ELSEVIER

Artificial Intelligence in Medicine 22 (2001) 111–131

Artificial Intelligence in Medicine

www.elsevier.com/locate/artmed

# Metaphors of movement: a visualization and user interface for time-oriented, skeletal plans

Robert Kosara*, Silvia Miksch[1]

*Institute of Software Technology (IFS), Vienna University of Technology, Favoritenstrasse 9-11/E188, A-1040 Vienna, Austria*

Received 3 March 2000; received in revised form 20 August 2000; accepted 13 November 2000

**Abstract**

Therapy planning plays an increasingly important role in the everyday work of physicians. Clinical protocols or guidelines are typically represented using flow-charts, decision tables, or plain text. These representations are badly suited, however, for complex medical procedures.

One representation method that overcomes these problems is the language Asbru. But because Asbru has a LISP-like syntax (and also incorporates many concepts from computer science), it is not suitable for physicians.

Therefore, we developed a visualization and user interface to deal with treatment plans expressed in Asbru. We use graphical metaphors to make the underlying concepts easier to grasp, employ glyphs to communicate complex temporal information and colors to make it possible to understand the connection between the two views (Topological View and Temporal View) available in the system.

In this paper, we present the design ideas behind AsbruView, and discuss its usefulness based on the results of a usability study we performed with six physicians. 

*Keywords:* Information visualization; Therapy planning; Time-oriented skeletal plans; Graphical metaphors

## 1. Introduction — what is medical therapy planning?

Physicians are faced with the task of improving the quality of health care while at the same time reducing the costs of treatment. One way to do this is *clinical protocols* or

* Corresponding author. Tel.: +43-1-58801-18833;
fax: +43-1-58801-18899; URL: http://www.ifs.tuwien.ac.at/~rkosara.
*E-mail addresses*: rkosara@ifs.tuwien.ac.at (R. Kosara), silvia@ifs.tuwien.ac.at (S. Miksch).
[1] URL: http://www.ifs.tuwien.ac.at/~silvia.

0933-3657/01/$ – see front matter 
PII: S0933-3657(00)00103-2

*guidelines* that describe general procedures of treatment, but have to be adapted to the needs of a particular patient.

Authoring clinical protocols is a non-trivial task. Mostly, these protocols are expressed in natural language or flow diagrams [7,19], but these kinds of representation cannot easily be transformed into a formal and structured framework [9]. The benefits of existing representations are: (1) writing in free text is easy; (2) medical experts are used to work with free text or flow diagrams; (3) flow diagrams are useful for representing sequential states and actions in a graphical way.

From the point of view of computer science, however, these representations have significant limitations: (1) existing clinical protocols are partly vague concerning their intentions and their temporal, context-dependent representations; (2) the variability of clinical protocols is hard to represent in a structured way (e.g. a medical goal can be achieved by different therapeutic actions); (3) it is quite difficult to cope with all possible orders of plan execution and all the exception conditions that might arise; (4) it is hard to represent concurrent actions, different temporal dimensions, high numbers of possible transitions, and mutual dependencies of parameters in an easy to comprehend way.

Hardly any of the existing protocols are formulated in an appropriate way that would facilitate computer support. This is why we set out to develop a framework to make the design and execution of treatment plans easier. Making this system usable for physicians is an important task, a part of which is described in this paper.

In Section 2, we introduce the plan representation language Asbru. The visualization challenges Asbru poses are described in Section 3. Our answer to these challenges, AsbruView, is introduced in Section 4. We discuss the findings of a small study we did to assess AsbruView's usability in Section 5.

## 2. Introduction to Asbru

Asbru [20,21] is a plan representation language that is used in the Asgaard Project[2] to represent clinical guidelines as time-oriented, skeletal plans. It can be used to express clinical protocols as skeletal plans [5] that can be instantiated for every patient (for an example see Fig. 1).

In Asbru, the following parts of a plan can be specified: *preferences*, *intentions*, *conditions*, *effects*, and *plan body (actions)*.

### 2.1. Preferences

Preferences constrain the applicability of a plan (e.g. select-criteria: exact-fit, roughly-fit) and express the kind of behavior of the plan (e.g. kind of strategy: aggressive or normal).

[2] In Norse mythology, *Asgaard* was the home of the gods. It was located in the heavens and was accessible only over the rainbow bridge, called *Asbru* (or *Bifrost*) (for more information about the *Asgaard* project, see http://www.ifs.tuwien.ac.at/asgaard/).

```
(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-
entilation *))
  (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))
)
```

Fig. 1. An example of Asbru code (part of a clinical treatment protocol for infants' respiratory distress syndrome (I-RDS)).

### 2.2. Intentions

Intentions are high-level goals that should be achieved by a plan, or maintained or avoided during its execution. This information is very important not only for selecting the right plan, but also for critiquing treatment plans as part of the ever ongoing process of improving the treatment. This makes intentions one of the key parts of Asbru.

### 2.3. Conditions

Conditions need to hold in order for a plan to be *started*, *suspended*, *reactivated*, *aborted*, or *completed*. Two different kinds of conditions (called preconditions) exist, that must be true in order for a plan to be started: filter-preconditions cannot be achieved through treatment (e.g. subject is female), setup-preconditions can. After a plan has been started, it can be suspended (interrupted) until either the restart-condition is true (whereupon it is continued at the point where it was suspended) or it has to be aborted. If a plan is aborted, it has failed to reach its goals. If a plan completes, it has reached its goals, and the next plan in the sequence is to be executed.

### 2.4. Effects

Effects describe the relationship between plan arguments and measurable parameters by means of mathematical functions. A probability of occurrence is also given.

### 2.5. Plan body (actions)

The plan body contains plans or actions that are to be performed if the preconditions hold. A plan is composed of other plans, which must be performed according to the plan's type (Table 1): in sequence, in any order, in parallel, or periodically (as long as a condition holds, a maximum number of times, and with a minimum interval between retries).

A plan is decomposed into sub-plans until a non-decomposable plan — called an action or a user-performed plan — is found. All the sub-plans consist of the same components as the plan, namely: preferences, intentions, conditions, effects, and the plan body itself.

