Time Shapes - A Visualization for Temporal Uncertainty in Planning

eingereicht von

Peter Messner

Diplomarbeit
zur Erlangung des akademischen Grades
Magister rerum socialium oeconomicarumque
Magister der Sozial- und Wirtschaftswissenschaften
(Mag. rer. soc. oec.)

Sozial- und Wirtschaftswissenschaftliche Fakultät
Universität Wien
Technisch-Naturwissenschaftliche Fakultät
Technische Universität Wien

Studienrichtung: Wirtschaftsinformatik

Begutachterin: ao. Univ.-Prof. Mag. Dr. Silvia Miksch
Institut für Softwaretechnik der Technischen Universität Wien

Wien, im April 2000

Dedicated to my grandmother Anna,
for her love and support through all the years.

Kurzfassung

Diese Arbeit beschreibt eine zweidimensionale Darstellung von zeitlichen Unsicherheiten im Planungsprozess, die wir SOPOView nennen. Sie ist Bestandteil des Programms AsbruView, das entwickelt wurde, um die Arbeit mit Plänen, die in der Sprache Asbru definiert wurden, zu unterstützen. Der primäre Anwendungsbereich dieser Sprache ist die medizinische Therapieplanung; die von uns präsentierte Lösung ist jedoch nicht auf diesen Anwendungsbereich beschränkt.

Wir stellen zunächst die Sprache Asbru sowie das Programm AsbruView in dem Ausmass vor, wie es für das Verständnis des Kontexts sowie der Anforderungen an SOPOView notwendig ist.

Unsere Darstellung erlaubt die Definition von Unsicherheiten in der Startzeit, Endzeit und Dauer eines Planes. Wir stellen das zugrundeliegende Konzept der SOPOs vor, sowie unsere Anpassungen und Erweiterungen, die notwendig waren, um den Anforderungen der Sprache Asbru gerecht zu werden. Wir skizzieren kurz den Entwicklungsprozess vom Design bis hin zur Implementierung und gehen dabei auf die besondere Rolle der Usability (Benutzbarkeit) ein.

Wir haben SOPOView mit insgesamt acht ÄrztInnen evaluiert. Die Ergebnisse und Konsequenzen dieser Evaluation werden präsentiert und ausführlich diskutiert.

Abstract

This thesis introduces a two-dimensional visualization of temporal uncertainty in planning, which we called SOPOView. It is part of the user interface AsbruView, that supports the understanding and manipulation of plans written in the representation language Asbru. Asbru's primary application domain is that of medical therapy planning; however, our solution is not restricted to that domain.

We present the basics of the language Asbru and of the user interface AsbruView, that are necessary in order to understand the context and requirements for our own solution SOPOView.

Our visualization enables a definition of uncertainty in a plan's starting time, ending time, and duration. We present the underlying concept of SOPOs and our adaptations and enhancements that were necessary in order to meet Asbru's requirements. We briefly outline the process from early design to the implementation and explain the role of usability within our approach.

We have evaluated SOPOView with eight domain experts (physicians). The findings of this evaluation are presented and discussed in detail.

Contents

List of Figures

Acknowledgments

First of all, I would like to thank my supervisor Silvia Miksch for her support and the friendly atmosphere within which I could carry out my work. It was approximately a year ago, when I first came into contact with her and started to get interested in the medical application domain. It was she who encouraged me to engage in the rather difficult and complex topic that this thesis was for me in the very beginning.

Robert Kosara deserves special thanks for his numerous suggestions and substantial contributions to my work. It was he who developed AsbruView, which lays the foundation of my own program. At all times during design and implementation, Robert offered me his help and knowledge, which I cannot appreciate enough. Without him, this work would have been a different one.

The physicians who helped us during evaluation also deserve special thanks. They not only took their time during working hours but gave us valuable feedback for further development. The participants were (in alphabetical order and using German titles):
Dr. Shahram Adel, Dr. Sophia Brandstetter, Dr. Maria Dobner, Dr. Thomas Dobner, Dr. Lieselotte Kirchner, Dr. Manuel Langer, ao. Univ.-Prof. Dr. Christian Popow, Dr. Birgit Rami, and Dr. Sabine Wagner, who assisted during one test.

Thanks to Michael Sundell for an early revision of this thesis.

A picture is worth a thousand words,
unless of course, you're talking about
a picture of a thousand words.

(Anonymous)

To be uncertain is to be uncomfortable.
To be certain is to be ridiculous.

(Chinese Proverb)

Chapter 1

Introduction

In this chapter, we introduce the application domain briefly as well as the motivation and overall structure of this thesis.

1.1 Motivation

Within the medical domain, clinical guidelines and protocols are used to communicate procedural knowledge and to guide the medical staff. Thus, treatment planning from scratch is typically not necessary. Additionally, these guidelines and protocols help to assess and assure a treatment's quality, to improve decision-support, and to reduce the cost.

Besides free text, several methods have been presented that support the authoring of such guidelines and protocols, including flowcharts or decision tables. Using those methods, a high number of clinical protocols have been acquired. However, these representations often lack the means to fully capture both the complexity and the temporal dimension of treatment plans.

In the Asgaard Project, a time-oriented machine-readable language, called Asbru, was developed to represent and to annotate durative treatment plans. As it would nearly be impossible for physicians and other medical staff to compose plans in Asbru, a user interface, called AsbruView, was developed to provide means for the understanding and manipulation of those plans used in medical therapy planning. Basically, AsbruView consists of two different views: one representing the topology, i.e. the structure, of plans, and the other representing the temporal dimension.

Especially, dealing with temporal aspects is of fundamental importance in planning and decision processes, not only in the medical domain. Most of the clinical data are time-stamped, telling the physician what to do during a certain time interval or at an exact time point. However, uncertainty in those temporal relations can be hard to visualize.

This thesis introduces a two-dimensional alternative representation of uncertain temporal aspects of plans, which we called SOPOView.

Utilization of SOPOView is, however, not restricted to the medical application domain. This visualization can be used in planning in general, where uncertainty in temporal information emerges.

1.2 Structure of the Thesis

In the next chapter, we present the framework for this thesis, namely the Asgaard/Asbru Project. The basic concepts of the language Asbru as well as the user interface AsbruView will be described briefly in order to understand the context and the requirements for our solution SOPOView. The presentation of three selected approaches in computerization of clinical guidelines rounds out the chapter. Afterwards, fundamental concepts for the visualization of temporal information will be discussed and an in-depth description of the concept we used for our approach will be given. In chapter 4, we introduce the basics of the field of user-interface design and explain the role of usability. Besides, we describe our own process from the early beginning in design up to the implemented program.

Finally, our solution SOPOView will be presented. The ideas behind the representation and the user interaction possibilities will be discussed in detail. The following chapter describes the evaluation we performed with physicians to test SOPOView's usability and suitability and reports the results that led to the final conclusion, given in chapter 7. In the appendix, we attached the questionnaires we used for the evaluation in both the german original version and the english translation, and the numerical results of the evaluation.

Chapter 2

The Asgaard/Asbru Approach

This chapter introduces the general framework for this thesis, the basics of the language Asbru and the user interface AsbruView that supports the understanding and manipulation of time-oriented, skeletal plans written in Asbru.

2.1 About the Project

To assure a certain level of quality in clinical care and treatment, guidelines and protocols became an important tool to communicate clinical procedural knowledge, collected through experience or clinical studies, over the last decades. *Clinical guidelines* can be considered as a set of reusable skeletal plans that are used for management of patients who have a particular clinical condition. They are instantiated and refined dynamically by care providers, such as physicians, over significant periods of time. *Skeletal plans* can be defined as plan schemata at various levels of detail that capture the essence of a procedure, but leave room for execution-time flexibility in the achievement of particular goals [Friedland and Iwasaki, 1985]. *Clinical protocols* are a more detailed version of clinical guidelines. They are often used when guidelines need to be applied uniformly to enable statistical analysis of outcomes for comparison among a set of guidelines (e.g., experimental chemotherapy protocols for cancer therapy) [Miksch et al., 1997][1].

In the Asgaard Project, a set of tasks and computational models are investigated that support the application and execution of clinical guidelines by a care provider other than the guideline's designer [Shahar et al., 1998]. The project tries to build the bridge between the planning approaches and the medical approaches, addressing the demands of the medical staff on the one side and applying rich plan management on the other side [Miksch, 1999].

1 Throughout this thesis, the terms *protocol*, *guideline*, and *(treatment) plan* will be used interchangeably.

The Asgaard system was designed to support a wide range of medical applications, from intensive care units to competitive sports. So far, parts of the project have been applied to clinical treatment protocols for artificial ventilation of newborn infants and for gestational diabetes mellitus.

The underlying requirement for the development of task-specific problem-solving methods is the existence of a powerful modeling language. As the concept of time is a fundamental decision-criterion in medicine, this language needs to be expressive with respect to temporal annotations. Furthermore, it needs to have a rich set of parallel, sequential, and iterative operators. Given these requirements, the time-oriented, intention-based plan representation language Asbru was developed.

2.2 The Asbru Language

Asbru is the language used for the representation of clinical guidelines in the Asgaard Project. In Asbru, guidelines are modeled by a set of hierarchical skeletal plans. A detailed description of the language is given by Miksch et al. [1997] or Shahar et al. [1998]. In the following sections, we will give a brief overview of those features of Asbru that are necessary in order to understand the concepts of the user interface, especially the one used for representing temporal aspects of plans.

2.2.1 Basic Concept

The basic entity in Asbru is the plan. A plan consists of a unique name, an optional set of arguments including a time annotation (representing the temporal scope of the plan), and the following five optional components: a plan body, preferences, intentions, conditions and effects. A plan is composed hierarchically of a set of sub-plans. Each plan may contain any number of sub-plans, which may themselves be composed of sub-plans again, and so forth. A plan that is not further decomposed is called an action. Plans can be executed in parallel, in sequence, in a particular order, or every time measure.

2.2.2 Time Annotations

In order to define temporal aspects of plans, time annotations are used. They allow a representation of uncertainty in starting time, ending time, and duration [Rit, 1986] (for

a detailed description of this concept, see section 3.3). The time annotation supports multiple time lines (e.g., different zero-time points and time units) by providing reference annotations. The reference annotation can be an absolute reference point, a reference point with uncertainty (defined by an uncertainty region), a function of a previously executed plan instance (e.g., start plan B thirty minutes after having completed plan A), or a domain-dependent time point variable (e.g., conception) [Miksch et al., 1997].

The uncertainty in starting time, ending time, and duration is represented by temporal shifts from the reference annotation (see Table 2.1).

Name	**Short**	**Description**
Earliest Starting Shift	ESS	Defines the earliest point in time when the plan (or the action) can start.
Latest Starting Shift	LSS	Defines the latest point in time when the plan must start.
Earliest Finishing Shift	EFS	Defines the earliest point in time when the plan can end.
Latest Finishing Shift	LFS	Defines the latest point in time when the plan must end.
Minimal Duration	MinDu	Defines the minimal duration of the plan.
Maximal Duration	MaxDu	Defines the maximal duration of the plan.

Table 2.1: Components of a time annotation.

The temporal shifts are associated with time units (e.g., minutes, days) or domain-dependent units (e.g., gestational weeks). Each shift may or may not be defined in a time annotation. This is useful because during the design phase of a plan, some parts may not be known or they may not be of interest (e.g., only the duration is of importance, but not the plan's starting and ending interval).

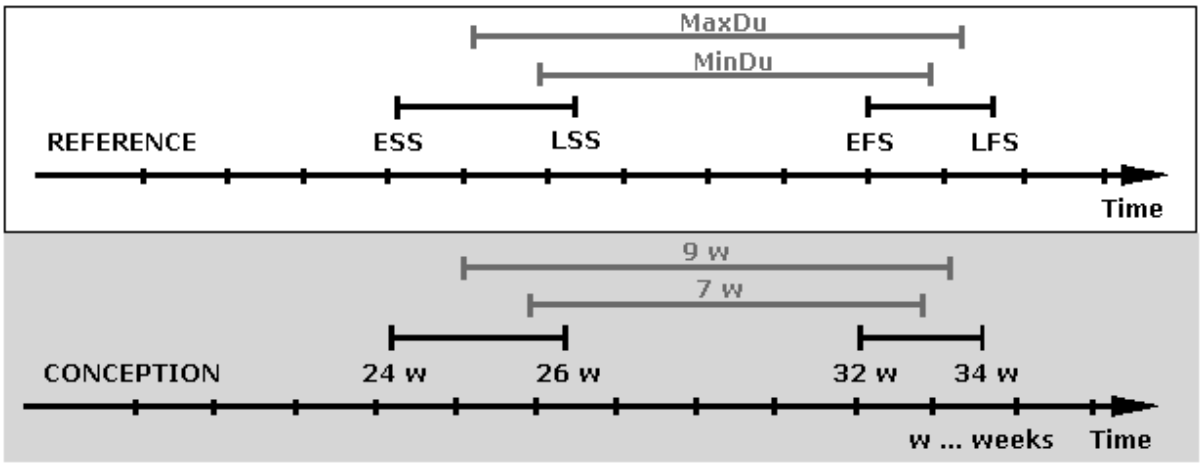


Figure 2.1: A schematic illustration of the Asbru time annotations. The upper part presents the generic annotation; the lower part shows a particular example [Miksch et al., 1997].

In Asbru, a time annotation is written like this:

```
[[ESS, LSS], [EFS, LFS], [MinDu, MaxDu], REFERENCE].
```

The time annotation of the example shown in Figure 2.1 would thus be

```
[[24 WEEKS, 26 WEEKS], [32 WEEKS, 34 WEEKS], [7 WEEKS, 9 WEEKS], CONCEPTION],
```

which means "starts 24 to 26 weeks after conception, ends 32 to 34 weeks after the conception, and lasts 7 to 9 weeks".

As mentioned before, all the temporal parameters can be unknown or undefined to allow incomplete time annotations. In Asbru, this is denoted by an underscore, "_". For an example of undefined parameters, see Figure 3.3. In addition, certain short-cuts, such as one for the current time (using the symbol "`*NOW*`"), or the duration of the plan (using the symbol "*") are allowed. Furthermore, sets of cyclical time points are defined to allow temporal repetitions (e.g., `MORNINGS`, or `MIDNIGHTS`).

2.2.3 Constraints within Time Annotations

Taking any time annotation [[ESS, LSS], [EFS, LFS], [MinDu, MaxDu], REFERENCE], one has to assure that every single component must be reasonable and all components of this time annotation must be compatible in the following way:

- For every single time point within the starting interval [ESS, LSS], there must exist at least one duration out of the interval [MinDu, MaxDu] that allows the finishing interval [EFS, LFS] to be reached.

- Every single time point within the finishing interval [EFS, LFS], must be reachable by at least one duration out of the interval [MinDu, MaxDu] from the starting interval [ESS, LSS].

To enable the previous two statements to be true, we can summarize seven requirements for a reasonable time annotation. A detailed description is given by Duftschmid [1999]:

1. ESS ≤ LSS.

2. EFS ≤ LFS.

3. ESS ≤ EFS.

4. LSS ≤ LFS.

5. MinDu ≤ MaxDu.

6. (EFS - LSS) ≤ MinDu ≤ MINIMUM [(EFS - ESS), (LFS - LSS)].

   First, MinDu only makes sense if it's value is at least (EFS - LSS), as otherwise EFS could never be reached with a plan's duration equal to MinDu.

   Second, MinDu should not be larger than the minimum of (EFS - ESS) and (LFS - LSS), as this would mean that either for the starting point ESS the finishing point EFS could not be reached, or for the starting point LSS the finishing point LFS could not be reached.

7. MAXIMUM [(EFS - ESS), (LFS - LSS)] ≤ MaxDu ≤ (LFS - ESS).

   First, MaxDu only makes sense if it's value is at most (LFS - ESS), as otherwise LFS could never be reached with a plan's duration equal to MaxDu.

   Second, MaxDu should always be larger than the maximum of (EFS - ESS) and (LFS - LSS), as this would mean that either for the starting point ESS the finishing point EFS could not be reached, or for the starting point LSS the finishing point EFS could not be reached.

2.2.4 The Plan Body

The plan body is a set of sub-plans to be executed in a certain way, specified by the plan type. There are several types of plans - whereas one single plan body can only have one type: sequential, concurrent or cyclical. A sequential plan specifies a set of sub-plans that are executed in total sequence. Concurrent plans can either be executed in parallel or in any order. There are two dimensions for classification of sequential or concurrent plans: the number of plans that should be completed to enable continuation and the order of plan execution. For an overview, see Table 2.2. A cyclical plan includes a plan that can be repeated, and optional temporal and continuation arguments that can further specify its behaviour. These basic plan types will be described here briefly:

2.2.4.1 Sequential Plans

A plan with the type `DO-ALL-SEQUENTIALLY` (which is the proper expression in Asbru) consists of several sub-plans that are to be executed in total sequence. All sub-plans must be executed in a predefined order. When the last sub-plan is completed successfully, the whole plan is completed, too.
A variant of this plan type is the type `DO-SOME-SEQUENTIALLY` where only a subset of the sub-plans have to be performed in order to complete. These plans are part of the continuation condition. All the other plans, i.e. those that do not have to be performed, are called optional plans.

2.2.4.2 Any-Order Plans

If the order of execution of a plan's sub-plans is not known before treatment, one can use one of the following two plan types: `DO-ALL-ANY-ORDER` and `DO-SOME-ANY-ORDER`. `DO-ALL-ANY-ORDER` plans consist of sub-plans that are to be executed in any possible order. All sub-plans however have to be completed successfully for continuation.
Like a sequential plan, it is also possible that only a subset of the plans must be executed in order to complete. The appropriate plan type will then be `DO-SOME-ANY-ORDER`. Again, the plans that have to be performed are within the continuation condition, all the others are optional plans.

2.2.4.3 Parallel Plans

A plan with the type `DO-ALL-TOGETHER` consists of a set of sub-plans that have to start together but may have different durations. All of the sub-plans must be successfully completed for the containing plan to succeed. If only a subset of the sub-plans must be performed, the containing plan will have the type `DO-SOME-TOGETHER`.

2.2.4.4 Cyclical Plans

In order to represent repetitive actions (e.g., performing a blood glucose test before every meal), cyclical plans (`DO-EVERY`) can be used. Temporal and continuation arguments can be specified, e.g., the maximum number of attempts or the delay between retries. A cyclical plan can only have one sub-plan, this sub-plan however can consist of several sub-plans again.

	All plans must complete to continue	**Some plans must complete to continue (continuation condition specified as subset of plans)**
Start together	`DO-ALL-TOGETHER`	`DO-SOME-TOGETHER`
Execute in any order	`DO-ALL-ANY-ORDER`	`DO-SOME-ANY-ORDER`
Execute in total order (sequence)	`DO-ALL-SEQUENTIALLY`	`DO-SOME-SEQUENTIALLY`

Table 2.2: Plan Types in Asbru.

2.2.5 Preferences, Intentions, Conditions and Effects

Preferences, intentions, conditions and effects are not of importance for the present thesis. However, for reasons of completeness, they will be described here briefly. A detailed description of these components of a plan is given by Miksch et al. [1997] and Shahar et al. [1998].

