). To use them for future events, we had to design a glyph more complex than a simple bar, that would capture all the (uncertain and possibly undefined) parts of a Time Annotation.

A glyph is a graphical object (often vaguely representing a real object, like a face) whose features express the values of certain attributes that are to be shown [1, 2]. A glyph must be distinguished from a metaphor, which usually more closely represents a real object (like the traffic signs used in Topological View), and whose features do not change with values it represents. Our glyph (Figure 3) is also based on a simple metaphor. The four starting and ending times are drawn as little supports for the "MaxDu bar" lying on them. On top of that bar, supported by diamonds (LSS, EFS), rests the "MinDu bar". Little arrows point towards the ESS, LSS, EFS and LFS lines so that it is possible to recognize a time annotation in which the EFS appears before the LSS (which is possible if an action has to be performed within an interval that is at least twice as long as the duration of that action).

The minimum duration cannot be shorter than the difference between the LSS and EFS, otherwise its bar would fall down between the supports. The same is true for the maximum duration, of course.

Any undefined parts are displayed in gray, like undefined conditions in Topological View (section 3.1).

If the LSS or EFS are not defined, the diamonds supporting the MinDu bar become rolls, indicating that they might move. If, for example, the LSS and MinDu are defined (see Figure 3, lower left), then the EFS is implicitly also defined, and moves whenever one of the other two values is changed.

The glyph also communicates information about the granularity (time unit) that was used to enter the data relative to the one it is displayed in (this has not been implemented in the prototype yet, however). If the same granularity is used, all parts have sharp edges. If they are different, and the granularity of the current display is higher (coarser), a circle is drawn instead (see Figure 3, upper right), in analogy to open intervals in mathematics (e.g., "from 0 to, but not including, 1"). If the current granularity is finer, a zigzag pattern is drawn that covers the area of uncertainty (Figure 3, lower right).

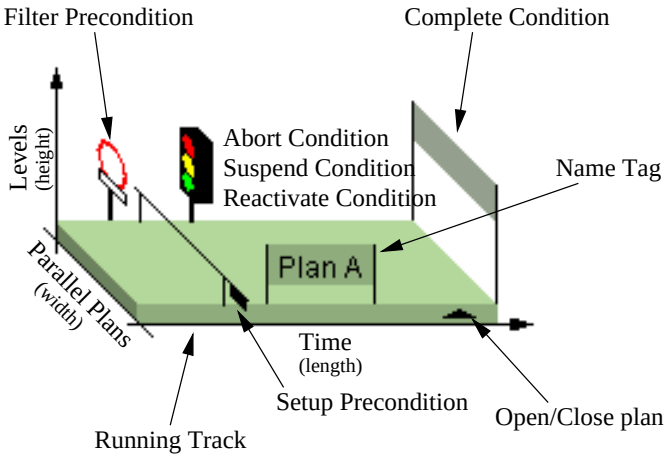

Figure 2. Anatomy of an AsbruView Plan in Topological View.

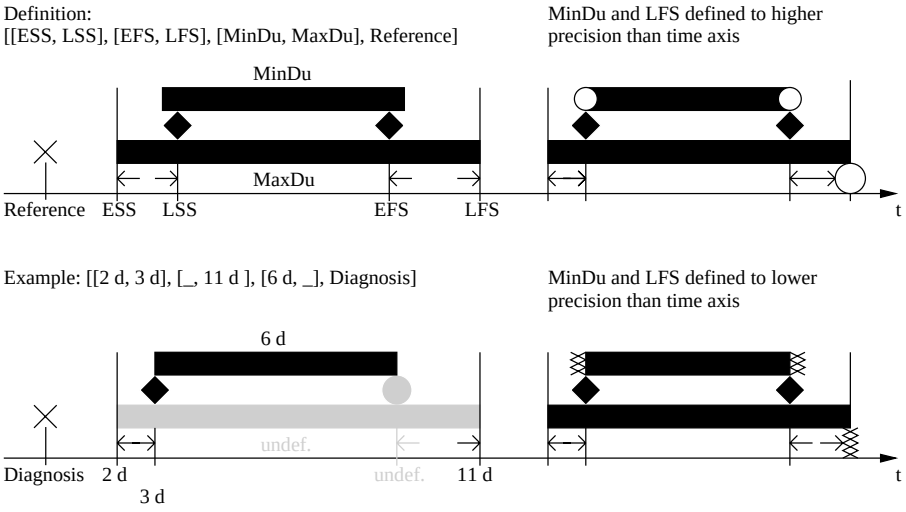

Figure 3. Time Annotations, outline (adapted from [4]).