Plans are executed (i.e. their parameters monitored, conditions checked and reacted to) by an execution unit. User-performed plans are displayed to the user so that he or she can react and then tell the machine if and when the action is finished and if it was successful.

### 2.6. Time annotations

An important part in specifying the complex temporal aspects of plans are Time Annotations. A Time Annotation specifies four points in time relative to a reference point (which can be a specific or abstract point in time, or a state transition of a plan): the earliest starting shift (ESS), latest starting shift (LSS), earliest finishing shift (EFS) and latest finishing shift (LFS). Two durations can also be defined: the minimum duration (MinDu) and maximum duration (MaxDu). Together, these data specify the temporal constraints

Table 1
Plan types in Asbru

| | All plans must complete to continue | Some plans must complete to continue |
|---|---|---|
| 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) |
| 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) |

within which an action must take place, or a condition must be fulfilled for a condition to trigger.

## 3. Visualization challenges — state of the art

Asbru's structure poses a number of difficult problems when trying to graphically represent plans written in that language. These problems are discussed in this section together with possible solutions (or approaches).

Information Visualization is often concerned with making huge amounts of data easy to understand. A plan visualization has to deal with smaller amounts of information, but with more complex structures.

An enormous amount of work has been done in the field of scientific and information visualization in the last few years, but most of these approaches focus on large amounts of multi-dimensional data. For this kind of problem, a number of good visualizations exist now, that make data accessible [8,10,22]. An overview of the state of the art can be found in [28].

The specific combination of problems faced here, however, has apparently never before been investigated. Solutions (or at least basic approaches) exist only for parts of the problem.

### *3.1. Hierarchical decomposition*

The connection between a plan and its sub-plans must be made clear, i.e. that a plan is made up of its sub-plans. The difficulty here does not so much lie in this problem alone, but in the fact that it must be communicated together with the other concepts described in this section. Any kind of tree view, like it is now used in many programs (especially file managers), could be used. A special method for this kind of information can be found in [33], which uses the hierarchical structure of a file system to partition a rectangular area into parts whose relative sizes correspond to the relative sizes of the files. A very interesting way of visualizing hierarchies in 3D are cone trees [30], which are trees where the child nodes of each node are placed around a circle, with the diameter of the circle decreasing with each level. Any node's contents can be rotated so that all the information is accessible.

These methods have two drawbacks for our problem, however: They do not contain any information other than the structure, and thus only provide links to it. And they do not allow for the visualization of topological relations between the children, i.e. if they are executed in parallel, in sequence, etc.

Ways of displaying more *and* less detailed information in the same view at the same time are distortion techniques, like fish-eye views [6], the stretchable rubber sheet [31], and the perspective wall [18] (a very good overview can be found in [16]). These techniques display a part of the information at its normal size and scale the remaining data according to its 'distance'. This way, detailed data can be viewed without getting lost because no or little context information is available.

A similar idea is to use a logarithmic time-scale and display cruder information the cruder the time scale gets. This way, an enormous amount of patient information can be displayed on just one sheet of paper [27].

PERT charts are not useful in this case, because they do not provide a connection between the topology of a plan and its temporal extent.

### 3.2. Plan types

A plan's type (see Table 1 and Section 2.5) should be easy to tell from its graphical representation, especially if it has sub-plans — but also if it has not. This is a contradiction to the previous definition of a plan — which either has sub-plans or is an action (which, by definition, has no type. The type only specifies the way its sub-plans are to be performed). For practical work with plans, however, users might want to define a plan's type before any sub-plans are added to it. Also, whether a plan is optional (i.e. not part of the continuation condition of its super-plan) must be visible.

### 3.3. Temporal order ('topology')

The way a plan's sub-plans are to be performed must be communicated by the visualization.

Flow-charts [7,19] are usually used for this purpose (order of execution), but they do not cover parallel plans or sets of plans that can be performed in any order (the latter is possible,[3] but only with considerable effort that leads to diagrams that are impossible to read — which definitely is not what flow-charts were intended for). Additionally, flow-charts scale very poorly, i.e. become unreadable when a large number of plans is defined, and they do not cover the temporal aspect (see below).

For *any-order* plans (see Table 1), only the set of plans to be used is known, but not the order in which they will be performed. A way of depicting a plan has to be found where the order in which they are depicted does not necessarily correspond to the order in which they will be executed.

Another interesting idea is the way railroad schedules are drawn for the Japanese Shinkansen trains [38]. On such a diagram, there is a horizontal time axis, and the train

[3] By defining one path for every possible permutation of the plans. For $n$ plans, this means $n!$ different paths.

stations are put one above the other on the vertical axis. A line is drawn for every train that connects the points in time when the train stops at a station. This way, a large number of trains can be drawn on one diagram without sacrificing readability. It is also easy to see connections between trains. But any-order plans are still hard to draw, and the duration of a plan (or even temporal uncertainty) is next to impossible to include in such a diagram.

### 3.4. Compulsory versus optional plans

A sub-plan can be used in two different ways: it either *must* be executed (compulsory plan) or it *can* be (optional). While a compulsory plan is easy to understand (and to depict), a way of indicating that a plan is optional is a lot more difficult, especially if it must be different from the representation of temporal uncertainty (see below). A blurred depiction of plans [17] therefore cannot be used.

### 3.5. Cyclical plans

Cyclical plans are the most difficult, because not only their duration and end times (see next section) can vary over a long time, the number of applications of their single sub-plan is not known, either.

We tried sphere and cylinder metaphors (inspired by [8]), but that did not lead to usable representations.

### 3.6. Temporal uncertainty

The time a plan takes, but also time spans that are considered for the relevance of symptoms are not defined in terms of exact durations (see Section 2.6).

A very easy-to-understand way of visualizing time are LifeLines [25,26], which in turn are based on an old and quite powerful concept called Time Lines [37] — Gantt charts are another application of this concept. Time Lines can only be used for time spans that are known, i.e. past events (they are used in many other applications as well, like [4,32]).

But a way of visualizing time spans, where only part of the information (e.g. the minimum duration) is known, must be found. This information may be refined later; this is called a *minimum-commitment approach* [36] (even though the term *late commitment* would be more appropriate here).

A related problem is that of temporal granularity. It should be possible to tell to what accuracy a point in time has been defined (e.g. seconds, minutes, etc.).