Preferences are plan parameters that express a kind of behaviour of the plan, or bias or constrain the selection of the plan and thus influence the overall applicability of the plan. Intentions are high-level goals at various levels of the plan. They can be thought as temporal patterns to be maintained, achieved or avoided. Conditions are temporal patterns that need to hold at particular plan steps to induce a particular state transition of the plan instance. By using conditions, one can decide which plan should be applied due to a certain state of the patient that the plan was designed for. Effects describe the relationship between plan arguments and measureable parameters or the overall effect of a plan on parameters when applied to a patient.

2.3 Visualization of Asbru Plans

In order to give users access to Asbru, a user interface has been developed that will be presented in the following section. Before, we briefly describe the challenges in visualizing plans written in Asbru (a detailed description is given by Kosara [1999]).

Hierarchical Decomposition. Plans can either be actions or consist of sub-plans, and they can be reused as as sub-plan of another plan. The visualization has to communicate this concept.

Temporal Order. The way a plan's sub-plans are to be performed, indicated by the plan type, should be easy to recognize from the representation.

Compulsory vs. Optional Plans. Plans (or actions) either *must* be executed, or they can be optional. The visualization has to elucidate this difference.

Cyclical Plans. Many tasks in treatment plans can be repetitive. A way has to be found to represent the cyclical nature of a plan.

Temporal Uncertainty. A plan's time annotation does not have to be complete. Therefore, the visualization must indicate undefined parts. Furthermore, uncertainty in the plan's beginning, end, and duration must be made clear.

Figure 2.2 shows an early example of a graphical representation of a guideline specification [Miksch et al., 1997]. One can easily see that the given requirements are hardly met, especially regarding temporal uncertainty, about which the representa-tion cannot tell anything.

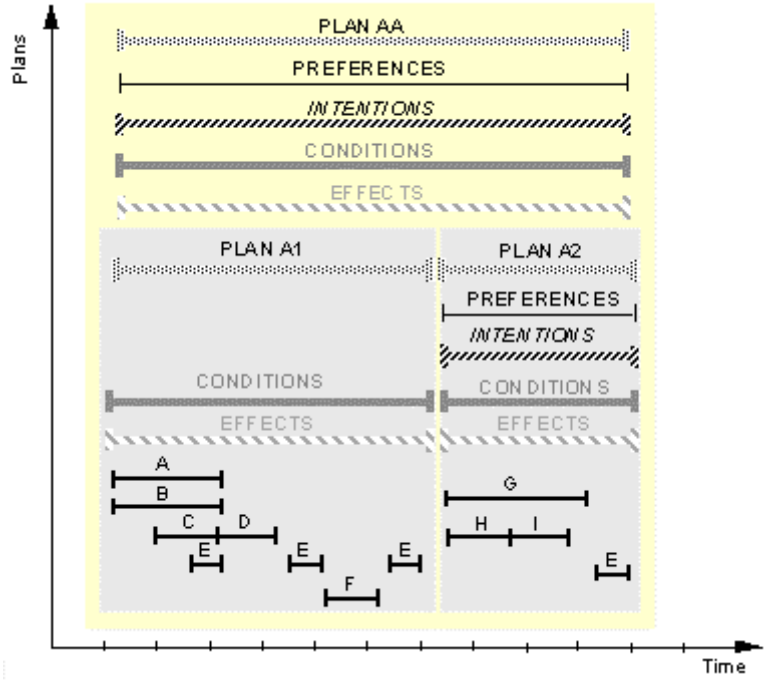


Figure 2.2: Graphical representation of a clinical guideline specification.

2.4 The User Interface AsbruView

AsbruView is the user interface that was developed to provide a means for manipulating time-oriented, skeletal plans written in Asbru [Kosara, 1999]. As the solution for visualizing temporal aspects of plans presented in this thesis (SOPOView) will be part of AsbruView, one has to know the basics of this user interface and the way plans and their temporal dimensions are depicted.

2.4.1 Program Design

The program consists of three main components: the model, a controller, and any number of views (see Figure 2.3). The model contains the raw data, i.e. the representation of the plans themselves. All changes made to the model are handled by the controller which provides a basic user interface for the actions. The controller notifies the views of changes in the model. Views display the model and accept user inputs that they convert into method calls to the controller. Communication between views and controller is performed through interfaces. In AsbruView, there are two views: the topological view and the temporal view. The way of visualizing temporal aspects of plans presented in this thesis (SOPOView) was implemented as another view, thus extending the user interface AsbruView.

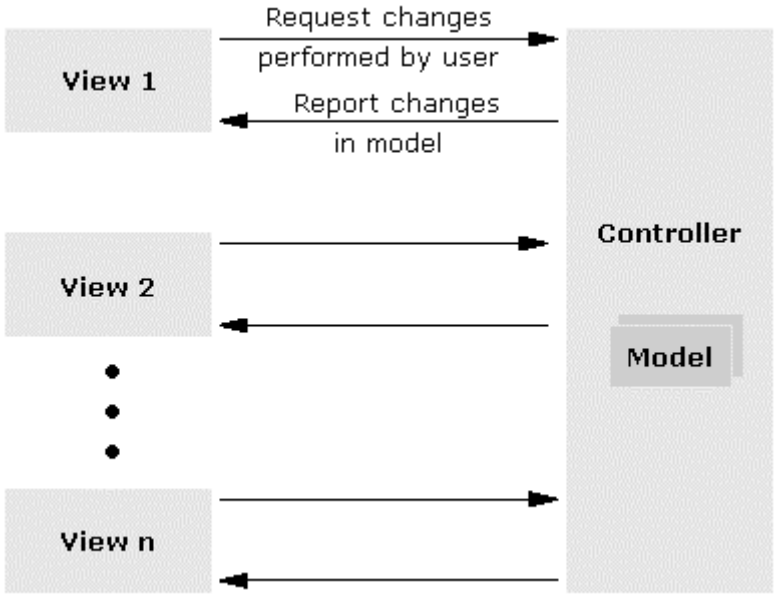


Figure 2.3: AsbruView's principal architecture: A controller containing a model and communicating with a number of views [Kosara, 1999].

2.4.2 Topological View

This view deals with the topology, i.e. the layout or the hierarchical structure of the plans. It shows whether plans are to be performed in sequence, in parallel, etc. However, it does not deal with temporal issues (that is done in the Temporal View, see below). In order to make the view easier to understand, a number of metaphors are used for Asbru's concepts. The basic metaphor behind a plan is that of a running track. During the

execution of the plan, the physician (or the patient) is considered to be running along the plan, starting at the left side, until reaching the finishing flag on the right when the plan has succeeded. Plans are stacked on top of each other to depict the hierarchical decomposition of plans. Optional plans are marked by a question-mark texture on the top face. This texture proved to be easy to understand and is used throughout all views. The plan types are indicated by the way the sub-plans are arranged. For an example of the Topological View, see Figure 2.5.

2.4.3 Temporal View

This view is two-dimensional and based on the idea of timelines or LifeLines (e.g., [Plaisant et al., 1998]; a detailed description of this concept is also given in section 3.1). However, timelines are not usable for the kind of temporal uncertainty used in Asbru. Therefore, a special glyph was developed that replaces the simple line. This time annotation glyph, as we call it, will be discussed in detail in section 3.2.
In the Temporal View, a plan is represented as a colored rectangle that may contain other rectangles depending on whether the plan has sub-plans or not. The rectangles are divided into a left and a right part. In the left part, the plans' names and the plan types are shown whereas in the right part, the time annotation glyphs are drawn. The two parts are separated by a dotted line, that also marks the beginning of the time axis. As the left part of this visualization is also used in SOPOView, a detailed description can be found in section 5.1. For an example of the Temporal View, see Figure 2.5.

2.4.4 Common Concepts

Throughout all views, plans are represented by graphical objects which need to be assigned a color to differentiate between them. These colors are the same for a given plan in all the views, thus making identification across views easier.

The user can always choose the level of detail to be shown regarding the hierarchical decomposition of plans. Plans can therefore be opened or closed in order to see or hide the structure of the underlying sub-plans. This functionality is made available through triangles that can be clicked in both Topological View and Temporal View.

Optional plans, as mentioned before, are drawn with a question-mark texture throughout all views, thus enabling users to recognize these plans more easily.

2.5 From Asbru to AsbruView: An Example

The following treatment plan for I-RDS (infants' respiratory distress syndrome) was presented by Miksch [1999] and shall help explain how such a plan is written in Asbru (Figure 2.4) and represented in AsbruView (Figure 2.5). The example covers only the highest level of the plan hierarchy.

2.5.1 Natural Language

> After infants' respiratory distress syndrome (I-RDS) is diagnosed, a plan dealing with limited monitoring possibilities is activated, called *initial-phas*e. Depending on the severity of the disease, three different kinds of plans are followed: *controlled-ventilatio*n, *permissive-hypercapni*a, or *crisis-managemen*t. Only one plan at a time can be activated; however, the order of execution and the activation frequency of the three different plans depend on the severity of the disease. Additionally, it is important to continue with the plan *weaning* only after a successful completion of the plan *controlled-ventilatio*n. After a successful execution of the plan *weanin*g, the extubation should be initiated. The extubation can be either a single plan *extubation* or a sequential execution of the sub-plans *cpap* and *extubatio*n.
>
> The most important part is the sub-plan *controlled-ventilation.* The intentions of this sub-plan are to maintain a normal level of the blood-gas values and the lowest level of mechanical ventilation (as defined in the context of controlled ventilation therapy) during the span of time over which the sub-plan is executed. This sub-plan is activated immediately, if peak inspiratory pressure PIP ≤ 30 and the transcutaneously assessed blood-gas values are available for at least one minute after activating the last plan instance *initial-phase* (as reference point). The sub-plan must be aborted, if PIP > 30 or the increase of the blood-gas level is too steep (as defined in the context of controlled ventilation-therapy) for at least 30 seconds. The sampling frequency of the abort condition is 10 seconds. The sub-plan is completed successfully, if F_iO_2 ≤ 50%, PIP ≤ 23, f ≤ 60, the patient is not dyspnoeic, and the level of blood gas is normal or above the normal range (as defined in the context of controlled ventilation-therapy) for at least three hours. The sampling frequency of the complete condition is 10 minutes. The body of the sub-plan *controlled-ventilation* consists of a sequential execution of the two sub-plans *one-of-increase-decrease-ventilation* and *observin*g.