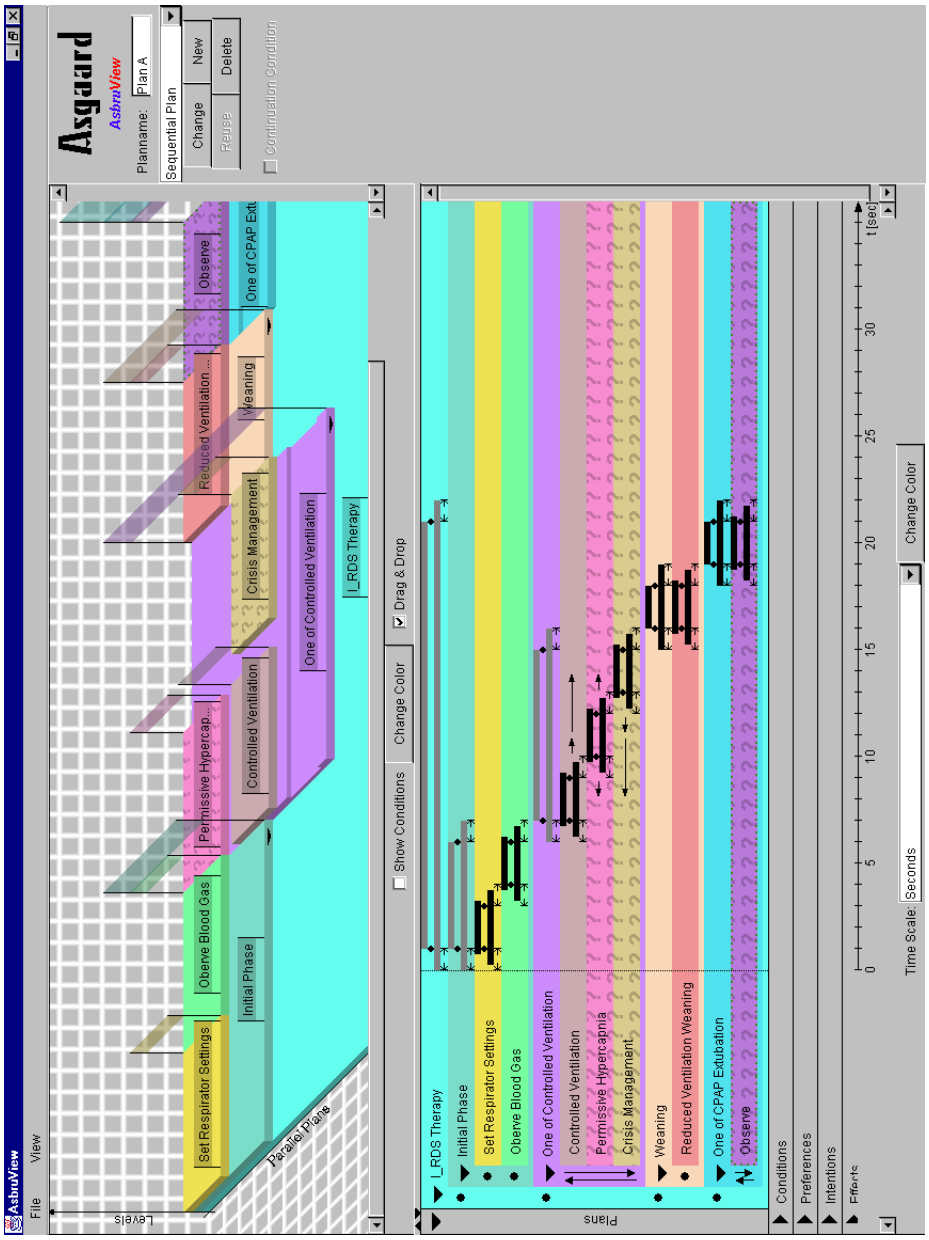

Figure 4. A screen-shot of the AsbruView prototype showing a part of a plan for ventilation of new-born infants. Topological View is in the upper part, Temporal View in the lower part.