Ways of indicating uncertainty can be found in [17,36], which use lines that look like hand-drawn sketches or that seem to have blurred edges. These approaches only tell the viewer that the data is uncertain, but not to which degree.

A very versatile — albeit difficult to understand — solution to this problem can be found in [29]: Sets of possible occurrences ('SOPOs') are drawn on a diagram with two time axes. While the methodology proposed there is very powerful, it is very hard to understand and use, even for people from computer science.

A time annotation in Asbru consists of seven values, and thus can be understood as a point in seven-dimensional space. A very interesting approach to visualize this kind of data

are parallel coordinates [10,11]: The coordinate axes in such a diagram are not perpendicular to each other, but parallel. Thus, a point in $n$-dimensional space becomes a line connecting one point on each of the $n$ 'axes'. But because parallel coordinates do not clearly indicate the relations between the different quantities, they are not useful in this case (and are generally better suited for data in spaces with independent axes).

The most promising way of visualizing temporal uncertainty are glyphs [2,24], or Chernoff faces [1], which is the solution we finally used. A design that is quite similar to the one presented in Section 4.2.3 (but which was developed independently) can be found in [3]. Glyphs are graphical objects whose features reflect values, and therefore change their shape or size according to them. Glyphs can be combined with metaphors (or based on them), but are otherwise completely independent, and do not have to resemble a real object, or have any semantic relation with the object they look like (there is some confusion in the literature about this distinction).

## 4. AsbruView

From Fig. 1, it should be obvious that a plan representation language is not usable for physicians or other medical staff. This is why we developed AsbruView. AsbruView is a visualization and user interface for Asbru plans. It was designed to meet all the requirements sketched in the previous two sections.

It consists of two parts to serve as many needs as possible: Topological View and Temporal View. Topological View mainly displays the relationships between plans, without a precise time scale. Temporal View concentrates on the temporal dimension of plans and conditions. Topological View uses graphical metaphors, Temporal View uses glyphs to make the underlying concepts easier to understand.

### *4.1. Topological view*

AsbruView's Topological View stems from the remains of the original design ideas [14]. 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.

In this view, we make heavy use of graphical metaphors [15]. A metaphor supports comprehension of an unknown complex concept through a well-known one. Instead of using an abstract diagram or object (e.g. rectangles, growing and shrinking circles), we use signs from (more or less) daily life to communicate the various components of Asbru.

The central metaphor in this view is that of a running track (Fig. 2). A plan is represented by such a track, which the physician is considered to run along while treating the patient. Tracks can be stacked on top of each other (hierarchical decomposition), and laid out in different ways (Figs. 4 and 10). The way these objects are bayed out is determined by their type.

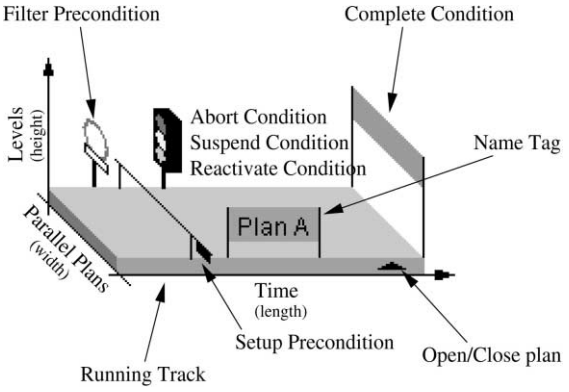


Fig. 2. Anatomy of an AsbruView plan in topological view.

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 (Fig. 2). Conditions are described in more detail in Section 4.1.7.

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).

*4.1.1. Hierarchical decomposition*

Plans are stacked on top of each other (*Levels* dimension, see previous section and Figs. 2 and 3) to depict the hierarchical decomposition of plans.

This decomposition at the same time provides a kind of abstraction hierarchy: a plan's sub-plans are more detailed than the containing plan. Thus, in order to get a better overview or to see more detail, one can hide or display the contents of a plan, respectively [34].

This is done by clicking on the small triangle on the right of the plan's front face. This triangle only appears when a plan has sub-plans, and then can be used to 'open' or 'close' the plan, i.e. show or hide, respectively, its sub-plans (Fig. 3).

*4.1.2. Plan types*

A plan's type is not indicated by any symbol, but can be told from the way its sub-plans are arranged — see next section (the different plan types are shown in Fig. 4).

*4.1.3. Temporal order ('topology')*

In this view, the temporal order is the only way a plan's type is indicated (except for cyclical plans, see Section 4.1.5). Fig. 4 shows examples of the different plan types/layouts.

*4.1.3.1. Sequential plan*. The sub-plans are put next to each other parallel to the time dimension, so that as soon as the metaphorical runner reaches the end of one sub-plan, she steps on the next one. This simple chronological order of actions is used in many other systems, and also in virtually every diagram that includes time.

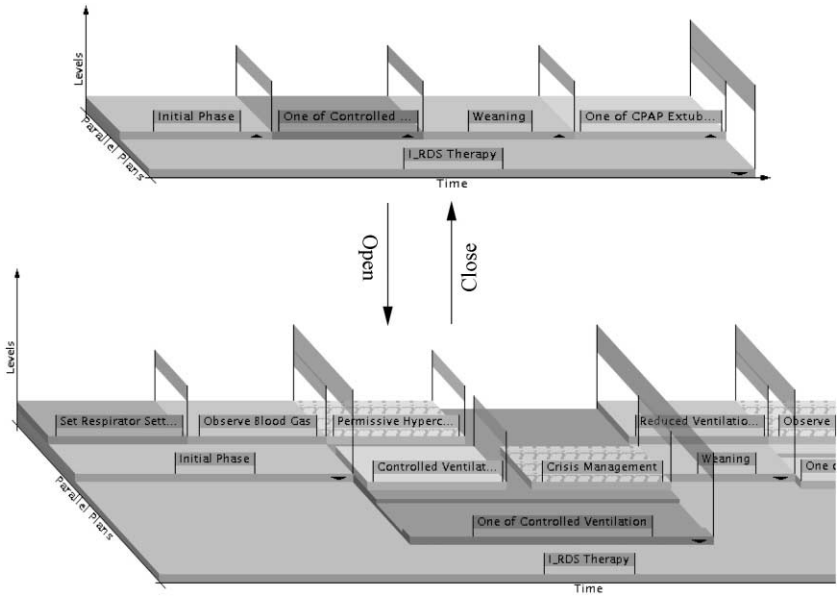


Fig. 3. Opening and closing of plans in Topological View. The plans in the upper image are closed, and thus show a higher level of abstraction. The small triangles on every plan indicate that they have sub-plans. The plans in the lower image are opened, revealing more detail, but making it harder to get an overview.