2.5.2 Asbru Code

```
(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-l#)]))
(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 DYSPNOEIC) 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 2.4: An example of Asbru code (part of a treatment plan for infants' respiratory distress syndrome (I-RDS)).

2.5.3 AsbruView

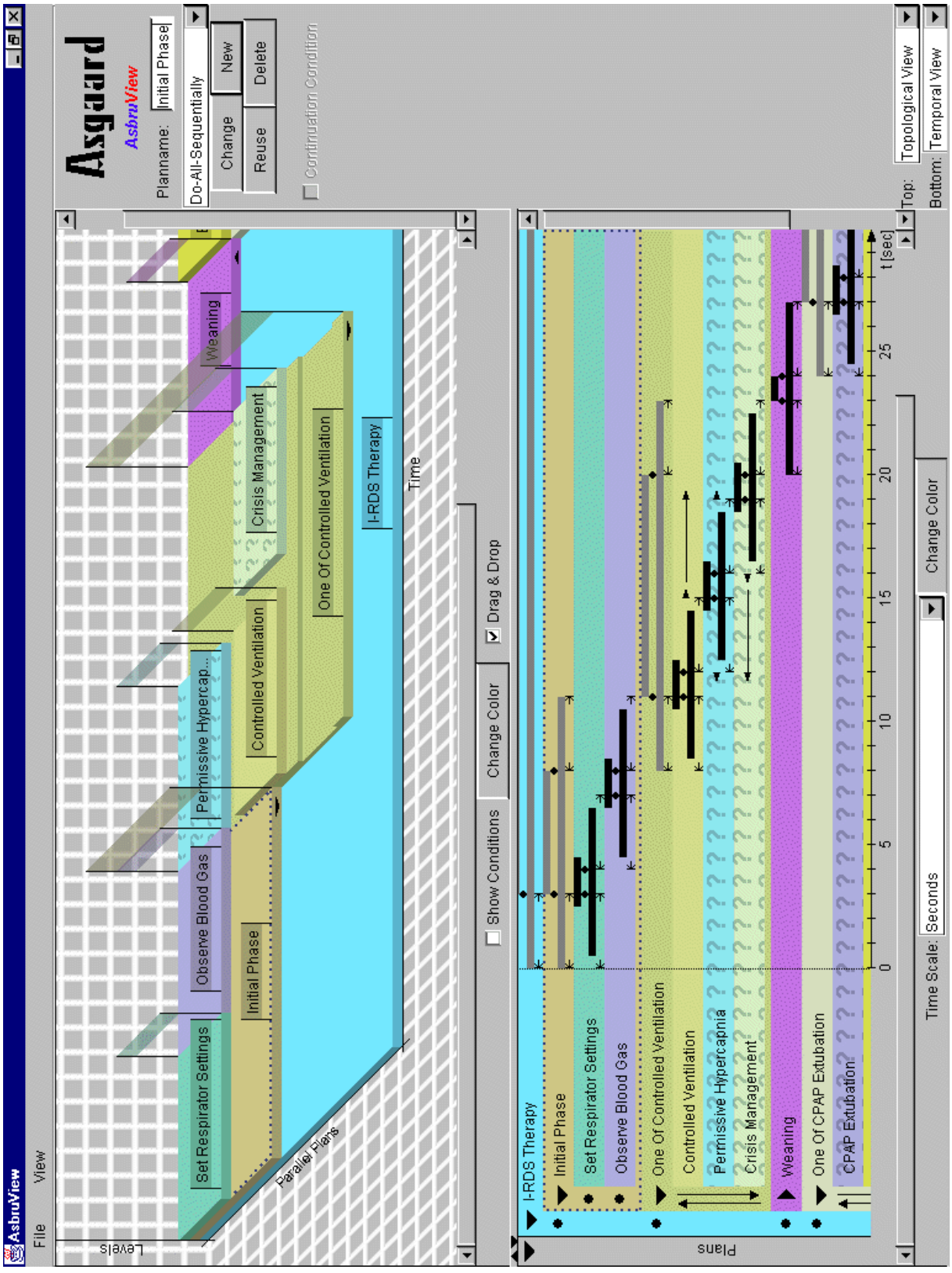


Figure 2.5: Screenshot of the AsbruView program showing the protocol for treating infants' respiratory distress syndrome (I-RDS). The left/upper half shows the Topological View, the right/lower half shows the Temporal View.

2.6 Related Work: Protocol-Based Care

In this section, three approaches in the area of computerization of clinical guidelines will be presented. These selected projects shall illustrate the research and work that has been done within the domain. A more comprehensive overview is given by Duftschmid [1999], for example.

2.6.1 MLMs and the Arden Syntax

Development of the Arden syntax was quickly advanced because of the pressing needs to facilitate exchange of protocols among health-care institutions. The Arden syntax is a standard procedural language that encodes situation-action rules [Hripcsak et al., 1994]. These rules are implemented as software modules and are called Medical Logic Modules (MLMs). A plan or protocol is modeled by a set of MLMs. They are mostly used for the implementation of alert- or reminder-systems.

Originally, MLMs were designed to have a single set of input data, a single application of criteria logic, and a single set of resulting actions. A single MLM is thus not sufficient for the representation of complex guidelines, because protocols consist of sequences of actions. Each sequence requires its own set of data including dependencies between different actions and decisions. Efforts have been made to implement guidelines by chaining together multiple MLMs [Sherman et al., 1995]. There, a guideline is modeled by a set of MLMs that are sequentially executed, and each MLM implements one single step within the guideline. Typically, MLMs interact with the physician through messages. These messages may serve to alert health care staff of a potentially worrisome patient situation, or they may act as reminders indicating the execution of a certain action by the physician. MLMs use various kinds of elements to structure the control flow: a *data slot* to specify the links to patients' records; an *evoke slot* to define the condition that triggers a MLM; and a *logic slot* to determine whether the one action defined within the *action slot* should be done. Additionally, there are slots concerning the maintenance of the module and references to the literature.

Examples of application areas are hypokalemia with digoxin therapy, tuberculosis cultures with positive or invalid results, calculation of creatinine clearance, and identification of patient records that include admission but no discharge summaries.

2.6.2 The EON Approach

The EON system is composed of a set of cooperating components, which represent independent, reusable software modules that developers can assemble to build systems that solve tasks related to the administration of protocol-directed therapy [Musen et al., 1996; Tu and Musen, 1996]. These components are: *problem-solving methods* that interpret the abstract protocol specifications, and that apply those specifications to individual patients; a *temporal-reasoning system*, called RÉSUMÉ [Shahar and Musen, 1993], that can infer from time-stamped patient data the presence of higher-level, interval-based, abstract concepts; a *temporal query system*, called Chronus, that can perform time-oriented queries on a time-oriented patient database; and the *domain-specific knowledge bases* (i.e. protocol specifications) required by all the other components. Similar to MLMs, EON provides recommendations about the treatment of patients, and allows for commenting on the interventions performed by the practitioner if they differ from those recommended by the protocol.

The elementary knowledge structures used within EON for the representation of a protocol are: procedures, protocol steps, intervention states, eligibility criteria, selection conditions, intervention rules, and revision rules. *Procedures* are used to implement the clinical algorithm underlying a protocol. *Protocol steps* may be intervention steps that specify an intervention or they may be assessment steps that specify data to be collected to assess or diagnose a patient. Intervention and assessment steps may also be combined within one protocol step. *Intervention states* are types of interventions, which may be in one of the states active, completed, aborted, or suspended to indicate the progress of the intervention. *Eligibility criteria* allow the identification of patients to whom a protocol may be applied and are represented explicitly within an EON protocol. *Selection conditions* define when to proceed from one protocol step to the next and which protocol step to choose from a set of alternatives. *Intervention rules* define the domain-specific attributes of protocols. *Revision rules* are instructions that define how to change the state of an intervention or modify attribute values such as the dose of a prescribed drug.

EON protocols have a hierarchical structure, where each protocol may be decomposed into a set of finer-granularity protocols. A protocol is composed of a declarative and a

procedural component: The declarative component defines parts of the protocol, their properties, and the relationships among them, whereas the procedural component specifies the temporal sequencing, branching, and looping of prescribed or suggested treatment interventions. The declarative part is modeled by a so-called ontology, which provides specific classes of protocols for different clinical domains. The procedural part is defined and visualized by means of a directed graph and an associated execution model.

Examples of protocols implemented in EON are clinical-trial protocols for breast-cancer treatment and protocols for the management of AIDS patients within the T-HELPER system [Musen et al., 1992].

2.6.3 The GLIF Approach

The GuideLine Interchange Format (GLIF) is a model for representing protocols in a machine-readable format [Ohno-Machado et al., 1998]. The main goal in the development of GLIF was to create a standard for representing protocols that facilitates protocol sharing across the software tools at different medical institutions that manipulate, analyze, or otherwise compute with an electronic representation of a health-care protocol. GLIF is the result of a joined effort of researchers from three different US universities (Columbia, Harvard and Stanford), called the InterMed Collaboratory. The basis of GLIF was provided by the analysis of four existing systems: MLM/Arden [Hripcsak et al., 1994], EON [Musen et al., 1996; Tu and Musen, 1996], MBTA [Barnes and Barnett, 1995], and GEODE-CM [Stoufflet et al., 1996].

The GLIF approach defines a specific data model, which comprises a set of classes for guideline entities, attributes of those classes, and data types for the attribute values. The topmost object of the model, which is the GLIF guideline object, contains a name, a list of authors, a characterization of the guideline's intention, a specification of the patient-eligibility criteria, an unordered list of all steps in the guideline, an indication of the starting step in the guideline, and a list of supporting didactic material. This syntax is based on a generic data interchange format called ODIF standing for Object Data Interchange Format [Pattison-Gordon, 1996], which allows the representation of object-instances in text. Sequences of decisions and actions can be explicitly defined within GLIF by specifying a collection of steps for a guideline, and by defining the starting step thereof. Guideline steps can be either *action steps* or *decision steps*, the latter being divided into *conditional steps*, *branch steps*, and *synchronization steps*. Beginning with the starting step, each guideline step comprises its successor step(s) within the specified sequence.

Examples of guidelines modeled with GLIF are guidelines for influenza vaccination, for cholesterol screening and management, for breast-mass workup, and for breast-cancer treatment protocol.

Chapter 3

Visualization of Temporal Information

The core issue of this thesis is the representation of temporal aspects in planning. In this chapter, we give an overview of how temporal information can be visualized. We present one basic concept, called timelines, and its adaptations in the medical domain. Finally, we discuss in detail the concept of SOPOs that we chose for our own visualization approach.

3.1 From Timelines to LifeLines

A timeline is a linear, graphical visualization of events over time [Karam, 1994]. They are widely used in project management (e.g., MS Project) or in historical presentations. The basic idea is to draw a diagram with a horizontal time axis and then divide the space above or below the axis vertically into regions for every plan or event to be depicted. In this vertical region, a horizontal line is drawn over the time span of the corresponding event. Tufte [1983] describes this concept extensively and presents many examples.

In the medical domain, this concept was adapted to represent patients' medical records. Bui et al. [1999] present a patient record interface, called TimeLine, that establishes a hierarchy of graphical timelines based on a patient's medical problem list. There, events that represent patient data are denoted along the timelines by iconic links and graphics. Furthermore, each timeline can be expanded to reveal sub-timelines. This hierarchical decomposition enables users to view information at several levels of detail.

Another representation of patient records is presented by Plaisant et al. [1998], and is called LifeLines. Originally, it was developed at the University of Maryland within a project for the Maryland Department of Juvenile Justice to visualize personal history records [Plaisant et al., 1996]. Later on, this technique was extended to medical applications. In

LifeLines, different aspects of the record, such as medical problems, allergies, diagnosis, medications, etc., are grouped in facets and are represented as horizontal lines, while icons represent discrete events such as physician consultations, progress notes or tests. Line color and thickness can be used to illustrate relationships or significance. Zooming and filters allow users to focus on part of the information, revealing more details on demand. Figures 3.1 and 3.2 show examples of LifeLines.

An experiment was conducted to investigate the benefits of the LifeLines interface [Lindwarm et al., 1998]. Thirty-six participants used a static version of either LifeLines, a graphical interface, or a tabular representation to answer questions about a database of temporal personal history information. Results suggest that overall the LifeLines representation led to much faster response times, primarily for questions which involved interval comparisons and making intercategorical connections.

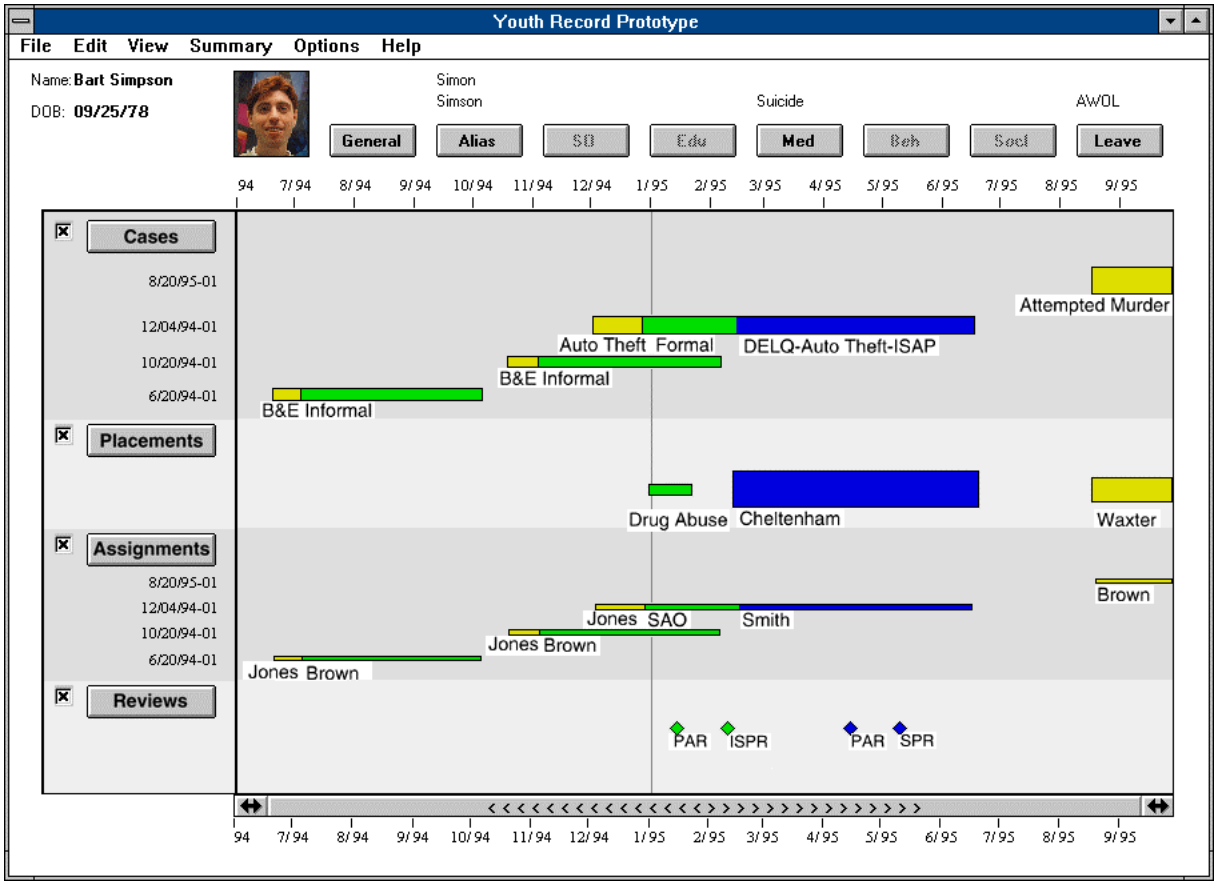


Figure 3.1: An example of LifeLines for the display of Juvenile Justice youth record. The facets are Cases, Placements, Assignments and Reviews (vertical region on the left). Line thickness indicates the severity of the offense, color the depth of penetration in the juvenile justice system [Plaisant and Shneiderman, 1997].

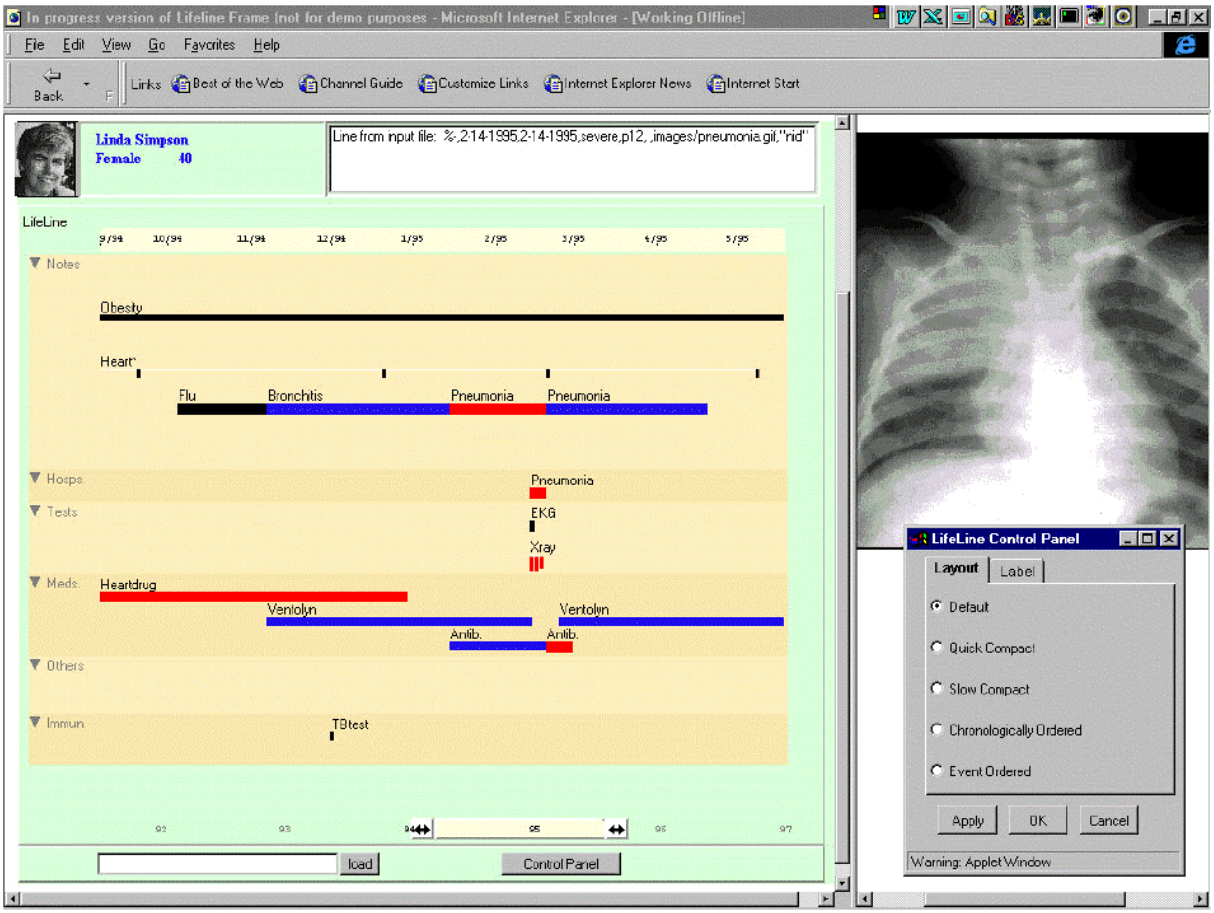


Figure 3.2: An example of LifeLines showing a patient's medical record [Plaisant et al., 1998].

3.2 AsbruView's Time Annotation Glyph

The depiction of time annotations in AsbruView is based on the idea of LifeLines (see previous section), but had to be enhanced to allow a visualization of uncertainty in starting, finishing and duration interval. As introduced in Chapter 2, a time annotation in Asbru is defined as follows:

```
[[ESS, LSS], [EFS, LFS], [MinDu, MaxDu], REFERENCE],
```

where [ESS, LSS] denotes the starting interval, [EFS, LFS] the finishing interval, and [MinDu, MaxDu] the duration interval. REFERENCE indicates the reference annotation,

e.g., a time point, from which the intervals' time shifts originate. A detailed description of the single parameters is given in section 2.2.2.

In AsbruView, a plan's time annotation is represented by a time annotation glyph. The time shifts ESS, LSS, EFS and LFS are shown as vertical lines with arrows, on which a bar representing the maximum duration (MaxDu) rests. On top of the MaxDu-bar, two diamonds are depicted carrying the second bar that represents the minimum duration (MinDu). MinDu is constrained by the time span between LSS and EFS. It cannot become shorter, as in that case, the MinDu-bar would fall down. If one of LSS or EFS is not defined, the corresponding diamond becomes a circle. This circle can move when MinDu is changed. In addition, undefined parts are colored in gray instead of black [Kosara, 1999]. In Figure 3.3, an example of a time annotation glyph is given.

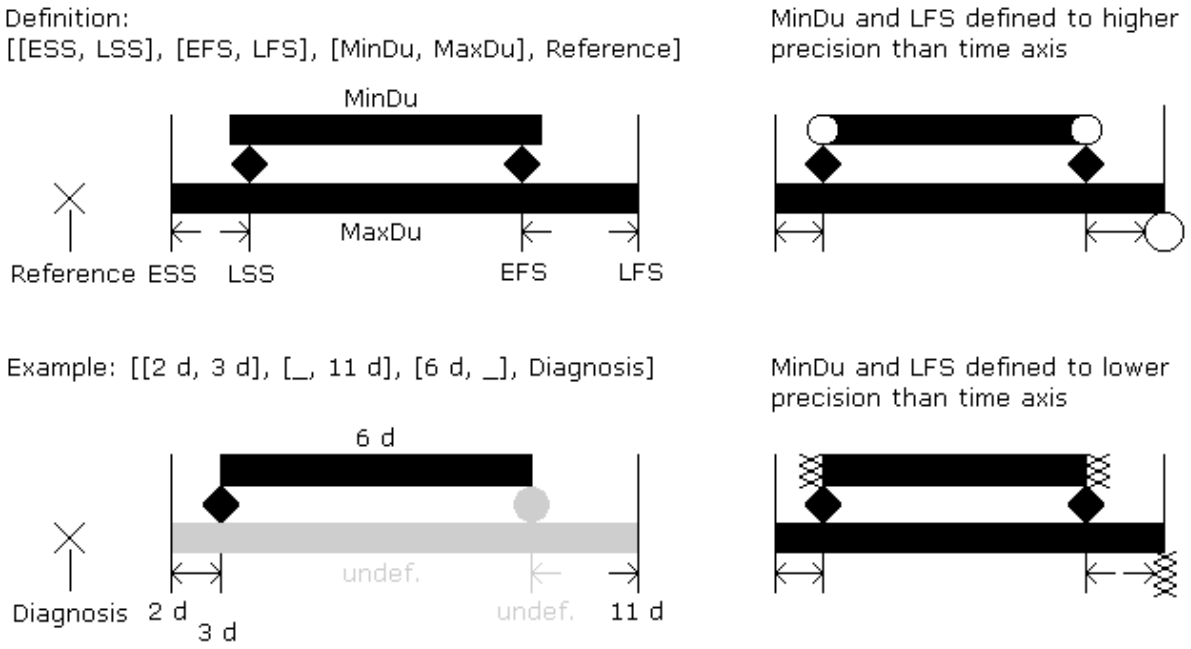


Figure 3.3: Time Annotation Glyph [Kosara and Miksch, 1999].

On the left side of Figure 3.3, the top illustrates the definition of time annotations, the bottom shows an example, which means "starts 2 to 3 days after diagnosis, ends 11 days after diagnosis at the latest and lasts for 6 days at a minimum". The maximum duration and the earliest finishing shift are undefined. The right side shows two cases where the time scale of the time annotation is not the same as that of the current time axis. In AsbruView, different time granularities have not been implemented.

3.3 SOPOs: Sets of Possible Occurrences

The graphical representation presented in the following was introduced as part of a general frame for propagating temporal constraints over events [Rit, 1986]. Events are characterized by sets of possible occurrences (SOPOs), where an occurrence is defined as a one-dimensional interval during which an event happens.

3.3.1 Occurrences

As an occurrence is a numerical interval, a timeline would be an obvious means for representing occurrences and their relations (see Figure 3.4). However, this representation is ambiguous because SOPOs have to be represented with intervals, while in fact they are sets of intervals. The interval in Figure 3.4 for example could represent the set of occurrences beginning after 3 and ending before 6, or the set of such occurrences with a duration equal to 3, or even the set of occurrences whose beginning cannot belong to [4, 5]. This ambiguity comes from the two-dimensional nature of an occurrence - beginning and end - and leads to the two-dimensional representation shown in Figure 3.5.

As occurrences begin before they end, all possible ones belong to the gray region of Figure 3.5. In our example, the occurrence *o* is clearly identified within the diagram. The diagonal axis has in addition a particular meaning: it is the locus of occurrences whose beginning equals the end, thus the locus of dates (i.e. time points with zero duration). This links the two-dimensional representation with the one-dimensional one: the beginning and the end of any occurrence *o* are dates and can be projected onto the diagonal, defining a segment. This segment is then the one-dimensional representation of *o*.

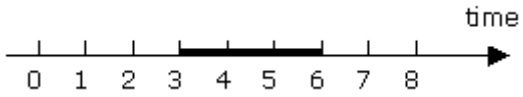


Figure 3.4: One-dimensional representation of occurrences.

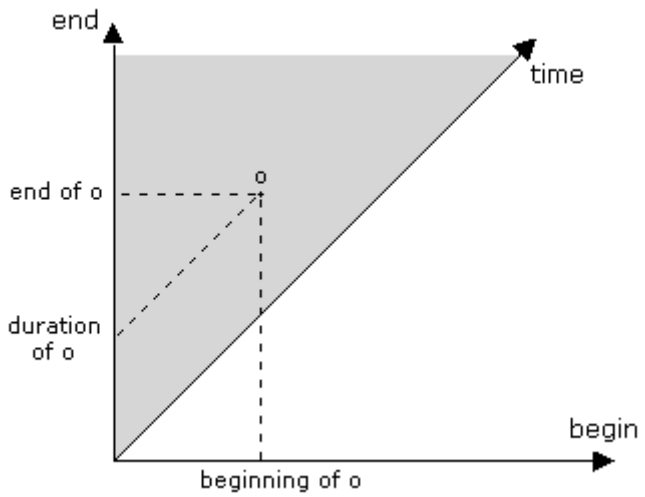


Figure 3.5: Two-dimensional representation of occurrences.

3.3.2 Representation of Relations

Using the two-dimensional diagram with a beginning and an end axis, one can represent temporal relations between different occurrences (intervals).

Allen [1983] has proposed a framework for temporal reasoning - the interval-based temporal logic. The only ontological temporal primitives in Allen's logic are intervals. Allen's motivation was to express natural-language sentences and to represent plans. Allen has defined 13 basic binary relations between time intervals (six of which are inverses of the other six): *before*, *after*, *overlaps*, *overlapped*, *starts*, *started by*, *finishes*, *finished by*, *during*, *contains*, *meets*, *met by*, and *equal to*.

Figure 3.6 summarizes the set of these temporal relations. The regions and lines are shown from which an occurrence must originate in order to be in a certain relation to occurrence *o*. Figure 3.7 shows an example of allowed regions, marked in gray, for occurrences, that happen before and after occurrence *o*.

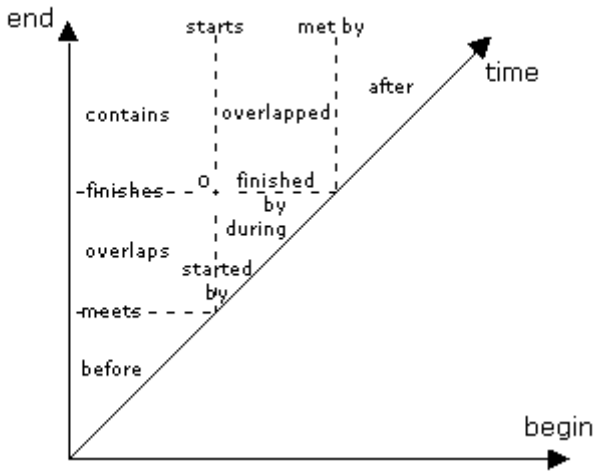


Figure 3.6: Two-dimensional representation of Allen's relations [Duftschmid, 1999].

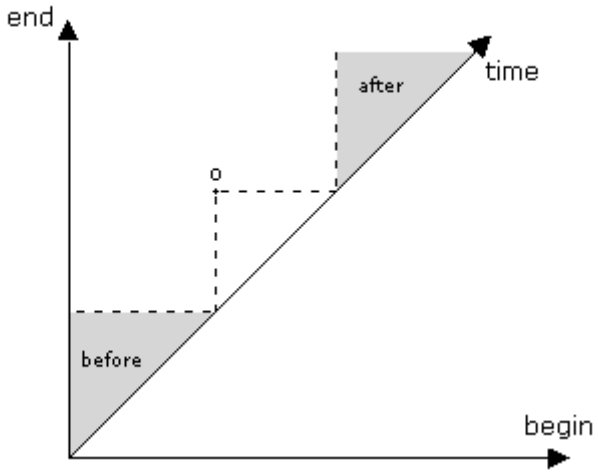


Figure 3.7: An example of allowed regions for occurrences that happen before and after a certain occurrence *o*.

3.3.3 Visualization of Asbru's Time Annotations

Rit's two-dimensional representation of temporal relations can be used to visualize Asbru's time annotations. Consider the following example of a time annotation in Asbru:

```
[[1 WEEKS, 3 WEEKS], [3 WEEKS, 6 WEEKS], [1 WEEKS, 4 WEEKS], CONCEPTION],
```

which means "starts 1 to 3 weeks after conception, ends 3 to 6 weeks after conception and lasts 1 to 4 weeks". A one-dimensional visualization of this time annotation is given in Figure 3.8.

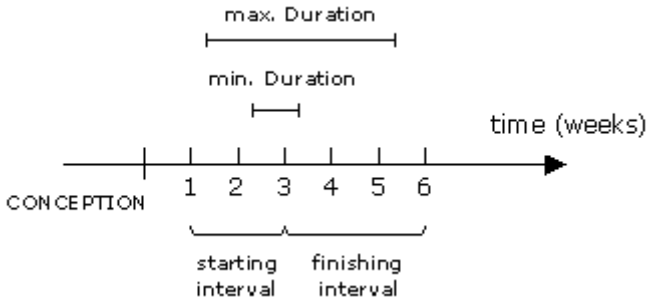


Figure 3.8: One-dimensional visualization of a time annotation.

While the starting interval [ESS, LSS] and the finishing interval [EFS, LFS] of the time annotation can be exactly indicated, a problem arises regarding the representation of the duration intervals. Any fixed positioning of the two duration intervals (as shown in Figure 3.8), will only allow the representation of a certain occurrence out of the complete set of allowed time intervals. In order to solve this visualization problem, the time annotation can be considered as a SOPO and thus be drawn within the two-dimensional diagram (see Figure 3.9).

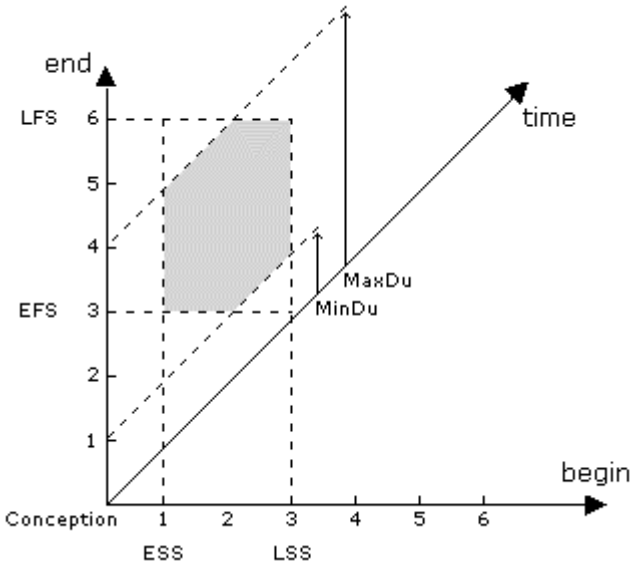


Figure 3.9: Two-dimensional visualization of a time annotation.

The gray region in Figure 3.9 defines the SOPO for the above time annotation. Any point inside the SOPO defines an allowed time interval, i.e. it can be considered as an occurrence as shown in Figure 3.5, with an exact beginning in [ESS=1, LSS=3], end in [EFS=3, LFS=6] and duration in [MinDu=1, MaxDu=4]. The reference point of the time annotation is, in our example, CONCEPTION - assumed to reside in the origin of the diagram. As the values for ESS, LSS, EFS and LFS are quoted relatively to the reference annotation, they can be directly applied to the horizontal and vertical axis. The duration constraints are incorporated in such a way that two parallel lines are inserted with a distance of MinDu and MaxDu from the diagonal axis. Graphically, they cut off the upper left and lower righthand corners of the rectangle (ESS, LSS, EFS, LFS).

3.3.4 Examples of Sequential and Parallel SOPOs

Figure 3.10 shows two SOPOs representing plans that are to be executed in sequence. We define TA1 as the first plan's time annotation, and TA2 as the second plan's time annotation. In Asbru notation, they are as follows (all in the same time unit):