Plan Tree

The topology of plans is also visible in the left part of Temporal View (see Figure 4), which is similar to tree views often used for displaying the contents of file systems. This tree could also be understood as Topological View seen from a bird's eye view (not in the literal sense, but there are parallels).

We use symbols to indicate different plan types (inspired by block structured diagrams used in programming, [8]).

Each line of the tree view shows the name of one plan. This vertical partition is colored using the plan's color, and used for showing its time annotation glyph.

4. AsbruView vs. Gantt and PERT Charts

The two views introduced above bear many similarities to Gantt (Figure 5) and PERT (Figure 6) charts. The two figures mentioned show the same plans as specified in Asbru syntax in Figure 1 and shown in the AsbruView screen-shot in Figure 4. Because we do not know any actual bounds for the durations of the plans in this example, the durations given in Temporal View are actually not correct. Temporal view does, however, provide the possibility of representing any-order plans etc. that Gantt and PERT charts do not.

But AsbruView is better suited for application in medical therapy planning (especially using Asbru) for the reasons listed below. This comparison does not only compare AsbruView to the other types of charts, but also contains a few references to Asbru features. This is hard to avoid, because of the close connection between Asbru and AsbruView.

Information Density. It communicates more information than Gantt and PERT charts, without at the same time cluttering the display. A plan's type, for example, is visible from the way its sub-plans are arranged (Topological View), or from symbols (Temporal View), so no arrows linking plans are necessary. Arrows are needed in Gantt as well as in PERT charts, but in neither of AsbruView's views (at least not for showing the sequence of plans — arrows are used in Temporal View to indicate that a plan that is used in an *any-order* plan can be performed at different positions within the set of sub-plans).

Control Flow. Topological View can be seen as AsbruView's counterpart of the PERT chart. Topological information is displayed in both diagrams, but in AsbruView, plans are arranged along axes to make their topology easier to understand at one glance. Topological View better shows the flow in the sense of "direction".

Topological View also contains information about which conditions are defined, which is another information PERT charts are missing (because that information simply does not play a role in these diagrams).

Temporal Uncertainty. While setting deadlines plays an important role in project management (see below), it does not in therapy planning. But a therapy or a sub-plan might be restricted in its maximum duration, or the latest point in time when it has to start. Such information is also important in conditions, etc. This (complex) temporal information is communicated by AsbruView's Time Annotation glyphs.

Knowledge Roles. In Asbru, a number of so-called knowledge roles can be defined: conditions, preferences, intentions, and effects. All of them have a temporal dimension, which can be displayed using AsbruView.

Optional Plans. Optional plans, which play a big role in Asbru, are displayed using a question-mark texture — they cannot be displayed in Gantt or PERT charts. This is also true for any-order-plans — in Gantt and PERT charts, the order of tasks has to be known.

Level of Detail. Temporal View as well as Topological View allow the user to select the amount of information displayed, thus making it possible to both browse through a plan quickly, and to see very detailed information. This is another feature most implementations of PERT charts lack (at least those we know).

Granularity of Time. AsbruView's Time Annotation glyphs also make it possible to see to which granularity a point in time was defined, which is another information not visible in typical Gantt charts. If the user views a Gantt chart at a high resolution (i.e., seconds), an event specified at an accuracy of an hour should not appear to be more accurate than it really is (in physics, such a specification usually implies an uncertainty of plus or minus one half base unit).

There are two concepts that are important in project management, but not in Asbru(View):

Milestones. The concept of a milestone does not play the same role in therapy planning as in project management. A milestone in a therapy cannot be specified by its date, only by conditions the patient must fulfill. Temporal constraints can play a role here, as well, but are only of secondary importance, and if they are not met, different methods are used and new milestones are introduced. The strict adherence to exact dates necessary in project management is not of the same importance in therapy planning.