*4.1.3.2. Parallel plan, some-together plan*. In this case, the sub-plans are put next to each other along the parallel plans axis, and so have a common start time. In this view, they also have a common end time, but this is just an arbitrary decision. If the plans would have been given different lengths to illustrate the fact that they do not have to end at the same time, those lengths would have been as arbitrary (and probably more misleading — why is Plan B longer than Plan C?).

*4.1.3.3. Any-order plans*. Plans are put on the containing plan in a pattern that is meant to show that it has nothing to do with the real sequence of the plans. The fact that there is space between the plans is meant to add to the impression that the way the plans are put on their super-plan is arbitrary. The length of the containing plan is not — as is the case with sequential plans — the sum of the lengths of its sub-plans. This is done for esthetic reasons, but also adds to the perception of a random plan placement. The plans can be put in a groove at the front of the plan as soon as the order of application is clear.

*4.1.3.4. Cyclical plans*. This type of plan is described in detail in Section 4.1.5.

*4.1.4. Compulsory versus optional plans*

Optional plans are marked by a question-mark texture on the top face. An alternative would have been to draw optional plans semi-transparent, but this proved problematic when a number of plans (optional and compulsory) were stacked on top of each other.

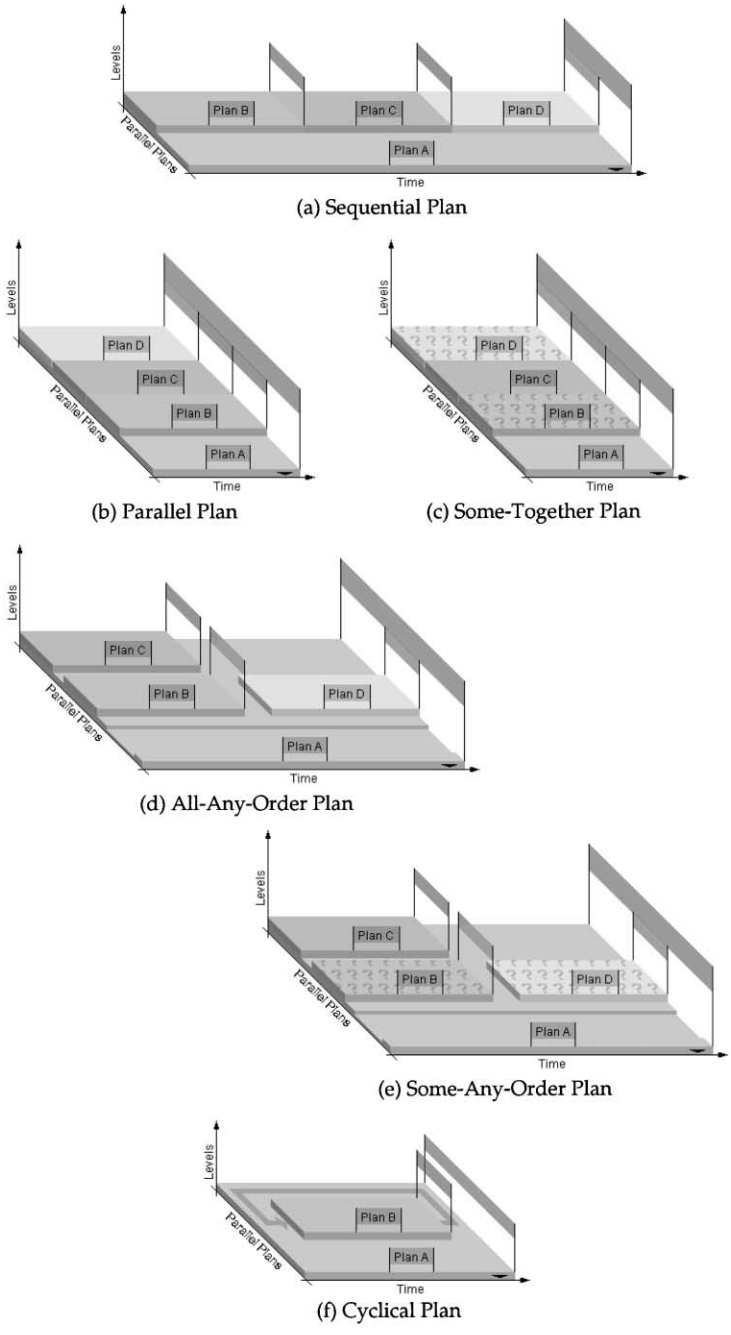


Fig. 4. All of Asbru's plan types in topological view.

Because the continuation condition (which decides about whether a plan is optional) is part of the containing plan, the optional mark is independent of whether or not the plan has sub-plans.

*4.1.5. Cyclical plans*

The single sub-plan of a cyclical plan is put on top of that plan, and an arrow is drawn from the end of the sub-plan to its beginning. In addition to the way this is done in the prototype, more information should be available in the Topological View, like the minimum or maximum number of retries. This can be done by putting a further flag on the arrow with a short description, like '<4' for a maximum of four tries (i.e. the arrow is used less than four times). This additional flag was not implemented in the prototype.

*4.1.6. Temporal uncertainty*

This aspect is not covered by the Topological View. Due to the perspective distortion, it is impossible to use a precise time scale, and draw plans so that they reflect temporal constraints correctly. The view to deal with this issue is the Temporal View described in Section 4.2.

*4.1.7. Conditions*

The expressions that make up the conditions are not shown in this view, but the information, which conditions are defined, is. Metaphors are used here as well (see Fig. 5). These come from the world of traffic control, which is not very closely related to track-and-field sports (where the running tracks come from). But both have to do with movement (and most people are familiar with the used symbols), so the connection should be easy to make.