```
TA1:  [[1, 3], [3, 5], [1, 3], 0].
TA2:  [[3, 5], [5, 7], [1, 4], 0].
```

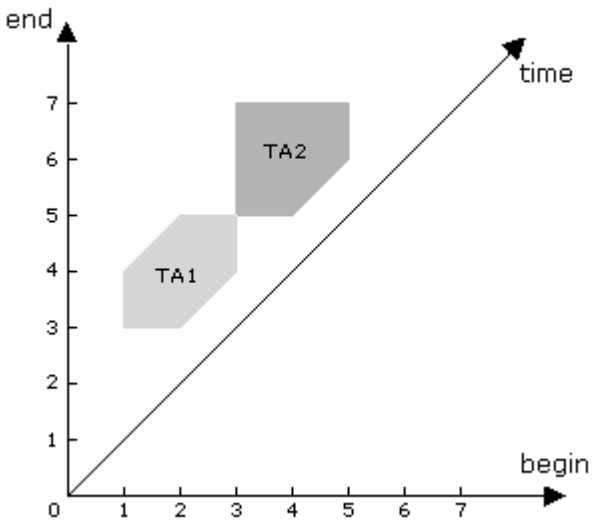


Figure 3.10: Example of Sequential SOPOs.

Figure 3.11 shows two SOPOs representing concurrent plans. Their time annotations, TA1 and TA2, are in Asbru notation as follows:

TA1: `[[1, 3], [3, 5], [1, 3], 0].`
TA2: `[[1, 3], [5, 7], [3, 6], 0].`

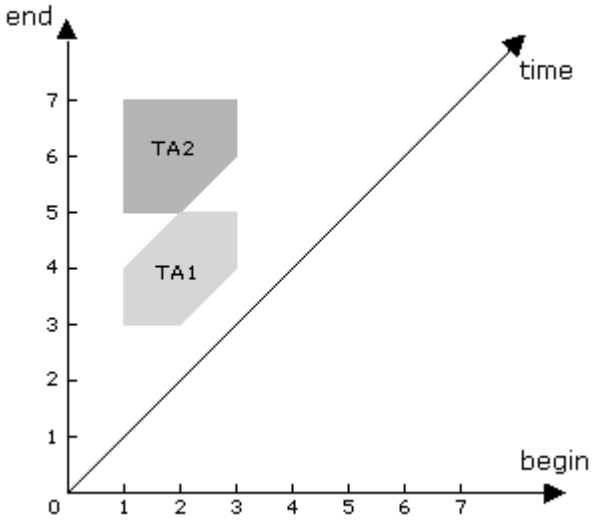


Figure 3.11: Example of Concurrent SOPOs.

Figure 3.12 shows two SOPOs representing concurrent plans whose finishing intervals overlap. Their time annotations, TA1 and TA2, are in Asbru notation as follows:

TA1: `[[1, 3], [3, 5], [1, 3], 0].`
TA2: `[[1, 3], [4, 6], [2, 5], 0].`

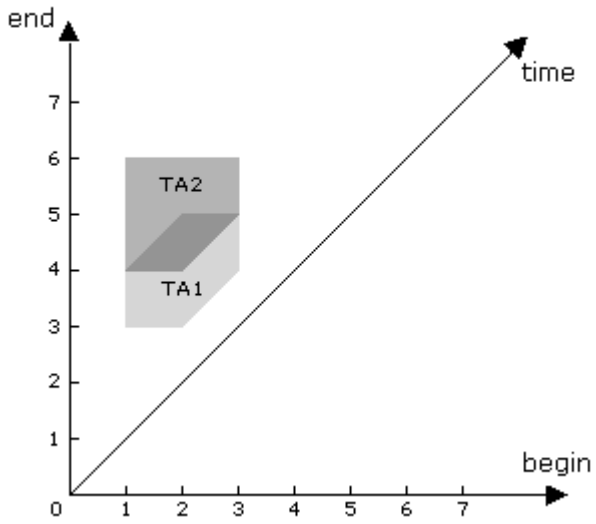


Figure 3.12: Example of Concurrent SOPOs with overlapping.

Chapter 4

User Interface Design and Usability

From our initial idea to the running of the software program, many steps had to be taken to be successful at the end. In this chapter, we present the basic stages of such an engineering process, focussing on the user interface and its usability.

4.1 The User Interface

The human-computer interface, or simply user interface, represents the only part of an application system with which the end-users come into direct contact [Ravden and Johnson, 1989]. One could say that for the user, the interface *is* the system. Whether the human can "use" the system or not therefore depends on that interface. If end-users feel uncomfortable using the system; if it causes confusion or frustration - either because it interferes with their work, or it takes much more time to carry out tasks - then they may find it inefficient to use, and finally stop using it altogether. Thus, designing a user interface that meets the end-users' requirements is *the* key-success factor. However, ...

... User interface design is a very difficult business. It combines two awkward disciplines: psychology and computer science. These disciplines have very different cultural backgrounds: psychology is concerned with people; computer science with computer machinery. Psychologists are supposedly sympathetic and understanding; computer scientists are supposedly mathematical and precise. Psychologists have enough trouble understanding people even when they are not using computers; computer scientists have enough trouble getting programs to work even when they are not being used by people. Good user interface design requires these two perspectives to be united. ... [Thimbleby, 1990].

One approach that not only takes those two disciplines but also some others (e.g., sociology or linguistics) into account, is called usability engineering. Before we present the stages of this lifecycle model, we will look at the term usability and its meanings.

4.2 Usability

Today, computers are not only tools for computer scientists, hackers or freaks. They have begun to play a significant role of daily applications, thus enabling users without the background of computer scientists to benefit from them. On our way into the information society, where bits and bytes will finally dominate over atoms (which characterize the industrial age we have begun to leave), usage of computers and information technology, like the Internet, increases almost day by day. Visionaries like Nicholas Negroponte draw an image of our future, in which computers play an ubiquitous part in our life [Negroponte, 1995].

Often, *user-friendliness* is mentioned when it comes to discussions about requirements and prerequisites. However, the expression *user-friendliness* is somehow improper as it first suggests a computer or system being able to be friendly (whatever that means), and secondly suggests only a one-dimensional classification. Instead, the expression *usability* was introduced which shall help us to consider the many dimensions we are talking about.

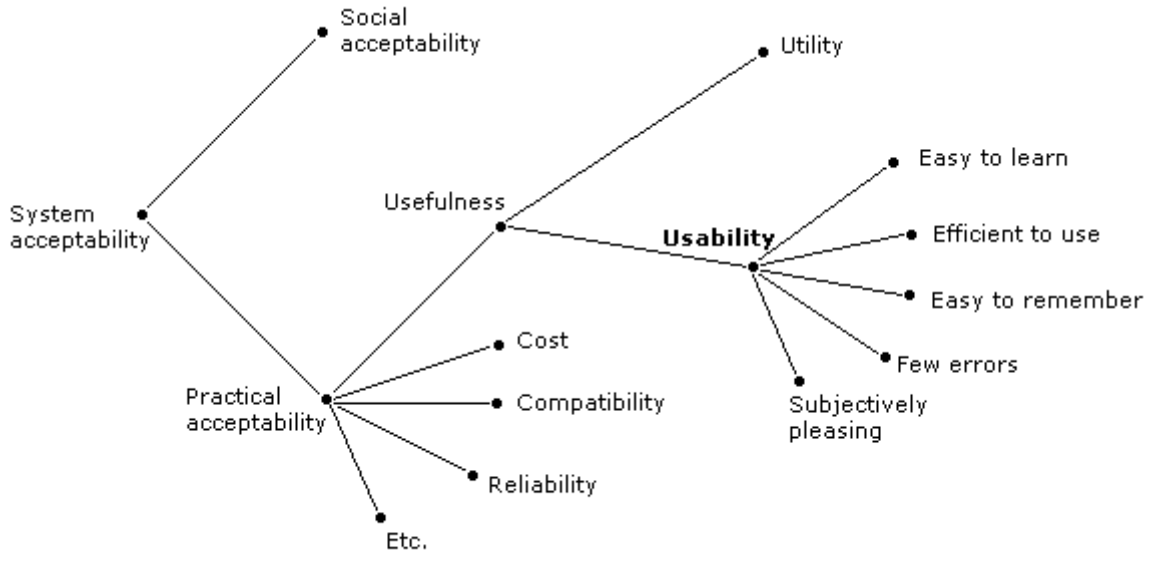


Figure 4.1: A model of the attributes of system acceptability [Nielsen, 1993].

The question whether a certain system satisfies all the various needs and requirements of the users and of other people involved, like users' clients, or managers, is not that of usability but more that of system acceptability as a whole. Usability itself is only part of that higher order goal. Figure 4.1 shows all the attributes of system acceptability, including usability. Their usefulness, for example, determines whether or not a system can be used to achieve some desired goal. It comprises utility, the question of whether the system's functionality provides what is needed, and usability, which is the question of how well users can access that functionality [Nielsen, 1993].

Following this model, usability is not a single, one-dimensional property of a user interface, but has several components. We define the following five usability attributes [Nielsen, 1993], that are sometimes also addressed as goals of the user-interface design [Shneiderman, 1998]:

- **Learnability**: the system should be easy to learn and the user should be able to carry out some basic tasks after a short period of time.

- **Efficiency of Use**: once the user learned how to use the system, tasks should be carried out productively.

- **Memorability**: this attribute addresses the retention over time. Casual users should remember how to use the system without having to learn everything again.

- **Few and Noncatastrophic Errors**: during work with the system, a few errors may occur, and the user should be able to recover from them.

- **Subjective Satisfaction**: the system should be pleasant to use. That does not mean that users have to be overwhelmingly happy when using it. They simply should like it.

Not all of those attributes need to have the same importance in software projects. That largely depends on the usability goals specified, due to user characteristics, their needs and requirements and the tasks they will carry out using the system. In designing our user interface SOPOView, the most important attributes were efficiency of use and a low error rate, as well as subjective satisfaction. In the medical domain, physicians have to work efficiently when using our program, and errors have to be prevented as consistently as possible, for they lead to delays during treatment. On the other hand, it probably

takes more time to learn how to use our program, in particular because the representation of temporal data in the diagram is unconventional and complex, and takes time to become familiar with.

4.3 The Design Process

Designing the user interface is not a one-shot affair; it is a process that starts with an initial analysis and finishes with a successful implementation. Basically, software engineering is the term that denotes this process. It was first introduced at the NATO Science Committee Conference in 1968. There, software engineering was defined as *"the establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines"* (Fritz Bauer).

In the traditional view, often referred to as the waterfall model [Sommerville, 1992], stages of the process were arranged in a linear fashion; in this manner the output of each stage became the input of the next. For example, requirements for a software were gathered at the beginning and used in the design, which was then coded and tested. When the product was completed, it was tested again and maintained until being eliminated or replaced. It soon became apparent that stages overlap and that information must flow in both directions, while during design problems with requirements were found, and during coding problems with design were found. An adaptation and enhancement to the waterfall model is the spiral model [Boehm, 1988], in which several iterations are required and in which a prototyping approach is used to better understand the requirements for the following stage.

4.4 User-Centered Design

Considering those two models, the waterfall and the spiral model, users are not really involved throughout the design process. They may be consulted during design phase, and they are of course necessary to carry out tests, but they obviously do not play the central role in development. In user-centered design, user issues become central to the design process, early testing and evaluation are carried out with users, and the whole process is highly iterative.

Many design methods have been evolved in this field of human-computer interaction (HCI). This field of study does not focus on just the design of user interfaces, but on all

the aspects that relate to the interaction between users and computers. Among others, approaches like the Soft Systems Methodology (SSM) [Checkland and Scholes, 1990], where the emphasis is on understanding the situation in which a problem is thought to lie, and not on finding a solution, or the Open Systems Task Analysis (OSTA) [Eason and Harker, 1989], where the focus is on understanding both the social and technical system and the transformation that occurs when introducing a computer system into a working environment, became very popular.

Our own approach in designing SOPOView is based on the star model (see Figure 4.2), in which no exact ordering of activities exists and in which evaluation is central. All aspects of user-interface design are subject to evaluation during the whole development process [Hix and Hartson, 1993]. In the star lifecycle approach, system development may begin in any of the stages shown in Figure 4.2. As the whole process is iterative, requirements, design, and the product gradually evolve over time, becoming well defined in the end. Basically, the star model's stages are equivalent to the stages of the usability engineering lifecycle model [Nielsen, 1993].

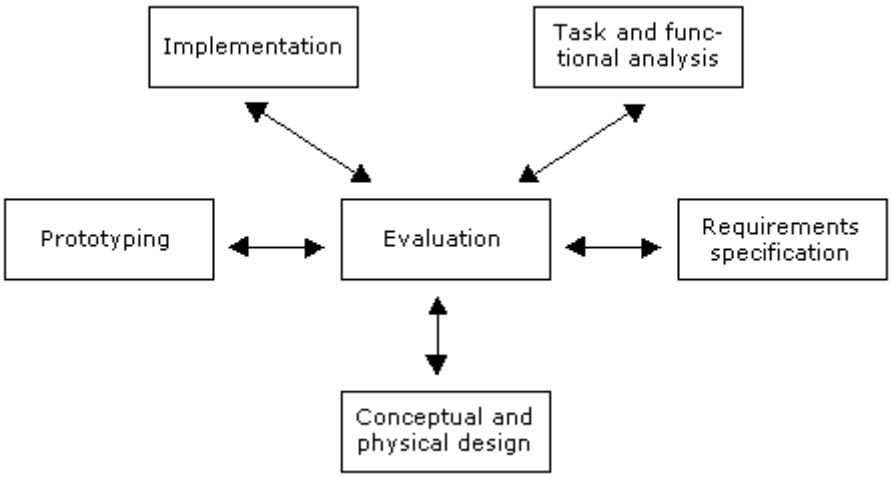


Figure 4.2: The star model [Hix and Hartson, 1993].

4.5 From Paper and Pencil to SOPOView

Designing our own solution, called SOPOView, was not intended to be a closed process with an implemented product for use at its end. Much more, it was part of the evolution of the user interface AsbruView, where we were looking for an alternative representation of temporal data. With AsbruView, we are still at the prototyping stage as a number of

intended features of the program, due to functional analysis, are not available at the moment. Continuous evaluation helps us to find hidden requirements and to further develop our program.

In developing SOPOView, we started with the concept of SOPOs, which was introduced in the previous chapter, and made early paper mockups to visualize the ideas of our conceptual design. The conceptual design concerned with questions of what is required. As the original concept [Rit, 1986] did not meet all the requirements from the language Asbru, we had to adapt and enhance it. A detailed description is given in the next chapter. The other requirements for our representation were derived from the results of the evaluation of AsbruView [Kosara, 1999].

After we had covered almost all requirements in our paper mockups, we started to implement a first prototype. Its aim was to find an appropriate means to visualize SOPOs and to define its parameters. Later, techniques for the overlapping of SOPOs were developed and the basic user interaction possibilities, like the dragging of SOPOs, were implemented and refined. For reasons of simplicity and flexibility, this first prototype was independent from AsbruView. After incorporating SOPOView into AsbruView, we conducted some tests within the team where we found a few incompatibilities and problems in terms of visualization, which could be fixed and solved within a short period of time. Our program was then ready for evaluation with physicians.

4.6 Evaluation

As shown in the star model (Figure 4.2), evaluation is central in the design process. Our own evaluation and the results will be presented in detail in Chapter 6, but some fundamental aspects should already be mentioned here. What is evaluation about? The following definition was taken from Preece [1994]:

Evaluation is concerned with gathering data about the usability of a design or product by a specified group of users for a particular activity within a specified environment or work context.

As one can see, evaluation does not aim to solve problems, but rather provides a means of identifying problem areas, difficulties, weaknesses, or areas of possible improvement. In doing our evaluation, we wanted to find out what physicians want and what problems they experience when carrying out tasks with our prototype. Our evaluation was a sum-

mative one, i.e. that it took place after the product, our prototype, has been developed (in contrast to a formative one, where evaluation meshes closely with design and guides design by providing feedback). When conducting a summative evaluation, sometimes also referred to as usability testing [Shneiderman, 1998], various techniques can be used, of which some shall be mentioned here briefly.

Mostly, evaluation depends on some kind of *observation or monitoring* of users' interactions. The observer collects data by making notes, or some other form of recording, like videotaping, may be used. That strongly depends on where the evaluation takes place. In a usability laboratory, video recording and keystroke logging can be done relatively easily, compared to the users' normal work environment, where the use of those methods will be restricted. A supporting technique during evaluation is the *thinking aloud* method [Nielsen, 1993]. Essentially, the test person uses the system while continuously thinking out loud. By verbalizing their thoughts, users enable the observer to understand how they view the system. Finally, users' subjective opinions about the system can be gathered by *questionnaires* and during *interviews*.

During our evaluation, we gathered data by observing the physicians while they were working with our prototype. In between, we asked questions about why they interacted with the system in a certain way to force them to think aloud. The evaluation concluded with a questionnaire (see Appendix A) and a final interview.

Chapter 5

SOPOView

This chapter presents the core part of this thesis, namely the implementation of the visualization of temporal aspects of plans. We call our solution SOPOView, due to the fact that it first uses the concept of SOPOs to visualize temporal relations (see section 3.3), and is then implemented as an additional view within AsbruView (see section 2.4).

In the next sections, we introduce the basic parts and layout of SOPOView. We describe the part representing the structure of plans as well as the diagram used for the visualization of SOPOs. As the basic concept of SOPOs had to be adapted in order to fulfill the requirements of the language Asbru, and in order to enable an acceptable visualization, all those enhancements will be discussed in detail.

Figure 5.1 shows a screenshot of SOPOView within the AsbruView program. The example of the plan is the same as the one introduced in Chapter 2, which is the treatment of infants' respiratory distress syndrome (I-RDS).

5.1 Left Part: Plan Structure

A plan here is represented as a colored rectangle that may contain any number of other rectangles (sub-plans). If a plan has sub-plans, a small black triangle is drawn on the left, pointing to the right if the plan is closed, or pointing down if the plan is opened (i.e. its sub-plans are shown). The plan's name appears to the right of the triangle. Below the triangle, symbols indicate the plan type, i.e. the way its sub-plans are to be performed. To the right of those symbols, the rectangles representing the sub-plans begin. This leads to a kind of tree structure that is quite widespread in standard software systems today. Examples of the different symbols for the plan types are shown in Figure 5.2 and are summarized in Table 5.1.

Plan Types	Asbru Notation	Symbols used
Sequential Plan	Do-All-Sequentially, Do-Some-Sequentially	Bullets (like in a bullet list)
Parallel Plan	Do-All-Together, Do-Some-Together	Two parallel lines
Any-Order Plan	Do-All-Any-Order, Do-Some-Any-Order	Two arrows pointing into opposite directions
Cyclical Plan	Do-Every	Circular arrow

Table 5.1: Plan Types and their representation.

Optional plans are indicated by a question-mark texture, as it is the case in the Topological and Temporal View (see section 2.4.4).

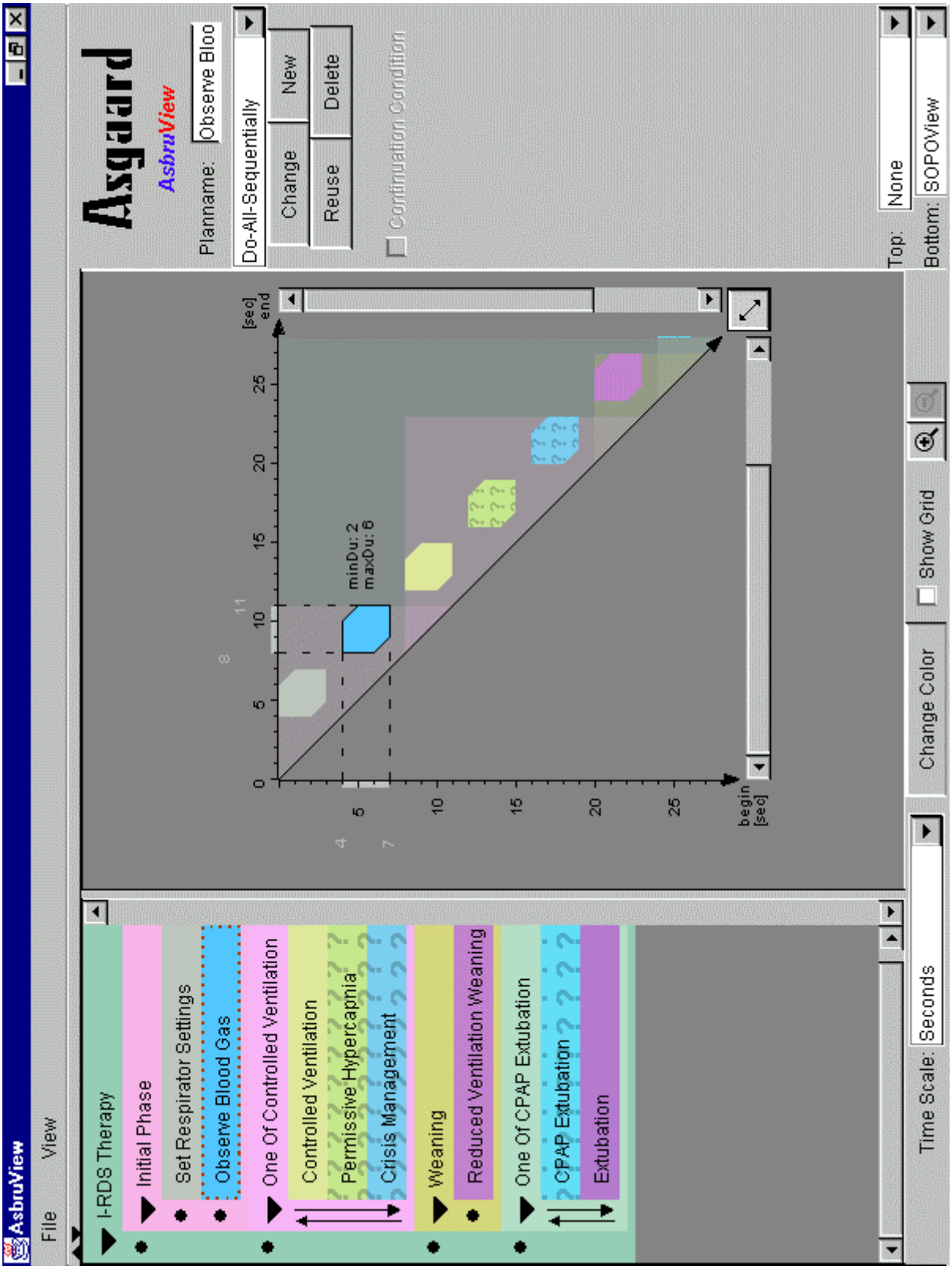
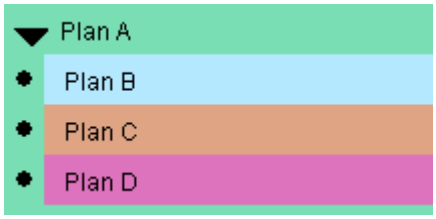
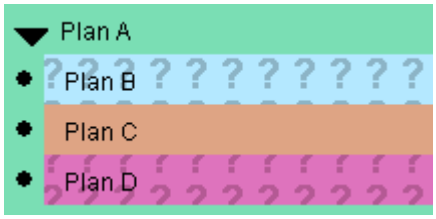


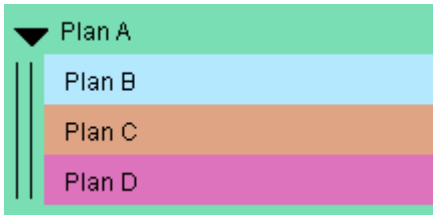
Figure 5.1: Screenshot from the AsbruView program showing SOPOView. The example plan is the treatment for infants' respiratory distress syndrome (I-RDS). The left/lower part shows the structure of the plan, the right/upper part shows the sub-plans' time annotations using SOPOs.



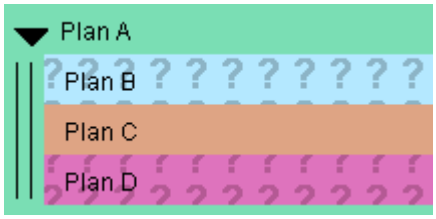
Do-All-Sequentially



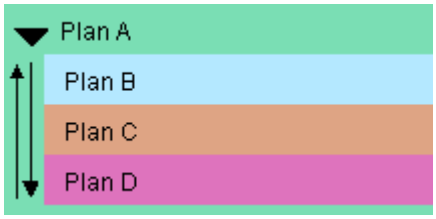
Do-Some-Sequentially



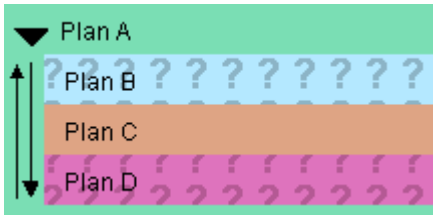
Do-All-Together



Do-Some-Together



Do-All-Any-Order



Do-Some-Any-Order

Do-Every

Figure 5.2: All of Asbru's plan types in SOPOView's left part.

5.2 Right Part: SOPO Diagram

Here, the concept of SOPOs introduced in Chapter 3 is used to represent the temporal aspects, i.e. the time annotations of plans written in Asbru.

5.2.1 Adaptations to the concept of SOPOs

In order to meet all the requirements from the language Asbru, the basic concept of SOPOs as introduced in Chapter 3 had to be enhanced. Still, a plan's time annotation is depicted as a colored region within the diagram (that is the SOPO) as it was shown in Figure 3.9. However, a hierarchical decomposition, i.e. the way plans are made up of sub-plans, and so forth, was not covered by the representation presented by Rit [1986], and therefore had to be added.

5.2.1.1 Colors

Since every plan's time annotation is represented as an area in the diagram (SOPO), it is assigned the same color that the plan itself has in the other views. Thus, identification of a plan's SOPO should be easy. However, problems may arise for color-deficient users.

5.2.1.2 Position of Axes

The way the three axes of the diagram are drawn is the first adaptation to Rit's representation. There, the horizontal axis marked the beginning in time, while the vertical axis marked the end in time. The origin of the two axes is found in the left bottom (see Figure 5.3). In SOPOView, the arrangement is rotated by 90 degrees to the right: the horizontal axis markes the end in time and the vertical axis the beginning in time. The origin of the two axes is found in the top left (see Figure 5.4).

There were two reasons for the change in position of the axes: first, the common way of scrolling on a screen is from left to right and from top to bottom. If the number of plans (and thus of SOPOs) to be drawn exceeded the available area of display, the user had to scroll from left to right, but also from bottom to top, as the SOPOs are arranged along the diagonal axis that starts in the bottom left and goes to the top right. The other