A kind of milestone is typically implemented by splitting a treatment up into several phases, so implicit milestones exist at the end of every phase.

Critical Path. The main application of PERT charts is to find the critical path through a project (i.e., the set of subprojects which delay the end of the whole project if they are delayed). Because the duration of a treatment does not play a role in treatments from a medical point of view, it is not covered in Asbru (and neither are other administrative tasks that are important for the operation of a hospital — Asbru is only concerned with the work of the physicians).

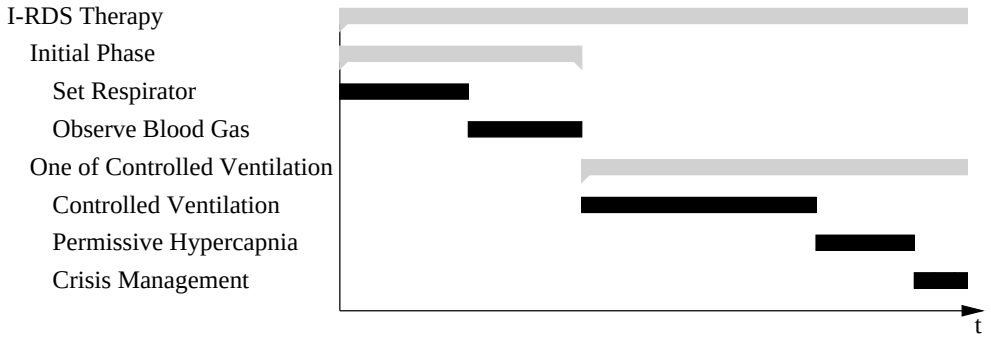

Figure 5. Gantt Chart example. Black Boxes represent basic (atomic) actions, gray boxes group actions into higher level tasks. This figure depicts the same plan as Figure 4 — the *any-order* **plan** *One Of Controlled Ventilation* **can not be displayed properly, neither can the fact that the bottom-most two plans shown are optional**

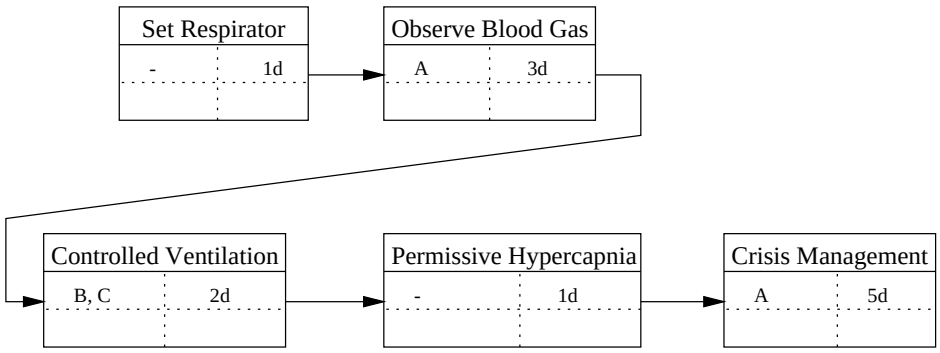

Figure 6. PERT Chart example. Each element contains its name, duration, predecessor(s), and start and end dates. As in Figure 5, optional and any-order plans are not visible.

5. Usability

We implemented a prototype of AsbruView (Figure 4), which we used to perform a usability study with six physicians [3]. In this study, we first explained the system (and the underlying design ideas of the Asbru language) to them and then asked them to design a plan from their everyday work. We used questionnaires based on [12] before and after the test to the participants' impression of the system.

All participants found the metaphors easy to understand, and remembered the different plan types after only one explanation. Most participants said they could understand the topology of a plan from the graphical depiction. Hierarchical decomposition and the depiction of optional and cyclical plans was found easy to understand by all participants. Manipulation of plans was judged good, okay or bad by equal amounts of participants. Manipulation suffered very much from the lack of speed of the implementation. All participants criticized the speed of the system — this was mostly due to its implementation in Java, but also because the test environment was not always perfect.

The overall impression of most participants was good, only one participant judged it okay.