These elements are drawn in gray when the corresponding condition is not defined, and in color otherwise (see Fig. 6).

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 very similar to the way the filter precondition works (see Section 2.3).

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.

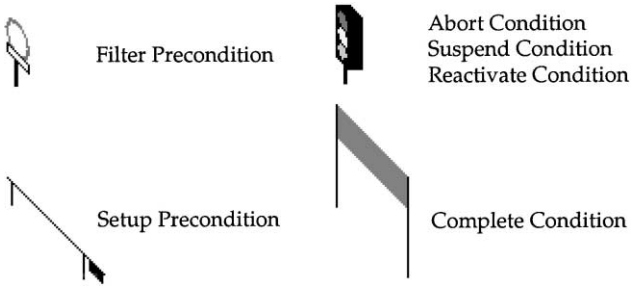


Fig. 5. Condition Metaphors (see Section 2.3 for details).

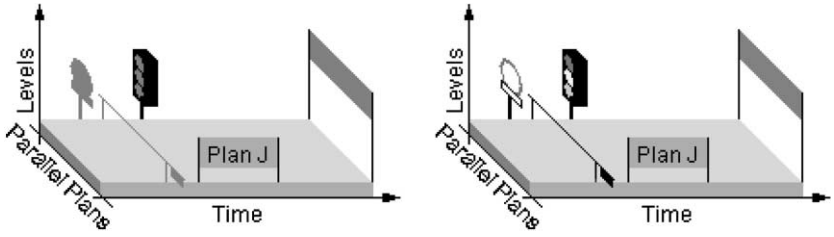


Fig. 6. Undefined (left) and defined conditions (right).

A traffic light is used for three 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. It is drawn with solid color (or solid gray) only when conditions are shown.

In the prototype, these condition metaphors only appear when the 'Show Conditions' check-box is checked (see Fig. 10), and then conditions are only shown for plans without visible sub-plans (i.e. plans without sub-plans or closed plans). This helps avoid cluttering the display with too much information (and also speeds up rendering).

### *4.2. Temporal view*

This view is based on time lines (see Section 3.6). We also make use of the idea of facets (also from [25,26]) 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.

#### *4.2.1. Hierarchical decomposition*

A plan's sub-plans are drawn a little further to the right, inside the rectangle of the containing plan. If this view is understood as a kind of bird's eye view of the Topological View, this is very similar to the depiction there, with sub-plans stacked on top of plans.

This leads to a tree structure that is quite similar to tree views that are now used in many programs for displaying hierarchies of data (directories, settings, etc.).

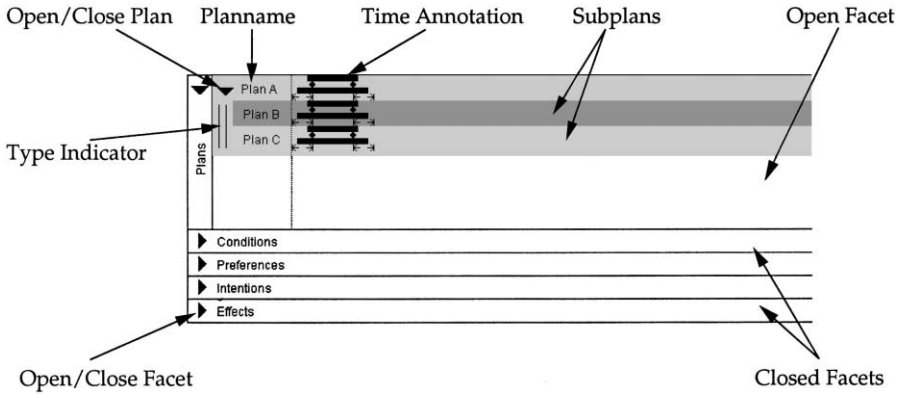


Fig. 7. Anatomy of a plan in temporal view.

*4.2.2. Plan types*

When a plan has sub-plans, its type is indicated by a small symbol left of its sub-plans (see Figs. 7 and 8). When it has no sub-plans, no such symbol is drawn (a plan's type mainly influences the order of execution of its sub-plans).

*4.2.3. 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 to use instead of a simple bar, that captures all the (uncertain and possibly undefined) parts of a Time Annotation (see Section 2.6).

Our glyph (Fig. 9) is also based on a simple metaphor, but is not a metaphor as such. 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 4.1).

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 Fig. 9, 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

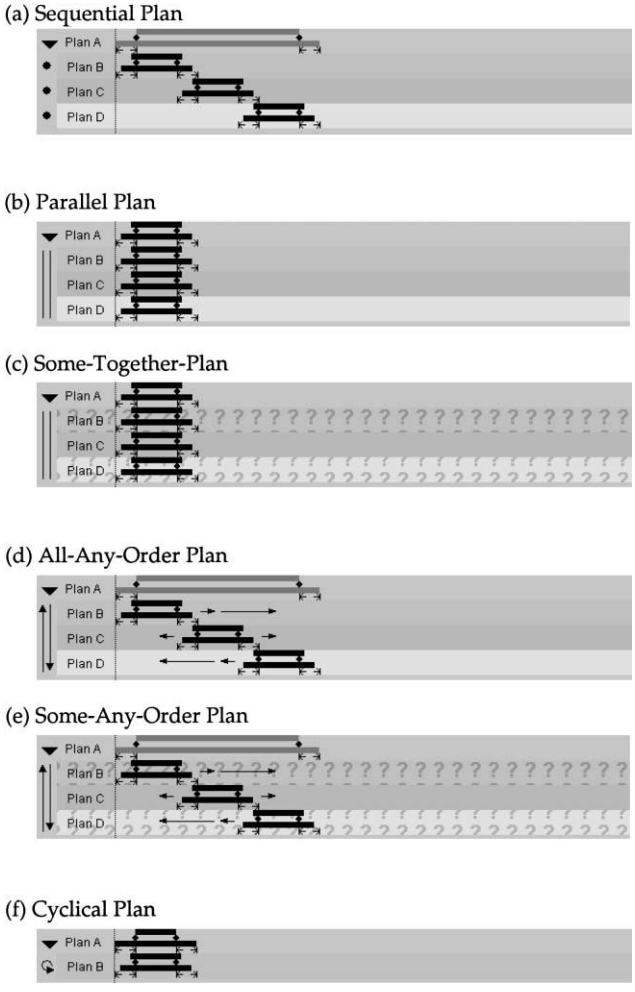


Fig. 8. All of Asbru's plan types in temporal view.