reason is that if new sub-plans are added the left part of SOPOView must grow toward the bottom, as this is the way plans are depicted there. As mentioned before, however, the SOPO diagram would grow toward the top right - that is, in the opposite direction. For example, when finding a certain plan in the left part, the user would have to scroll down, but at the same time scroll up and right within the SOPO diagram. We simply wanted the user to be able to scroll in the same direction in both parts of SOPOView, to reduce both confusion and inconsistency.

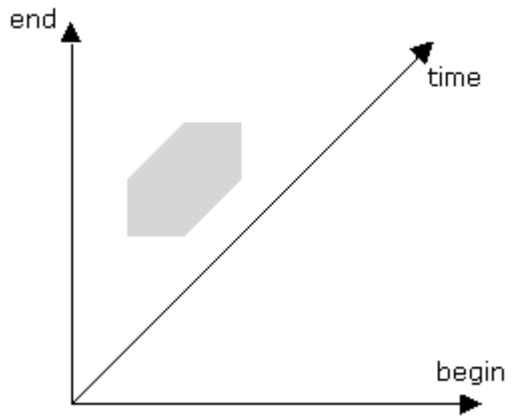


Figure 5.3: Axes position by Rit [1986].

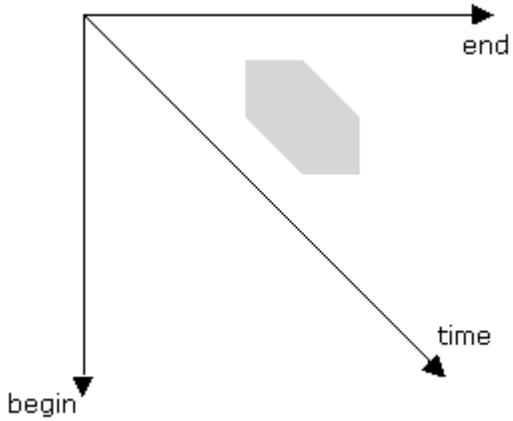


Figure 5.4: Axes position in SOPOView.

5.2.1.3 Hierarchical Decomposition

Plans may consist of sub-plans, which themselves may consist of other sub-plans, and so forth. A plan's time annotation is represented by a SOPO in the diagram. A way had to be found to visualize the hierarchy of SOPOs, since a plan's time annotation is composed of its sub-plans' time annotations. This is done by using background colors.

Figures 5.5 to 5.7 show an example, where the plan hierarchy is as follows: Plan A consists of the plans B, C and D, that are to be executed in sequence. Plan C is composed of two sub-plans, E and F, that are also executed in sequence. Plan D consists of the two sub-plans G and H, that are to be performed in parallel. The figures show screenshots of SOPOView.

In Figure 5.5, Plan E is marked, which is indicated by a dotted border around the colored rectangle in the left part (in AsbruView, this indication is called "ants trail" because the dots move around the rectangle), and by the marked SOPO with its parameters explicitly shown in the diagram. Plan E and F are the sub-plans of plan C, which is easy to grasp in the left part of SOPOView. In the diagram, the two plans' SOPOs can be found within a

colored triangle, which indicates not only their unity within plan C, but also plan C's earliest starting and latest finishing shift. The color of this triangle, however, is not the same as plan C's color in the left part.
First, this is due to the fact that we wanted to signalize the level of detail in the diagram. Plan C is opened, i.e. its sub-plans E and F are shown, and thus plan C shall move into the background. This is done by still using the same color, but semi-transparent. Second, we only used the plan color for the plan's SOPO. The background triangle, however, is not the plan's SOPO.
In order to get plan C's time annotations, the user simply has to mark plan C in the left part or click within the background triangle in the diagram. This would lead to the representation shown in Figure 5.6. Plan C is marked and its SOPO is shown in the diagram. One can see that the background triangle's horizontal edge corresponds to the plan's earliest starting shift (ESS, that is 4 seconds in the example), and that the background triangle's vertical edge corresponds to the plan's latest finishing shift (LFS, that is 15 seconds in the example). Within these two temporal shifts, all of the time annotations of plan C's sub-plans (the SOPOs), had to be found. In Figure 5.7, the closed plan C is shown, i.e. its sub-plans (and SOPOs) are not visible in both left part and diagram of SOPOView. The opening and closing of plans will be discussed in detail in section 5.3.

5.2.1.4 Undefined Parameters

In Asbru, it is possible to leave parts of a plan's time annotation undefined (see section 2.2.2). Thus, a way had to be found to visualize undefined parameters of SOPOs. However, undefined parameters mean that the according edges of the SOPO are unknown. The solution we chose was to assign the undefined parameter a certain value that is necessary to draw the according edge of the SOPO, but to use a dashed line instead. Figure 5.8 shows an example of one marked and one unmarked SOPO with undefined parameters. Still, this visualization was unsatisfactory, as the supposed values imply a certain width of the temporal interval in the diagram that does not exist. Further, depending on the colors of the SOPOs, the dashed lines can be difficult to see. So far, no optimal solution has been found.

5.2.1.5 Optional Plans

In the other views, optional plans (i.e. plans that do not have to be performed) are marked by a question-mark texture (see section 2.4.4). Following this principle, SOPOs of optional plans are also filled with question-marks (see Figure 5.9).

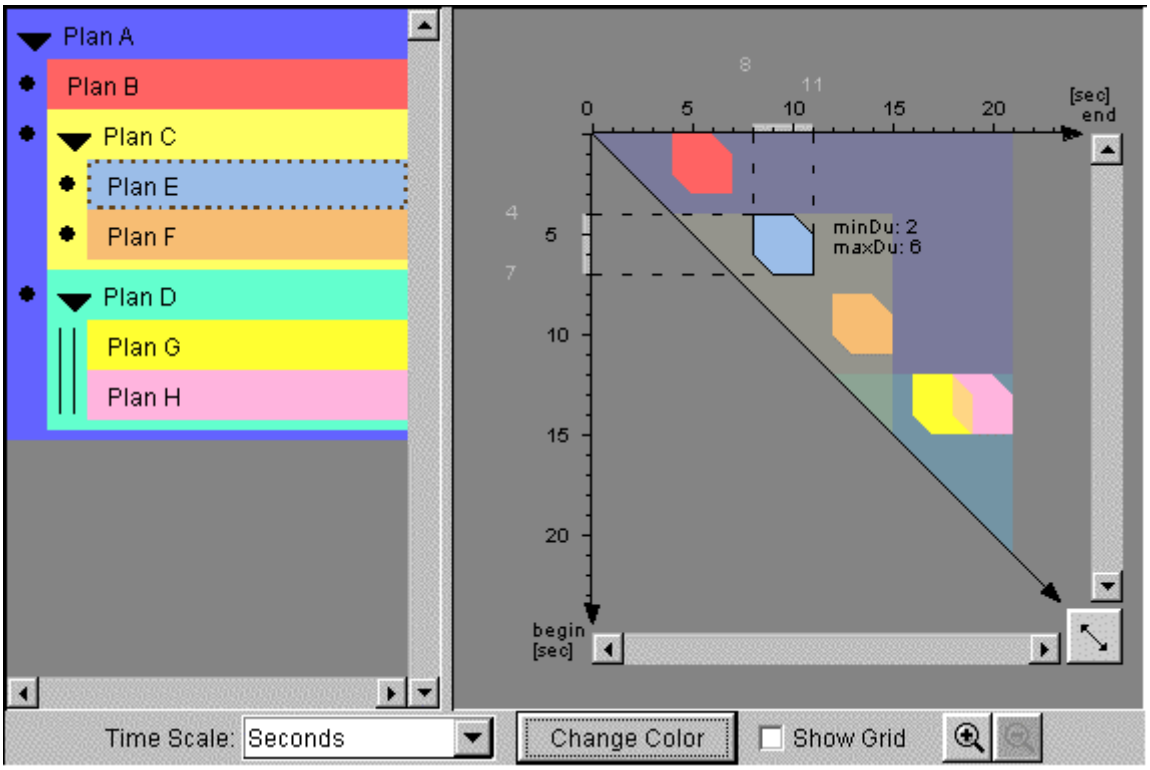


Figure 5.5: A screenshot from SOPOView, showing the hierarchical decomposition (1).

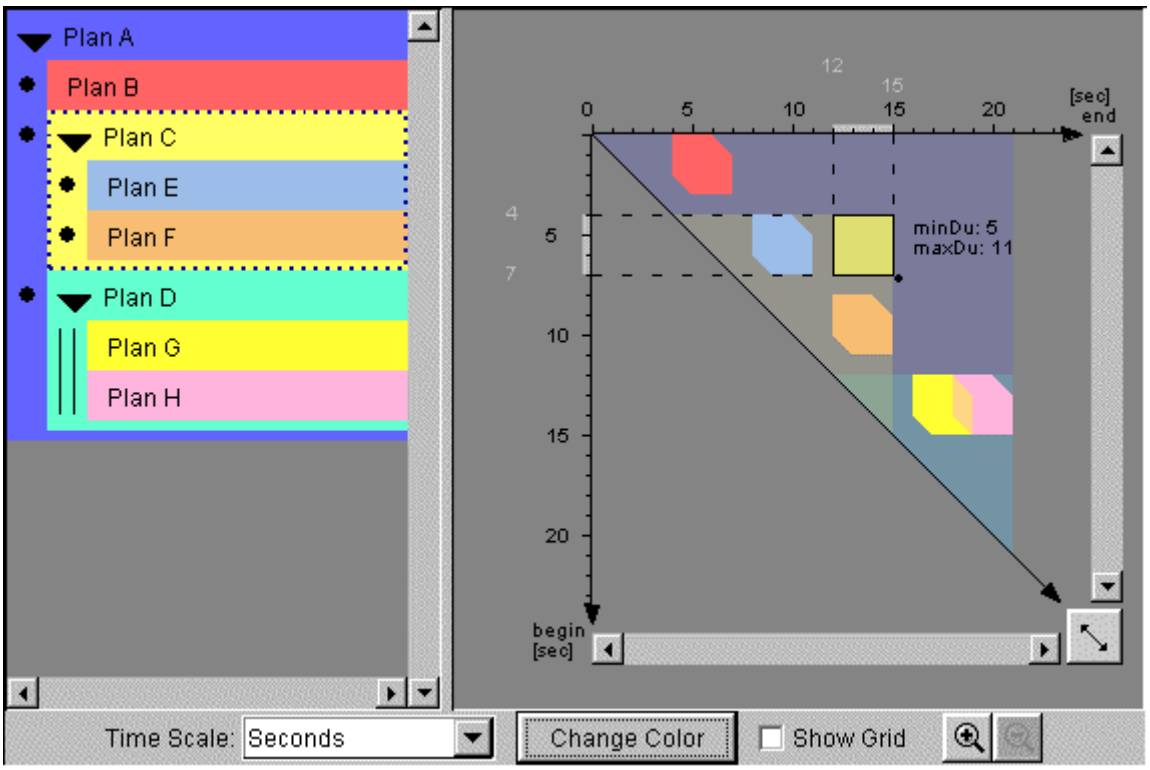


Figure 5.6: A screenshot from SOPOView, showing the hierarchical decomposition (2).

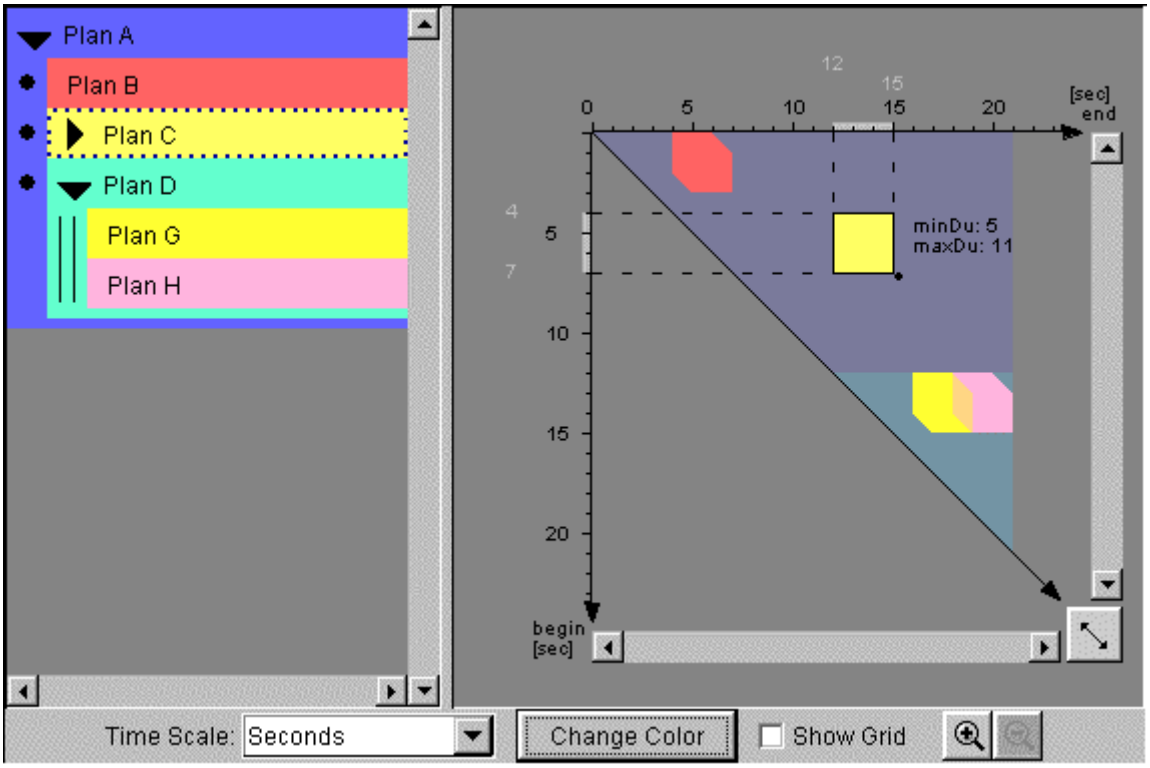


Figure 5.7: A screenshot from SOPOView, showing the hierarchical decomposition (3).

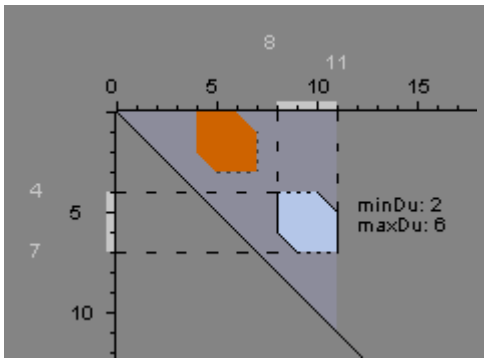


Figure 5.8: Part of the SOPO diagram showing undefined parameters.

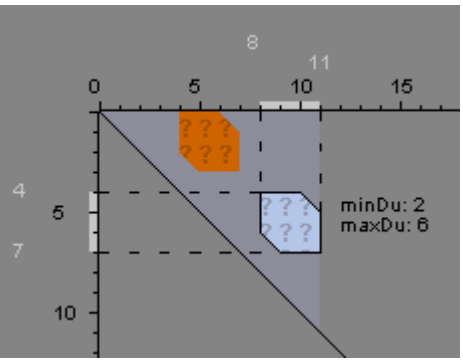


Figure 5.9: Part of the SOPO diagram showing SOPOs of optional plans.

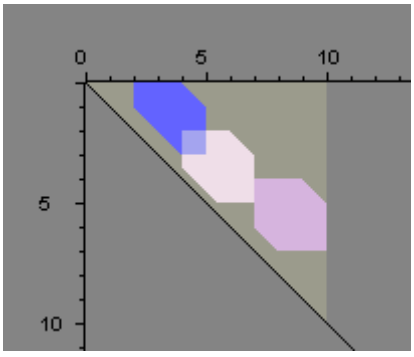


Figure 5.10: Part of the SOPO diagram showing a sequential plan's sub-plans, with overlapping.

5.2.2 Reading the Diagram

This section describes the way the user has to read the diagram in order to get to know the temporal information and relations of the plans.

5.2.2.1 Marked SOPOs

Once a plan has been marked, the corresponding SOPO also appears marked in the diagram. For an example, see Figure 5.5. There, Plan E is the current plan. In the diagram, dashed lines are drawn from both axes to the corresponding corners of the SOPO. On the axes, the temporal intervals thus created are marked by a thicker light-gray line. Further, the intervals' borders are given in light-gray numbers as well. Thus, one can easily ascertain a plan's starting interval (ESS = 4, LSS = 7, in our example) and finishing interval (EFS = 8, LFS = 11, in our example). The minimal and maximal duration of a plan's SOPO are written directly into the diagram, using "minDu" and "maxDu" as abbreviations. They normally appear to the right of the SOPO; in case there is not enough room to the right, they appear to the left of the SOPO. In our example, the minimal duration is 2, the maximal duration is 6. As introduced in section 3.3, the minimal and maximal duration can be interpreted as the distance between the diagonal edges of the SOPO and the diagonal axis of the diagram.

5.2.2.2 Temporal Order of Plans

Of particular interest is the temporal relation between plans. In Asbru, plan types indicate whether plans are to be performed in sequence, in parallel, or in any order (see section 2.2.4). According to Allen's relations (see Figure 3.6), these plan types are indicated by the way the plans' sets of possible occurrences (SOPOs) are arranged within the diagram.

Sequential Plans

In Asbru, sequential plans are defined as plans whose sub-plans are performed in total sequence. For a plan consisting of two sub-plans, the second one can start only when the previous one has finished. Again, let us take Figure 5.5 for an example. The marked Plan E is the first sub-plan of Plan C which has the type *Do-All-Sequentially*. Therefore, Plan F's earliest starting shift corresponds to Plan E's earliest finishing shift (8 seconds in the

example). If 8 seconds is the earliest point in time for Plan E to be finished, then it is also the first point in time for Plan F to start. Accordingly, Plan F's latest starting shift corresponds to Plan E's latest finishing shift (that is 11 seconds in the example).

Plans to be performed in sequence are therefore arranged like stairs along the diagonal axis. Still, it is possible that the plans' SOPOs overlap in the diagram. Figure 5.10 shows an example of a sequential plan that consists of three sub-plans. The first and the second SOPO overlap, due to the fact that both their finishing intervals overlap with their starting intervals. The second and the third SOPO do not overlap because the third plan's finishing interval does not overlap with its starting interval.

Any-Order Plans

These plans consist of several sub-plans, whose order of execution is not known at the time of the plan's design. In the left part of SOPOView, this plan type is indicated by two arrows pointing into opposite directions. In the diagram, the sub-plans' SOPOs are drawn in a sequence, like those of a sequential plan (see previous description).

Of course, looking at the diagram, one cannot tell whether a plan's sub-plans are to be performed in total sequence or in any order. Merely, assuming different starting and finishing intervals for the sub-plans does not communicate the nature of the any-order plan type. For the sub-plans' time annotations, it would be better to assume the same starting and finishing interval for all of them. The interpretation would be that all the sub-plans had to start and finish within certain temporal borders, but that their order of execution is unknown. However, this would lead to overlapping SOPOs, as is the case with parallel plans, and therefore would not be a unique representation either.

Parallel Plans

Sub-plans that are to be performed in parallel have the same starting interval, but may have different durations and finishing intervals. An example of a parallel plan is given in Figure 5.5. There, Plan G and H are to be executed in parallel. In the diagram, their SOPOs lie vertically in the same interval [ESS, LSS].

In our example, the plans' finishing intervals and therefore the SOPOs overlap. This overlapping area is drawn semi-transparent, so that one can still see the overlapped SOPO's edges. During evaluation, this proved to be an insufficient solution, especially when more than just two SOPOs overlap, or when the SOPOs are identical. A possible

solution to these deficiencies could be the introduction of a third dimension into the diagram. There, parallel SOPOs can be stacked on top of each other. However, an additional dimension would also increase the overall complexity of the diagram.

5.3 User Interaction

This section describes the basic user interaction concepts used in SOPOView.

5.3.1 Plan Selection

In both the left part and diagram, a plan, or SOPO, is selected by clicking on it. In the left part, a border consisting of moving dots ("ants trail") is drawn around the plan's colored rectangle. In the diagram, the plan's SOPO is marked (see section 5.2.2.1). In case there are several SOPOs overlapping, one can click into the overlapping area to switch between the plans. By clicking into a background triangle, one can select the corresponding opened superplan (see also section 5.2.1.3).

5.3.2 Level of Detail

This term refers to the visible levels within a plan hierarchy (see also section 5.2.1.3). Following the principle of *details-on-demand* in information visualization [Shneiderman, 1998], the user can choose the level of detail by opening and closing plans, i.e. by choosing whether the plan's sub-plans will be visible or not. To open or close a plan, this plan has first to be selected.

In the left part of SOPOView, the plan can then be opened or closed by clicking into the black triangle to the left of the plan's name. When the mouse cursor is moved over the triangle, it changes into a hand, which symbolizes that the triangle can be clicked on. In the diagram, a small dark circle is drawn in the lower right corner of the SOPO (see Figure 5.6 or 5.7, for example). Its function is the same as that of the black triangle in the left part. Accordingly, the mouse cursor changes into a hand when moved over this circle.

5.3.3 Changing Time Annotations

There are two ways to change a plan's time annotation: by directly manipulating the SOPO, or by using an editor. In order to enable a proper visualization of SOPOs, the restrictions introduced in section 2.2.3 were considered.

5.3.3.1 Direct Manipulation of SOPOs

Provision of direct manipulation offers many advantages: users can carry out tasks rapidly and can observe the results immediately; it allows easy learning and errors to be avoided; and it affords a higher subjective satisfaction. A detailed discussion of direct manipulation is given by Shneiderman [1998].

In our diagram, one can click on a SOPO and thus select it. Without releasing the mouse button, the SOPO can be dragged, moved around the diagram, and dropped at some other place ("drag and drop"). When moved outside the visible area, the diagram scrolls automatically to support the dragging. The SOPO moves by one time unit. However, there are certain restrictions imposed by the language Asbru to be considered:
First, only SOPOs representing plans without any sub-plans can be moved (such plans are called actions). If a plan has sub-plans, its time annotation is the propagation of the sub-plans' time annotations and can only be changed by changes in the sub-plans' time annotations.
Second, the SOPO (of an action) can only be moved horizontally. This is due to the fact that a plan's starting interval must correspond to the finishing interval of the previous plan (in a sequential plan). In addition, the very first plan in a plan hierarchy has to start with the earliest starting shift equal to zero.
Third, the leftmost possible position of a SOPO is the diagonal axis. Having a corner or an edge on the diagonal axis means that these points of the SOPO have the same starting and finishing point in time, i.e. a duration of zero. All parts of a SOPO left to the diagonal would have a finishing time occurring before the start, which is impossible.

The other way of directly manipulating a SOPO is by changing the position of its edges. If the user moves the mouse cursor over one edge of the SOPO, the cursor changes into a double-sided arrow pointing into the directions in which the edge can be dragged. For example, on the edges representing the finishing interval, this arrow would point to the left and right. After the cursor has changed into an arrow, the user can click and move the edge to the permitted direction. The value of the time annotation's parameter then

changes by one time unit. Thus, the arrows should only appear when it is permitted to change the corresponding time shift. For plans which consist of sub-plans, the arrows do not appear, as their time annotations cannot be changed directly. The same is true of the dashed lines which represent the undefined parts of a time annotation.

With direct manipulation, the user can only change the edges of the SOPO by one time unit (e.g., one second or one hour, depending on the current time scale). To change, for example, the minimal duration from 1 hour to 1.5 hours, one has to use the editor. Also, to specify a parameter as undefined, the editor has to be used (see below).

5.3.3.2 Time Annotation Editor

The other possibility to change a plan's time annotation is by using an editor. To open the editor, the user simply has to click on a SOPO with the right mouse button. For an example of the editor, see Figure 5.11.

In the editor, all the parameters, ESS, LSS, EFS, LFS, MinDu and MaxDu, can be assigned any value that meets the *constraints within time annotations*. Furthermore, checkboxes enable the user to specify parameters as undefined.

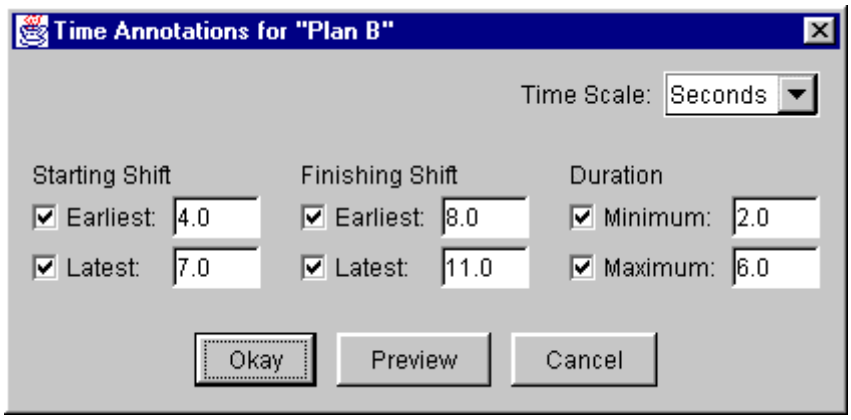


Figure 5.11: Time Annotation Editor.

5.3.4 Scrolling

To enable navigation in the diagram, a horizontal and a vertical scrollbar were implemented. The way the user scrolls in the diagram is indicated by the small square button

between the two scrollbars in the bottom right. By default, the user scrolls diagonally, which is indicated by a double-headed diagonal arrow on that button (see Figure 5.1, for example). The two scrollbars are then synchronized. By clicking on the button, the user can change the way of navigation. Doing this, two double-headed arrows appear on the button: a horizontal and a vertical one. The user can then scroll horizontally using the horizontal scrollbar, or vertically using the vertical scrollbar.

5.3.5 Control Bar

The following possibilities of changes concerning the appearance are made available through a control bar at the bottom of SOPOView. See Figure 5.1, for example.

5.3.5.1 Choosing the Time Scale

The user can select any time scale from seconds up to years using the pull-down list. The selection is used for the representation in the diagram, and is indicated on both the beginning and ending axis by an abbreviation in brackets (by default, it is seconds, indicated as [sec]).

5.3.5.2 Changing a Plan's Color

The colors that are used throughout AsbruView are generated randomly. Sometimes, colors might be misleading or contrasts might be bad. At any time, the user can change a plan's color, which of course affects also the other views in AsbruView.

5.3.5.3 Showing a Grid in the Diagram

By activating this checkbox, a grid is drawn in the diagram. The distance between the lines is always five time units. This should enable a better orientation in the diagram.

5.3.5.4 Zoom in and out

The two buttons that show a magnifier with a plus and a minus in the glass can be used to zoom in and out the diagram.

5.4 Integration Into AsbruView

In section 2.4.1, the basic structure of the user interface AsbruView, that supports the understanding and manipulation of time-oriented, skeletal plans written in Asbru, was introduced. In AsbruView, any number of views can be used to visualize certain aspects of the underlying model. SOPOView was implemented as such a view. In fact, SOPOView consists of two views: the left part showing the structure of the plans, and the SOPO diagram. This is due to the fact that the program code for the left part was reused from the Temporal View and integration into SOPOView was thus easier. For practical working, there are no disadvantages, of course.

5.4.1 Technical Details

The whole program was written in the programming language Java (release 1.2, also known as the Java Platform 2). In Java, the source code is first compiled into a platform-independent byte-code. To run the program, this byte-code will then be interpreted by the Java Virtual Machine on any platform. This way, programs developed on a certain platform can be moved and run (theoretically) on all the other platforms.
The other main reason for us to use Java are the features the language provides for user interface design. The program makes heavy use of the possibilities of the Swing classes of Java, which not only provide all the necessary components used in a modern user interface, but also good event handling methods and listeners used to react to changes done by the user.