6. Conclusion and Future Work

Medical therapy planning might look similar to project management, but there are a few important differences. We tried to cover them in the design of Asbru and AsbruView, and so can provide a tool that is better suited for this task.

AsbruView seems to be quite usable for the target group (physicians), which was, of course, the main reason for developing it.

We believe that users can work with complex temporal information, even if they have little or no formal education in logic or computer science, if they are provided with the right tools.

We are now working on implementing other features of Asbru in AsbruView, like intentions, preferences, etc. We also have to find a way to represent cyclical plans — the way this is done now is not satisfactory.

7. Acknowledgements

We would like to thank Dr. Shahram Adel, Dr. Sophie Brandstetter, Dr. Maria Dobner, Dr. Gerhard Miksch, Primarius Dr. Franz Paky, and ao. Prof. Dr. Christian Popow for taking part in the evaluation.

The Asgaard Project is supported by "Fonds zur Förderung der wissenschaftlichen Forschung" (Austrian Science Fund), grant P12797-INF.

References

[1] H. Chernoff. The use of faces to represent points in k-dimensional space graphically. *Journal of the American Statistical Association*, 68:361–368, 1973.

[2] M. C. Chuah and S. G. Eick. Glyphs for software visualization. In *5th International Workshop on Program Comprehension (IWPC '97) Proceedings*, pages 183–191. IEEE Computer Society Press, Dearborn, Michigan, May 1997.

[3] R. Kosara. Metaphors of Movement — A User Interface for Manipulating Time-Oriented, Skeletal Plans. Master's thesis, Vienna University of Technology, Vienna, Austria, May 1999.

[4] R. Kosara and S. Miksch. Visualization Techniques for Time-Oriented, Skeletal Plans in Medical Therapy Planning. In W. Horn, Y. Shahar, G. Lindberg, S. Andreassen, and J. Wyatt, editors, *Proceedings of the Joint European Conference on Artificial Intelligence in Medicine and Medical Decision Making (AIMDM'99)*, pages 291–300, Aalborg, Denmark, June 1999. Springer Verlag.

[5] R. Kosara, S. Miksch, Y. Shahar, and P. Johnson. AsbruView: Capturing Complex, Time-oriented Plans — Beyond Flow-Charts. In *Second Workshop on Thinking with Diagrams 1998 (TwD98)*, pages 119–126. University of Wales, Aberystwyth, UK, Aug. 22–23 1998.

[6] S. Miksch, Y. Shahar, W. Horn, C. Popow, F. Paky, and P. Johnson. Time-oriented skeletal plans: Support to design and execution. In *Fourth European Conference on Planning (ECP'97)*. Springer, September 24–26 1997.

[7] S. Miksch, Y. Shahar, and P. Johnson. Asbru: A task-specific, intention-based, and time-oriented language for representing skeletal plans. In *Proceedings of the 7th Workshop on Knowledge Engineering: Methods & Languages (KEML-97)*. Milton Keynes, UK, Open University, 1997.

[8] I. Nassi and B. Shneiderman. Flowchart techniques for structured programming. *SIGPLAN Notices*, 8(8):12–26, 1973.

[9] C. Plaisant, B. Milash, A. Rose, S. Widoff, and B. Shneiderman. Lifelines: Visualizing personal histories. In *Proceedings of ACM CHI 96 Conference on Human Factors in Computing Systems*, volume 1 of *PAPERS: Interactive Information Retrieval*, pages 221–227, 1996.

[10] C. Plaisant, R. Mushlin, A. Snyder, J. Li, D. Heller, and B. Shneiderman. Lifelines: Using visualization to enhance navigation and analysis of patient records. In *Proceedings of the 1998 American Medical Informatic Association Annual Fall Symposium*, pages 76–80, Nov. 9–11 1998.

[11] J.-F. Rit. Propagating temporal constraints for scheduling. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 383–388. Morgan Kaufman Publishers, Inc., August 11-15 1986.

[12] B. Shneiderman. *Designing the User Interface: Strategies for Effective Human-Computer-Interaction*. Addison Wesley Longman, 3rd edition, 1997.

[13] E. R. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, Cheshire, CT, 1983.