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 Fig. 9, 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 zig-zag pattern is drawn that covers the area of uncertainty (Fig. 9, lower right).

*4.2.4. Plan tree*

The topology of plans is also visible in the left part of Temporal View (see Fig. 10), 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.

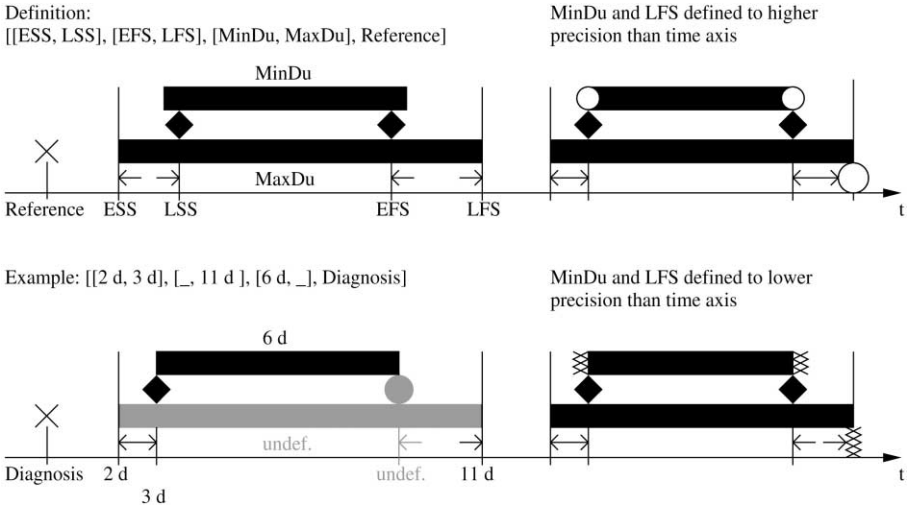


Fig. 9. Time Annotations, outline (adapted from [13]).

We use symbols to indicate different plan types (inspired by block structured diagrams used in programming [23]).

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.

### *4.3. User interaction*

Describing all the user interactions possible in the system is beyond the scope of this paper, but a short list of possibilities shall be presented here. Plans can be

- *Created*: Pressing the 'New' button (see Fig. 10) creates a new plan of the type currently selected in the field above the button.
- *Changed*: Pressing 'Change' changes the type of the current plan (which is the one with the dotted border) to the type indicated in the type field.
- *Reused*: Pressing 'Reuse' creates a reference to the current plan, which can then be moved anywhere, and behaves exactly like the original plan.
- *Deleted*: Pressing 'Delete' deletes the current plan and all the plans it contains.
- *Moved*: A plan can be grabbed in Topological View by simply pressing the mouse button when the mouse pointer is over it, and can be moved to any other location.
- *Folded*: Clicking the left mouse button in the little triangle at the front side of a plan, it can be folded (closed), so that its contents are not visible, or unfolded (opened), so that they are (see Fig. 3).

Plan libraries created this way can be saved and loaded, of course, and some control is possible how the plans are displayed (see for example the 'Show Conditions' checkbox underneath Topological View in Fig. 10).

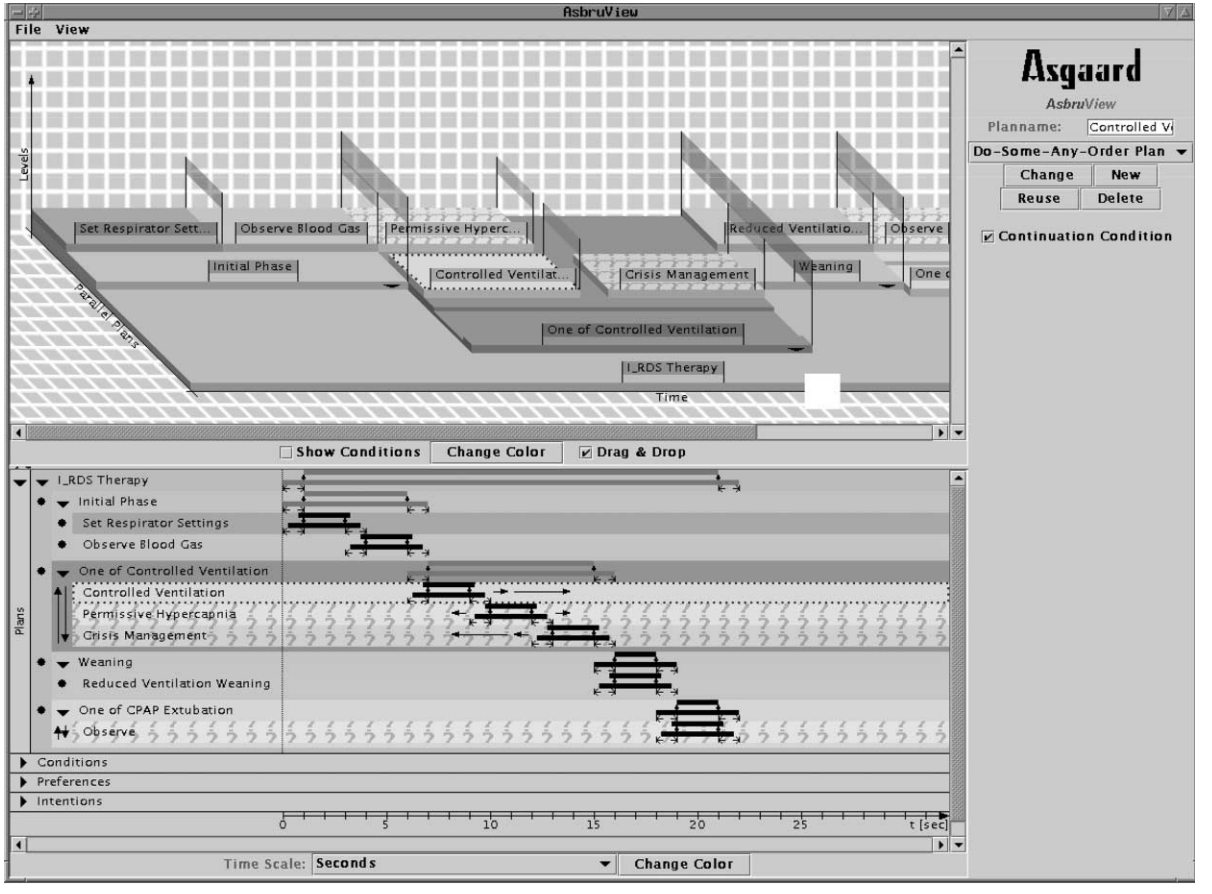


Fig. 10. A screen-shot of the AsbruView prototype showing a part of a plan for ventilation of new-born infants. Topological View is in the upper part, Temporal View in the lower part.