However, since a lot of computing power and memory is necessary to draw the three-dimensional plan structure of the Topological View and then move there, or to react properly and smoothly to the user interactions on the SOPO diagram, work can be very difficult and demanding if such a computing system is not available. The system used for our evaluation (see next chapter) was fast enough for our purposes.

Chapter 6

Evaluation

This chapter reports the results of the evaluation we did to assess the usability and suitability of our visualization presented in this thesis.

As introduced in Chapter 4, evaluation plays the central role within a software engineering process. Without evaluating a system or user interface, the designer would never know whether users feel comfortable in using the software or whether the software makes work easier and more efficient at all. Especially in the medical domain, thought patterns of physicians can be so very different from our own (as computer scientists), that design decisions we thought good may turn out to be inappropriate.

6.1 Goals

Essentially, this evaluation was done to answer the following questions:

- Do physicians understand the concept of SOPOs used for the visualization of temporal aspects of plans?
- Is the depiction of plans' time annotations with SOPOs easy to understand, or do physicians have problems in reading the diagram?
- Does the SOPO diagram make understanding the temporal relations between plans easier?
- How do physicians evaluate SOPOView in comparison to Temporal View?

6.2 Sample

The participating persons (four male, four female) are all practicing physicians in different fields: neonatal intensive care, pediatrics, intensive care, and neurological surgery. Half of them (two male, two female) already helped us to evaluate the user interface AsbruView approximately a year ago. They not only could remember the program and

important features, but also had some basic knowledge about the plan representation language Asbru, e.g., hierarchical decomposition of plans and plan types.
All the participants use a computer for their daily work. They are all familiar with word processing software, and all except one use spread sheets and Internet browsers. Only one person had experience with a project planning software. Overall, participants rated their computer skills to be good; only two rated their skills to be low.
All but one use clinical protocols or guidelines in some way. They are written down in plain text or drawn as flowcharts. On an average, these representations were rated to be okay, but improvements were said to be very welcome.

6.3 Evaluation Procedure

Evaluation was done separately with every participant. After being told what to expect, the participant was asked to fill out a first questionnaire (see Appendix A) about basic computer skills and the usage of clinical protocols. Then, the basics of the user interface were explained: how to author plans using the Topological View and how to define the plans' time annotations in SOPOView and in Temporal View. To change from a rather general explanation to a practical example, a predefined plan was presented. After this, the test person was asked to author an example plan on his or her own, including the definition of time annotations in SOPOView, in comparison to the Temporal View. At the end, the participant was asked to fill out another questionnaire (see Appendix A), to assess the program's usability. The tests usually lasted about an hour and a half.

6.4 Test Equipment and Environment

The same system was used in all the tests: An Intel Celeron 433 MHZ portable PC, with 64 MB RAM and a 1024x768 LCD panel. A separate mouse was also used. All but one of the tests were conducted in the physicians' usual work environment in the hospital. One test was conducted at the participant's home.

6.5 SOPOView: Findings

The numerical results of our evaluation are presented in Appendix B. Here, these results will only be summarized and discussed together with the findings of the observations and interviews done during evaluation.

All participants understood the concept of SOPOs used to visualize temporal aspects and relations of plans. However, that did not mean that participants did not have any problems in reading the diagram and thinking in an unconvential way (what the representation in a two-dimensional diagram is, in the beginning). In fact, a plan's time annotation consists of three parts: the beginning shown on the vertical axis, the end shown on the horizontal axis, and the duration shown on the diagonal axis. Almost all participants had problems in leaving behind their traditional way of thinking, in which time only flows along one axis. Even when SOPOs were marked and their temporal information was highlighted along the axes, some participants had problems in understanding those values.

Two participants found the position of the axes misleading. In fact, they proposed the original orientation presented in section 3.3, where the horizontal axis marks the beginning, and the vertical axis marks the end in time. Other test persons changed the axes by mistake when asked to interpret a plan's time annotation.

Most participants had trouble understanding the temporal relations between plans, i.e. in reading the diagram along the diagonal axis. Taking the time annotations of sequential plans, one can mirror the first plan's finishing interval over the diagonal axis, to see whether the following plan's starting interval is the same. Problems occurred with any-order plans, as they are depicted like sequential plans and one cannot see any difference in the diagram. It would be useful, for example, to draw arrows between the SOPOs to visualize the nature of any-order plans in some way.

Some participants had problems understanding the representation of a plan's duration in the diagram. Unlike the traditional way of representing a plan's duration by a line or bar, which becomes broader the longer the plan lasts for, the duration is indicated by the distance between the SOPO and the diagonal axis. Therefore, there is no relation between the size of a SOPO and its duration. To help users, the minimal and maximal duration could be indicated, for example, by arrows pointing from the diagonal axis to the SOPO.

The colored background triangles used to communicate the hierarchical decomposition of plans were heavily criticized. The colors of the triangles are not the same as the ones used for the plans in the other views, and thus recognition of plans was rather difficult. The use of color in the whole was considered not only to be helpful, but also to be necessary to differentiate SOPOs in the diagram. However, some participants complained about contrasts between colors, in particular in connection with background colors.

One of the bigger problems turned out to be the representation of parallel plans by overlapping SOPOs. All participants found it difficult to recognize how many SOPOs are to be performed in parallel by just looking at the diagram. Especially when such parallel plans consist of sub-plans, the representation does not communicate that in an appropriate manner. Suggestions for improvement ranged from showing the number of overlapping SOPOs and using other hints to a three-dimensional diagram, where parallel SOPOs are stacked on each other. In fact, during design phase of SOPOView, a third dimension was taken into consideration, but finally was rejected because of its complexity.

The representation of undefined parts of a time annotation by dashed edges of a SOPO was rated to be satisfactory. Still, recognition strongly depends on the colors used. Some participants had to take a second look to recognize the dashed lines, and thus a better way has to be found. As described in Chapter 5, the current visualization of undefined parameters in SOPOs is pretty controversial: we assume a value for the edge of the SOPO to be drawn, although this value is undefined and therefore can be somewhere else.

Although it is possible to author time annotations in different time units, this proved to be inconvenient for representation in our diagram. SOPOs defined in a smaller time unit (e.g., seconds) will not be visible in a large time scale (e.g., days), and in the opposite case, the larger SOPOs will fill the whole visible area in a smaller chosen time scale. During evaluation, this happened only once, but interpreting the SOPO diagram then turned out to be almost impossible.

Another deficiency in visualization is the fact that SOPOs represent a set of possible occurrences, caused by intervals of the beginning, end, and duration, rather than a single occurrence. Some participants wanted to define a plan's time annotation as a time point or an exact occurrence with a certain duration. This would not lead to a colored region (that is, the SOPO), but to a single point or line. However, the current implementation supports only the definition of such a time annotation, not its visualization. The point or line was invisible then, and could not be marked any more.

Both possibilities for changing the SOPOs' parameters, either by direct manipulation or by using the editor, were rated good. However, the overall procedure to define a plan's time annotation was not rated so highly, as it was rather time-consuming and restricted by the language Asbru. Several times, participants wanted to have temporal breaks between the sequential execution of plans. In Asbru, this is not possible, so one would have

to define plans that represent the breaks in between. Also, a means of copying a plan's time annotation and reusing it for another plan would be helpful.

Another problem that arose was the fact that the user input in the editor is not checked if it meets the constraints within time annotations (see Chapter 2). For example, a user could define a duration interval, where it was impossible to reach the finishing interval. A solution would be to adapt the values of the finishing interval when the minimal and maximal duration are set, or vice versa.

One participant criticized the unused part of the diagram below the diagonal axis and suggested, as did another user, that the plan's name and other data such as the time annotation's parameters be shown there. Since the user may have to scroll to bring a SOPO into the visible area, this part is not shown all the time (unless implemented so that it *will* be shown all the time, or, of course, as an optional feature).

6.6 Comparison: SOPOView vs. Temporal View

Both SOPOView and Temporal View provide representations for Asbru's time annotations. As the final part of the evaluation, we wanted the participants to compare the SOPO diagram and the Temporal View according to certain criteria. The numeric results can be found in Appendix B, but are presented and discussed here.

Half of the participants rated the Temporal View to be better in terms of representing the structure of plans (hierarchical decomposition). Two found the SOPO diagram to be better, the other two said that representations were equal. Still, most participants found the diagram rather unusable without the left part, where structure and plan types are indicated the same way as in the Temporal View. Therefore, indicating the hierarchical decomposition of plans by colored background triangles in the diagram is not a sufficient means of visualization.

All except one found that understanding the temporal relations between plans is easier in the Temporal View; the last participant felt unable to give an answer. This result is probably due to the fact that thinking within a two-dimensional diagram is rather unconventional, and takes more time to get familiar with than the physicians had during evaluation.

Visualization of undefined parts of a time annotation was rated to be equal in both views. Two participants found the SOPO diagram to be better and two the Temporal View, while three rated them to be equal (although equally bad), and one could not give an answer.

All but one found that the Temporal View enables faster and better reading of the temporal information in general. Also, half of the participants rated the Temporal View to be a more clearly arranged representation than the SOPO diagram. Two said the opposite, and the remaining two could not give an answer.

Half of the physicians found the SOPO diagram to be better, regarding the faster and better reading of the temporal information of a single (marked) plan. The other half found the Temporal View to be better.

With the last question, we wanted to know from the participants which view they found to be better suited for representing temporal aspects of plans. Half of them preferred the Temporal View, one said they are equal, and three could not give an answer.

6.7 Discussion

Considering the time our participants had to get familiar with the diagram, results were quite surprising and not completely negative, as might have been expected. However, most physicians found the diagram to be too complex and not clearly arranged enough. In their opinion, learning how to use it efficiently would take time which they do not have in their daily schedule.

Of course, physicians want a software that makes their work with protocols easier and helps them to understand complex (temporal) relations between actions and plans. Using SOPOView, representations are still too complex and demand an additional effort in thinking. Thus, most participants did not recognize the program's benefits and rated the overall practicability to be rather bad.

In fact, almost all participants did not have to make use of the features the SOPO diagram offers. When they had to author their example plans during evaluation and had to define the temporal aspects of their plans, they either could not tell anything about the required time, or they had a rather exact knowledge. Representation by SOPOs, where one can visualize an uncertainty in beginning, end and duration, offers much more than they actually needed.

One lesson learned from the evaluation was that for simple cases or tasks, a simple representation is needed, as much as for complex cases a more complex depiction is probably needed. Still, this one should be simple enough to be usable. To use Einstein's dictum: everything should be made as simple as possible, but no simpler.

Chapter 7

Conclusion

This last chapter summarizes the main issues presented throughout this thesis, reflects the results from the evaluation and gives an outlook for possible future developments.

7.1 Summary

The need to improve the quality of health care has led to a strong demand for clinical protocols and computer systems that support both their creation and execution. Traditional ways for the representation of such protocols include free text, flowcharts, or decision tables. However, most of them lack the means to fully capture both the complexity and the temporal dimension of treatment plans.
In the Asgaard Project, the time-oriented, machine-readable language Asbru was developed to address these requirements. Asbru's time annotations allow a representation of uncertainty in start, end, and duration of a time interval that represents a plan's temporal dimension. Within the user interface AsbruView, which supports the understanding and manipulation of skeletal plans written in Asbru, a special glyph was developed to visualize this uncertainty in time.

This thesis introduced an alternative representation of a plan's temporal aspects, which we called SOPOView. It uses a two-dimensional diagram with three time axes - a vertical axis for the starting time, a horizontal axis for the ending time, and a diagonal axis for the duration - in order to visualize the plans' time annotations. Within the diagram, a single point represents an occurrence that has an exact beginning, end, and duration. Uncertainty in these three dimensions leads to a representation of a plan's time annotation not as an occurrence, but as a set of possible occurrences (SOPO), depicted as a colored region.
We presented our adaptations and enhancements to the original concept [Rit, 1986], that were necessary in order to enable a proper depiction of an Asbru plan's properties, and we described in detail the interaction mechanisms that the user has to define and change a plan's time annotation, and to work with the diagram as a whole.

The final part of this work was the evaluation we conducted with eight physicians to assess SOPOView's usability. We presented the results and the problems that had occurred during this evaluation.

7.2 Thoughts on SOPOs

The representation of temporal information within a two-dimensional diagram is an unconventional one. Indeed, during evaluation, most participants had trouble with the orientation in the diagram. They were rather used to a one-dimensional representation, as is the case in LifeLines (e.g., [Plaisant et al., 1998]; see also section 3.1).
Much more, the depiction of plans' time annotations as colored regions within the diagram led to some irritation and confusion. These SOPOs do not have a unique shape and position, and can hardly be compared to glyphs, which essentially change only some of their dimensions (e.g., their width) and can thus be recognized more easily for representing the same kind of information. SOPOs also change their appearance when the underlying time annotation changes. However, these changes not only comprise the width or height of a SOPO, but also its distance from the diagonal axis and its shape. A SOPO being depicted as a rectangle before may be depicted as a triangle or even as a hexagon after the time annotation has changed. This is due to the fact that every single edge of the SOPO corresponds to one parameter of the time annotation.
Additionally, a SOPO's width or height does not tell anything about the plan's duration, which one might have concluded from a one-dimensional depiction where the width of a timeline, for example, expresses the duration. In the diagram, the duration is indicated by the distance between the SOPO and the diagonal axis, which is hard to understand and remember in the beginning.

One might ask why we chose to use the concept of SOPOs for representing Asbru's time annotations at all. SOPOs are a very powerful means of visualizing temporal uncertainty. A SOPO's shape is directly derived from the time annotation's parameters, and linking different SOPOs to each other enables the representation of temporal relations. On a rather small area on the screen, a lot of information can thus be presented.

7.3 Limitations

Learning how to efficiently use the diagram is not an affair of a few minutes. One has to get used to this different (and still complex) kind of representation over time. During