## 5. Usability

We performed a usability study with six physicians to asses AsbruView's usefulness [12]. For this purpose, we implemented a prototype of the system in Java. The participants were all physicians working in a number of different fields (neonatal intensive care, adult intensive care, psychiatry, pediatrics).

### *5.1. Evaluation method*

Because we do not have access to enough physicians for a quantitative study, we used a qualitative method based on questionnaires [35].

Every participant was first asked to fill out a short questionnaire about his or her computer knowledge. Based on this, an introduction to Asbru and AsbruView was given that lasted about 30 min. After that, the participants were asked to use AsbruView to author a plan for their everyday work. The tester took notes of how the participant performed and which problems he or she encountered (in this case, help was given). A second questionnaire was filled out by the participant after the test. A typical test lasted about 2 h.

The questionnaires and a more thorough description of the test procedure can be found in [12].

### *5.2. Findings*

The physicians, even those that had little experience with computers, performed surprisingly well. All of them found the metaphors used easy to understand and use. They understood the different plan types from the way they were laid out in Topology View, and from the symbols used in Temporal View.

Also the way temporal uncertainty is handled was found easy to understand and use. Even the meaning of reuse and the fact that a change in a reused plan would affect all other uses of that plan, was understood by most of the participants.

The use of color was considered helpful by all participants, even though some remarked that colors might be misleading. The use of gray in contrast for undefined components was also understood by all.

All users liked the fact that they could change a plan's type at any time. Interestingly, most of them assumed that this would not be possible. Being able to change a plan's type means less stress when authoring, because changes can be easily made.

All but one participant said they could imagine using the program for their daily work, and considered the program usable. All participants were of the opinion that the program would be better suited for work with clinical protocols.

One conceptual problem that became apparent during the evaluation is the fact that a plan defines the way its sub-plans are to be executed, and that a plan can only have one type — so it is not possible, for example, to have one sub-plan be executed first, and then afterwards two sub-plans in parallel. In such a case an intermediate parallel plan would be needed that would contain the latter two plans. But this means that sometimes artificial plans must be inserted that do not have any meaning for physicians.

This is a clear consequence of the fact that Asbru was designed by computer scientists, but AsbruView should try to bridge the gap here. One possible solution would be to allow any plan layouts, and to insert invisible artificial plans where necessary in order to comply with the rules of the language.

Even though colors were generally considered a very important part of the interface, the way they are used at the moment must be questioned. Colors should reflect a parameter, like the level of a plan, or its relationship with other plans (especially with reused plans). Some colors also should not be used (like red and yellow), because they suggest an important or critical plan.

All participants criticized the speed (or lack thereof) of the system — this was mostly due to its implementation in Java, but also because the test environment was not always perfect (we used a laptop computer for some of the tests).

## 6. Conclusion

We introduced a visualization and user interface that can be used to author clinical protocols expressed in the language Asbru. This interface consists of two parts, Topological View (which, as its name suggests, is mostly used for the correct order of and relations between plans), and Temporal View (which is used to work with detailed temporal information).

The findings of a short study to assess AsbruView's usability were presented. Even though a number of shortcomings were found, AsbruView has proven to be quite useful for physicians that have little or no experience with formal language (or computers in general).

We believe some of the findings and methods developed can be used in other interfaces or visualizations as well, when similar problems must be solved.

## Acknowledgements

We would like to thank Dr. Shahram Adel, Dr. Sophie Brandstetter, Dr. Maria Dobner, Dr. Gerhard Miksch, Primarius Dr. Franz Paky, and Univ. 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] Chernoff H. The use of faces to represent points in *k*-dimensional space graphically. J Am Statist Assoc 1973;68:361–8.

[2] Chuah MC, Eick SG. Glyphs for software visualization. In: 5th International Workshop on Program Comprehension (IWPC '97) Proceedings, IEEE Computer Soc. Press: Silver Spring, Dearborn, Michigan, 1997 May. p. 183–91.

[3] Combi C, Portoni L, Pinciroli F. Visualizing temporal clinical data on the www. In: Horn W, Shahar Y, Lindberg G, Andreassen S, Wyatt J, editors. Proceedings of the Joint European Conference on Artificial Intelligence in Medicine and Medical Decision Making (AIMDM'99). Berlin: Springer, 1999 June. p. 301–11.

[4] Cousins SB, Kahn MG. The visual display of temporal information. Artif Intel Med 1991;3(6):341–57.
[5] Friedland PE, Iwasaki Y. The concept and implementaion of skeletal plans. J Automated Reasoning 1985;1(2):161–208.
[6] Furnas GW. Generalized fisheye views. In: Mantei MM, Orbeton P, editors. Proceedings of the ACM Conference on Human Factors in Computer Systems, SIGCHI Bulletin. New York, USA, Association for Computer Machinery, 1986. p. 16–23.
[7] Goldstine HH, von Neumann J. Planning and coding problems for an electronic computing instrument, Part II, vol. 1. US Army Ordinance Department, 1947. Collected Works, vol. V. New York: McMillan. p. 80–151 [Reprinted in von Neumann J, 1963].
[8] Gross MH, Sprenger TC, Finger J. Visualizing information on a sphere. In: Proceedings of the 1997 Conference on Information Visualization. 1997 Oct.
[9] Herbert SI. Informatics for care protocols and guidelines: towards a European knowledge model. In: Gordon CJ, Christensen JP, editors. Health telematics for clinical guidelines and protocols. Amsterdam: IOS Press, 1994.
[10] Inselberg A. Multidimensional detective. In: Proceedings of the 1997 Conference on Information Visualization, 1997 Oct. p. 100–7.
[11] Inselberg A, Dimsdale B. Parallel coordinates for visualizing multi-dimensional geometry. In: Tsiyasu LK, editor. Proceedings of CG International on Computer Graphics 1987. Berlin: Springer, 1987. p. 25–44.
[12] Kosara R. Metaphors of movement — a user interface for manipulating time-oriented, skeletal plans. Master's thesis, Vienna University of Technology, Vienna, Austria, 1999 May.
[13] Kosara R, Miksch S. Visualization techniques for time-oriented, skeletal plans in medical therapy planning. In: Horn W, Shahar Y, Lindberg G, Andreassen S, Wyatt J, editors. Proceedings of the Joint European Conference on Artificial Intelligence in Medicine and Medical Decision Making (AIMDM'99), Aalborg, Denmark. Berlin: Springer, 1999 June. p. 291–300.
[14] Kosara R, Miksch S, Shahar Y, Johnson, P. AsbruView: capturing complex, time-oriented plans — beyond flow-charts. Thinking with diagrams, vol. 98. 1998 Aug 22–23.
[15] George L, Mark J. Metaphors We live by. Chicago: University of Chicago Press, 1980.
[16] Leung YK, Apperley MD. A review and taxonomy of distortion-oriented presentation techniques. ACM Trans Computer-Human Interaction 1994;1(2):126–60.
[17] MacEachren AM. Visualizing uncertain information. Cartographic Perspective 1992;13:10–9.
[18] Mackinlay JD, Robertson GG, Card SK. The perspective wall: detail and context smoothly integrated. In: Proceedings of ACM CHI'91 Conference on Human Factors in Computing Systems. Information Visualization, 1991. p. 173–9.
[19] Martin JJ. The 'natural' set of basic control structures. SIGPLAN Notices 1973;8(12):5–14.
[20] Miksch S, Shahar Y, Horn W, Popow C, Paky F, Johnson P. Time-oriented skeletal plans: support to design and execution. In: Proceedings of the Fourth European Conference on Planning (ECP'97). Berlin: Springer, 1997 Sept 24–26.
[21] Miksch S, Shahar Y, Johnson P. Asbru: A task-specific, intention-based, and time-oriented language for representing skeletal plans. In: Proceedings of the Seventh Workshop on Knowledge Engineering: Methods & Languages (KEML-97). Milton Keynes (UK): Open University, 1997.
[22] Mukherjea S, Hirata K, Hara Y. Visualizing the results of multimedia web search engines. In: Proceedings of the IEEE Symposium on Information Visualization, 1996. p. 64–5.
[23] Nassi I, Shneiderman B. Flowchart techniques for structured programming. SIGPLAN Notices 1973;8(8):12–26.
[24] Pang A, Wittenbrink C, Lodha S. Approaches to uncertainty visualization. Technical Report UCSC-CRL-96-21, University of California, Santa Cruz, Jack Baskin School of Engineering, 1996 Sept.
[25] Plaisant C, Milash B, Rose A, Widoff S, Shneiderman B. LifeLines: visualizing personal histories. In: Proceedings of ACM CHI'96 Conference on Human Factors in Computing Systems, vol. 1 of PAPERS: Interactive Information Retrieval, 1996. p. 221–7.
[26] Plaisant C, Mushlin R, Snyder A, Li J, Heller D, Shneiderman B. LifeLines: using visualization to enhance navigation and analysis of patient records. In: Proceedings of the 1998 American Medical Informatic Association Annual Fall Symposium, 1998 Nov 9–11. p. 76–80.
[27] Powsner SM, Tufte ER. Graphical summary of patient status. The Lancet 1994;344:386–9.

[28] Purgathofer W, Löffelmann H. Selected new trends in scientific visualization, Technical report TR-186-2-97-17, Computer Graphics, Visualisation and Animation Group, Vienna University of Technology, 1997 Sept.
[29] Rit J-F. Propagating temporal constraints for scheduling. In: Proceedings of the Fifth National Conference on Artificial Intelligence. Los Altos: Morgan Kaufman, 11–15 August 1986. p. 383–8.
[30] Robertson GG, Mackinlay JD, Card SK. Cone trees: animated 3D visualizations of hierarchical information. In: Proceedings of ACM CHI'91 Conference on Human Factors in Computing Systems, Information Visualization, 1991. p. 189–94.
[31] Sarkar M, Snibbe SS, Tversky OJ, Reiss SP. Stretching the rubber sheet: a metaphor for visualizing large layouts on small screens. In: Proceedings of the ACM Symposium on User Interface Software and Technology, Visualizing Information, 1993. p. 81–91.
[32] Shahar Y, Cheng C. Model-based visualization of temporal abstractions. In: Khatib L, Morris R, editors. Proceedings of the Fifth International Workshop on Temporal Representation and Reasoning (TIME98). Silver Spring: IEEE Computer Soc. Press, 1998. p. 11–20.
[33] Shneiderman B. Tree visualization with tree-maps: a 2-D space-filling approach. ACM Trans Graphics 1992;11(1):92–9.
[34] Shneiderman B. The eyes have it: a task by data type taxonomy for information visualizations. In: Proceedings of the IEEE Symposium on Visual Languages, Washington, 1996 Sept 3–6. Silver Spring: IEEE Computer Soc. Press. p. 336–43.
[35] Shneiderman B. Designing the user interface: strategies for effective human–computer interaction. 3rd ed. Reading (MA): Addison Wesley, 1997.
[36] Stevenson DA, Guan X, MacGallum KJ, Duffy AHB. Sketching on the back of the computational envelope ... and then posting it? In: Proceedings of the Workshop on Visual Representation, Reasoning and Interaction in Design, Fourth International Conference on Artificial Intelligence in Design 1996 (AID'96). Stanford University, USA, 1996 June 23.
[37] Tufte ER. The visual display of quantitative information. Cheshire (CT): Graphics Press, 1983.
[38] Tufte ER. Envisioning information. Cheshire (CT): Graphics Press, 1990.