evaluation, time was rather limited; therefore it was rather hard for the participants to recognize the benefits that the diagram offers.

We did not develop SOPOView to be a cure-all to the various problems in visualizing temporal uncertainty. In fact, some of Asbru's properties that need to be communicated by the representation were not covered optimally, and should be summarized here:
First, the hierarchical decomposition of plans, indicated by colored background triangles, turned out not to be of much help. The colors do not look the same as the ones for the plans, and thus recognition suffered (in fact, we used the same color, but semitransparent). Having more than a few hierarchical levels, the diagram soon becomes pretty colorful and does not enable faster reading and understanding. Especially with parallel plans, recognition becomes nearly impossible at times.
Second, the current overlapping of SOPOs, e.g., that of parallel plans, is unusable. Having more than just a few plans, which may in addition consist of sub-plans as well, one cannot recognize their SOPOs and colors.
Third, the original concept of SOPOs was not meant to represent undefined parts, which is possible in Asbru's time annotations. The current solution, and all the other propositions presented, is far from perfect, as we assume a value in order to be able to draw the SOPO. However, the resulting depiction supposes a certain width of the interval, which does not really exist.
Fourth, Asbru allows the definition of parts of a time annotation in different time units. In our diagram, this proved to be inconvenient. For practical reasons, time annotation's parameters should be defined in the same time unit.
Fifth, in the current implementation, single occurrences, i.e plans that have an exact beginning, end, and duration, cannot be represented (in fact, they are invisible).

Especially this last point addresses the diagram's complexity, even when simple cases are to be depicted. As mentioned at the end of the previous section, using the diagram enables complex temporal aspects to be visualized (although still in a complex manner). However, there is no reason to use a complex depiction for simple cases, such as single occurrences. There, a timeline would be much easier and faster to read. In the following last section, we will come back to this issue.

7.4 Future Work

The development and implementation of SOPOView was part of the evolution of the user interface AsbruView, and thus cannot stand alone. In fact, some of the deficiencies and

limitations presented above result from the principle concepts used for representing Asbru plans. For example, during evaluation, some users complained about the necessity of defining intermediate plans that only provide a plan type for its sub-plans. In Asbru, it is not possible to define a plan that consists of a first plan, followed by two plans in parallel. Instead, the overall plan has to consist of two plans performed in sequence, where the second plan itself consists of the two plans in parallel. Obviously, this intermediate plan only helps to structure the overall plan, but has no other function. A future version of AsbruView might use this hierarchical decomposition only internally. The user interface could then allow more flexibility and could meet the users' needs in a more appropriate way.

Regarding the further development of SOPOView, it is not quite clear how far the visualization will be enhanced. We presented some challenges, e.g., a better representation of undefined parts, or the depiction of single occurrences. Especially the issue of overlapping SOPOs needs to be addressed in a future release. We already suggested the introduction of a third dimension into the diagram for this purpose.
One clear result of the evaluation is that a single view addressing the temporal dimension will definitely not solve all of the problems. Users' tasks and requirements are too different for one single representation to meet them all. SOPOs are a powerful means to visualize complex time annotations. However, when it comes to single occurrences, a timeline would do the same and is probably much easier and faster to read.
A future version of AsbruView has to provide different means for the representation of temporal aspects, depending on users' definitions and needs. SOPOs could be one of them.

Appendix A

A.1 Questionnaires

On the next few pages, the questionnaires which we used during evaluation are shown: One to be filled out before the test (Figure A.1), and one after the test (Figure A.2). The reason we used two different questionnaires rather than a big one was that we wanted to get to know the computer skills and familiarity with computer software of our participants before the test (Questionnaire I). This way, we could decide how much we had to explain during introduction of our software.

Questionnaire II, which was used for evaluating our prototype's usability, consists of three parts. In Part I (SOPO-Diagram) and Part II (SOPOView as a whole), test-users had to rate different aspects of the user interface on a scale between paired adjectives, referred to as a *semantic differential*. Additionally, Part II also provides some open questions. In Part III, test-users were asked to compare the SOPO-Diagram with the Temporal View due to some given criteria. Ideas for the questionnaires were taken from Ravden and Johnson [1989], Shneiderman [1998] and Kosara [1999].

Figures A.1 and A.2 show the original German questionnaires used for our evaluation; the English translation is attached afterwards (Figures A.3 and A.4). They were not used during evaluation.

Fragebogen I. Vor dem Test.

Bitte Zutreffendes ankreuzen!

(1) Wie of verwenden Sie einen Computer?

Beruflich:	☐ Täglich	**Privat:**	☐ Täglich
	☐ Mehrmals wöchentlich		☐ Mehrmals wöchentlich
	☐ Gelegentlich		☐ Gelegentlich
	☐ Nie		☐ Nie

(2) Wie würden Sie Ihre Computerkenntnisse einstufen?

☐ Sehr gut ☐ Gering
☐ Gut ☐ Sehr gering
☐ Mittel

(3) Mit welchen der folgenden Eingabegeräte sind Sie vertraut? (Mehrfachnennungen möglich)

☐ Tastatur ☐ Touchscreen
☐ Maus ☐ Sonstige, nämlich:
☐ Trackball

(4) Mit welchen der folgenden Softwaresysteme sind Sie vertraut? (Mehrfachnennungen möglich)

☐ Textverarbeitung (zB MS Word)
☐ Tabellenkalkulation (zB MS Excel)
☐ Projektplaner (zB MS Project)
☐ Zeichenprogramme (zB MS PowerPoint, Corel Draw)
☐ Windows Explorer
☐ Internet Browser (zB Internet Explorer, Netscape Navigator)
☐ Andere, nämlich:

(5) Spielt der Faktor Zeit bei Ihrer Arbeit bzw bei Behandlungen in Ihrem Bereich eine wesentliche Rolle (zB zeitkritische Aktivitäten)?

☐ Ja
☐ Nein

(6) Verwenden Sie in irgend einer Form klinische Protokolle/Leitlinien bei Ihrer Arbeit?

☐ Ja:

Wie werden diese Protokolle/Leitlinien bisher dargestellt:

Wie zufrieden sind Sie mit den bisherigen Darstellungen?

☐ Sehr zufrieden ☐ Wenig zufrieden
☐ Zufrieden ☐ Gar nicht zufrieden
☐ Mittel

☐ Nein:

Erwarten Sie Vorteile oder Verbesserungen durch die Verwendung von Protokollen/Leitlinien?

☐ Ja
☐ Nein

Vielen Dank!

Figure A.1: Questionnaire I. Before the test. (German original).

Fragebogen II. Nach dem Test.

Teil I: SOPO-Diagramm

Bitte kreisen Sie die Nummern ein, wobei jeweils "1" dem Begriff links davon zugeordnet ist und "5" dem Begriff rechts. Keine Angabe = kA.

1. Wie verständlich ist für Sie die Darstellung der zeitlichen Informationen von Plänen oder Protokollen mittels SOPOs?

Sehr verständlich **1 2 3 4 5** unverständlich **kA**

---

2. Für wie übersichtlich halten Sie die Darstellung der zeitlichen Informationen von Plänen oder Protokollen mittels SOPOs?

Sehr übersichtlich **1 2 3 4 5** unübersichtlich **kA**

---

3. Für wie geeignet halten Sie die Darstellung der zeitlichen Informationen von Plänen oder Protokollen mittels SOPOs?

Sehr geeignet **1 2 3 4 5** nicht geeignet **kA**

---

4. Für wie hilfreich halten Sie die Verwendung von Farben zur Unterscheidung der Pläne?

Sehr hilfreich **1 2 3 4 5** nicht hilfreich **kA**

---

5. Wie gut wird Ihrer Meinung nach die Darstellung der Struktur (hierarchische Zusammensetzung) von Plänen oder Protokollen durch Hintergrundfarben verdeutlicht?

Sehr gut verdeutlicht **1 2 3 4 5** sehr schlecht verdeutlicht **kA**

---

6. Für wie gut halten Sie die Farbüberlagerungen bei überlappenden SOPOs (vor allem bei parallelen Plänen)?

Sehr gut **1 2 3 4 5** sehr schlecht **kA**

---

7. Wie gut werden Ihrer Meinung nach die zeitlichen Zusammenhänge/Abhängigkeiten zwischen Plänen verdeutlicht? (zB ob Pläne sequentiell oder parallel ablaufen)

Sehr gut verdeutlicht **1 2 3 4 5** sehr schlecht verdeutlicht **kA**

---

8. Für wie aussagekräftig halten Sie die Darstellung optionaler Pläne?

Sehr aussagekräftig **1 2 3 4 5** nicht aussagekräftig **kA**

Figure A.2: Questionnaire II. After the test. (German original).

9. Für wie aussagekräftig halten Sie die Darstellung von nicht definierten zeitlichen Informationen? (strichlierte Ränder der SOPOs)

Sehr aussagekräftig **1 2 3 4 5** nicht aussagekräftig **kA**

---

10. Für wie gut halten Sie die Darstellung der zeitlichen Informationen bei markierten Plänen (Anzeige der Informationen an den Achsen bzw im Diagramm)?

Sehr gut **1 2 3 4 5** sehr schlecht **kA**

---

11. Für wie gut halten Sie die direkten Manipulationsmöglichkeiten der SOPOs? (Verschieben von SOPOs, Ränder ziehen)

Sehr gut **1 2 3 4 5** sehr schlecht **kA**

---

12. Für wie gut halten Sie den Editor zum Verändern der zeitlichen Informationen von Plänen?

Sehr gut **1 2 3 4 5** sehr schlecht **kA**

---

13. Für wie einfach halten Sie insgesamt das Verändern der zeitlichen Informationen von Plänen oder Protokollen?

Sehr einfach **1 2 3 4 5** sehr schwierig **kA**

---

14. Wie beurteilen Sie die Navigationsmöglichkeiten (Scrolling) im SOPO-Diagramm?

Sehr praktisch **1 2 3 4 5** unpraktisch **kA**

---

15. Wie beurteilen Sie die Möglichkeit, im Diagramm zu zoomen?

Sehr praktisch **1 2 3 4 5** unpraktisch **kA**

Teil II: SOPOView gesamt

16. Erhöht die Baumansicht (Bereich links vom SOPO-Diagramm) die Verständlichkeit der Struktur bzw hierarchischen Zusammensetzung von Plänen oder Protokollen?

Auf jeden Fall **1 2 3 4 5** keinesfalls **kA**

---

17. Erhöht die Baumansicht die Verständlichkeit der zeitlichen Zusammenhänge/Abhängigkeiten von Plänen oder Protokollen (zB ob Pläne sequentiell oder parallel ablaufen)?

Auf jeden Fall **1 2 3 4 5** keinesfalls **kA**

Figure A.2: (Continued).

18. Könnten Sie sich vorstellen, Ihre Protokolle (und da vor allem die zeitlichen Informationen) mit SOPOView darzustellen?

Auf jeden Fall **1 2 3 4 5** keinesfalls **kA**

---

19. Wie würde der Einsatz von SOPOView Ihre Arbeit mit Protokollen beeinflussen?

erleichtern **1 2 3 4 5** erschweren **kA**

Was hat Ihnen an SOPOView besonders gut gefallen? Warum?

Was hat Ihnen an SOPOView gar nicht gefallen? Warum?

Gibt es irgend etwas in der Darstellung oder in der Funktionalität, das Sie für missverständlich oder irreführend betrachten?

Welche Veränderungen würden Sie durchführen, um SOPOView aus Benutzersicht aus zu verbessern?

Fällt Ihnen sonst noch etwas (positives wie negatives) zu SOPOView ein?

Figure A.2: (Continued).

Teil III: Vergleich SOPODiagramm - TemporalView

Geben Sie bei den folgenden Aussagen jeweils an, für welche der beiden Darstellungen der zeitlichen Aspekte (SOPODiagramm oder TemporalView) diese eher zutreffen bzw ob beide Ansichten gleich gut sind. Keine Angabe = kA.

	SOPO-Diagramm	Temporal-View	beide gleich gut	kA
Bessere Darstellung der Struktur (hierarchische Zusammensetzung) von Plänen	☐	☐	☐	☐
Bessere Darstellung der zeitlichen Zusammenhänge/Abhängigkeiten zwischen Plänen (zB ob Pläne sequentiell oder parallel ablaufen)	☐	☐	☐	☐
Bessere Darstellung nicht definierter zeitlicher Informationen von Plänen	☐	☐	☐	☐
Überblicksmässig besseres/schnelleres Ablesen der zeitlichen Informationen von Plänen	☐	☐	☐	☐
Besseres/schnelleres Ablesen der zeitlichen Informationen einzelner Pläne	☐	☐	☐	☐
Insgesamt übersichtlichere Darstellung der zeitlichen Informationen von Plänen	☐	☐	☐	☐
Welche Ansicht halten Sie insgesamt für die Darstellung der zeitlichen Informationen von Plänen für geeigneter?	☐	☐	☐	☐

Vielen Dank!

Figure A.2: (Continued).

Questionnaire I. Before the test.

Please check the appropriate answers!

(7) How often do you use a computer?

For work:	☐ Daily	**At home:**	☐ Daily
	☐ Several times a week		☐ Several times a week
	☐ Every now and then		☐ Every now and then
	☐ Never		☐ Never

(8) How would you rate your computer skills?

☐ Very good ☐ Low
☐ Good ☐ Very low
☐ Satisfactory

(9) Which of the following input devices are you familiar with? (Multiple answers possible)

☐ Keyboard ☐ Touchscreen
☐ Mouse ☐ Others:
☐ Trackball

(10) Which of the following software systems are you familiar with? (Multiple answers possible)

☐ Word Processor (e.g., MS Word)
☐ Spread Sheet (e.g., MS Excel)
☐ Project Planner (e.g., MS Project)
☐ Drawing Program (e.g., MS PowerPoint, Corel Draw)
☐ Windows Explorer
☐ Internet Browser (e.g., Internet Explorer, Netscape Navigator)
☐ Others:

(11) Does time play a major role at your work or at treatments in your domain (e.g., time-critical activities)?

☐ Yes
☐ No

(12) Do you use clinical protocols or guidelines for your work?

☐ Yes:

How are these protocols and guidelines represented:

How satisfied are you with these representations?

☐ Very satisfied ☐ Little satisfied
☐ Satisfied ☐ Not satisfied
☐ Medium

☐ No:

Do you expect advantages or improvements from using protocols or guidelines?

☐ Yes
☐ No

Thank you!

Figure A.3: Questionnaire I. Before the test. (English translation).

Questionnaire II. After the test.

Part I: SOPO-Diagram

Please circle the numbers, whereas "1" refers to the word on the left and "5" to the word on the right. Not Applicable = NA.

1. How intelligible is the representation of temporal information of plans or protocols with SOPOs?

Very intelligible **1 2 3 4 5** unintelligible **NA**

---

2. How clearly arranged is the representation of temporal information of plans or protocols with SOPOs?

Very clearly arranged **1 2 3 4 5** Badly arranged **NA**

---

3. How useful is the representation of temporal information of plans or protocols with SOPOs?

Very useful **1 2 3 4 5** Not useful **NA**

---

4. How helpful is the use of colors for the distinction of plans?

Very helpful **1 2 3 4 5** Not helpful **NA**

---

5. How does the use of background colors make the structure (hierarchical decomposition) of plans or protocols clear?

Very good **1 2 3 4 5** Very bad **NA**

---

6. How good in your opinion is the solution of using transparent colors with overlapping SOPOs (in particular with parallel plans)?

Very good **1 2 3 4 5** Very bad **NA**

---

7. How well are temporal dependencies clarified (e.g., whether plans are sequential or parallel)?

Very good **1 2 3 4 5** Very bad **NA**

---

8. How good is the representation of optional plans?

Very good **1 2 3 4 5** Very bad **NA**

Figure A.4: Questionnaire II. After the test. (English translation).

9. How good is the representation of undefined temporal information? (dashed edges of SOPOs)

Very good **1 2 3 4 5** Very bad **NA**

---

10. How good is the representation of temporal information with marked plans (information shown along the axes and within the diagram)?

Very good **1 2 3 4 5** Very bad **NA**

---

11. How good are the possibilities of direct manipulation of SOPOs (moving of SOPOs, dragging of edges)?

Very good **1 2 3 4 5** Very bad **NA**

---

12. How useful is the editor in order to change the temporal informaion of plans?

Very useful **1 2 3 4 5** Not useful **NA**

---

13. How easy is it to change temporal information of plans in general?

Very easy **1 2 3 4 5** Very difficult **NA**

---

14. How would you rate the possibilities of navigation (scrolling) within the SOPO-Diagram?

Very practical **1 2 3 4 5** Not practical **NA**

---

15. How do you rate the possibility to zoom in the diagram?

Very practical **1 2 3 4 5** Not practical **NA**

Part II: SOPOView as a whole

16. Does the tree view (part of the screen left to the SOPO-Diagram) make the structure (hierarchical decomposition) of plans or protocols easier to understand?

Absolutely **1 2 3 4 5** Not at all **NA**

---

17. Does the tree view make the temporal dependencies of plans or protocols easier to understand (e.g., whether plans are sequential or parallel)?

Absolutely **1 2 3 4 5** Not at all **NA**

Figure A.4: (Continued).

18. Can you imagine to represent your protocols (and in particular the temporal information) with SOPOView?

Absolutely **1 2 3 4 5** Not at all **NA**

---

19. How would the use of SOPOView change your work with protocols?

It would make it easier **1 2 3 4 5** It would make it harder **NA**

What are the best aspects of SOPOView? Why?

What are the worst aspects of SOPOView? Why?

Are there any parts of the viewing which you found confusing or difficult to fully understand?

What changes would you make to SOPOView in order to make it better from the user's point of view?

Is there anything else about SOPOView (positive or negative) you would like to add?

Figure A.4: (Continued).

Part III: Comparison SOPODiagram - TemporalView

Please check where the given statements are more appropriate (SOPODiagram or TemporalView) or whether both views are equal. Not Applicable = NA.

	SOPO-Diagramm	Temporal-View	Equal	NA
Better representation of the structure (hierarchical decomposition) of plans	☐	☐	☐	☐
Better representation of temporal relations between plans (e.g., whether plans are sequential or parallel)	☐	☐	☐	☐
Better representation of undefined temporal information of plans	☐	☐	☐	☐
Better and faster reading of temporal information of plans (general view)	☐	☐	☐	☐
Better and faster reading of temporal information of a single plan	☐	☐	☐	☐
More clearly arranged representation of temporal information of plans	☐	☐	☐	☐
Which view do you rate better suited for the representation of temporal information of plans?	☐	☐	☐	☐

Thank you!

Figure A.4: (Continued).

Appendix B

B.1 Evaluation: Numerical Results

In the following, the results of Questionnaire II will be presented. Figure B.1 summarizes the results from the questions of Part I (SOPO diagram) and Part II (SOPOView; without open questions) of the questionnaire. Since our sample (eight practicing physicians) was not big enough to draw any statistical conclusions, we only present the average ratings.
The results of the comparison between the SOPO diagram and the Temporal View are given in Figure B.2.

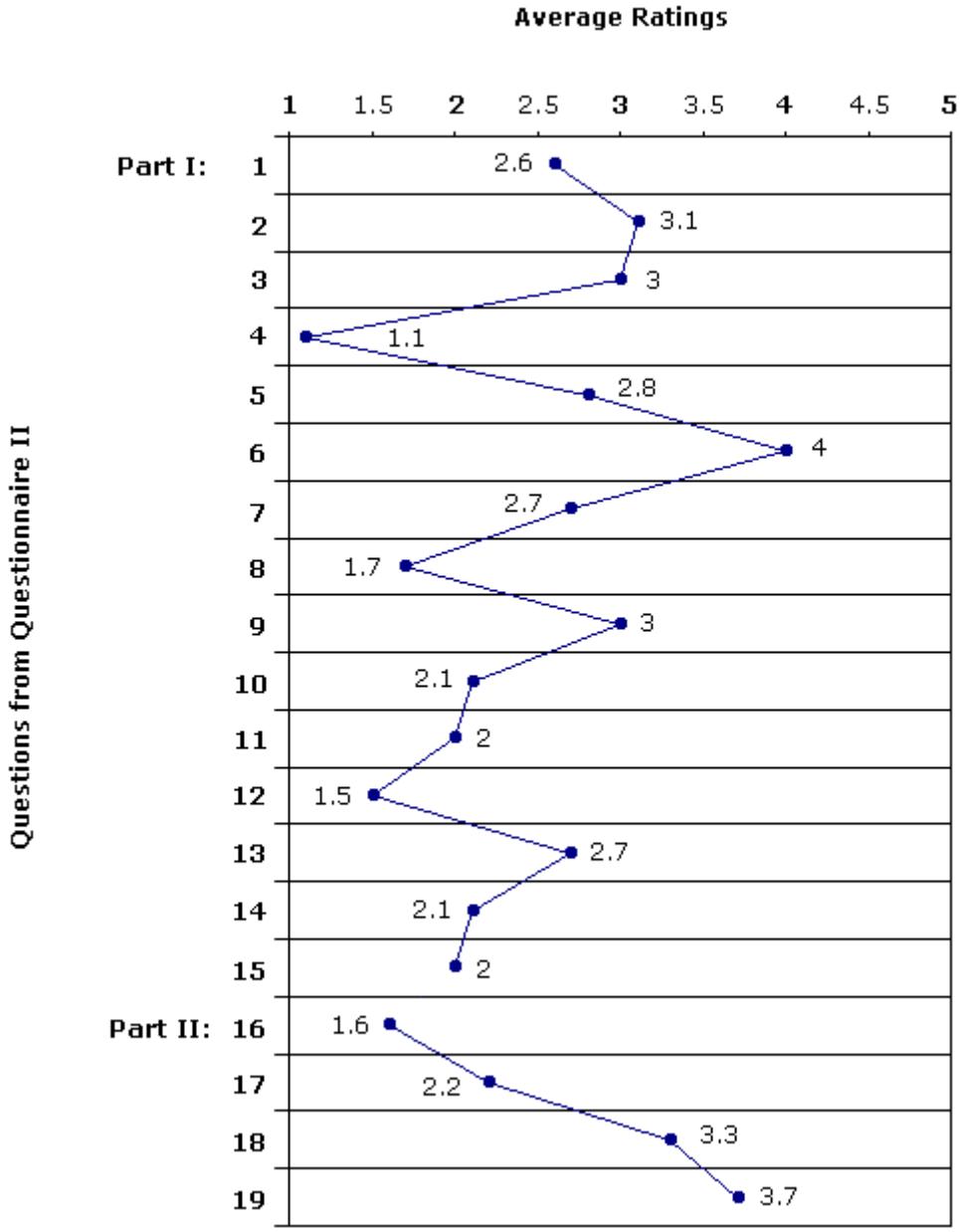


Figure B.1: Numerical results from Questionnaire II, Part I and II. The numbers along the vertical axis denote the questions (see Figures A.2 and A.4 respectively). The horizontal axis denotes the range for the ratings (1 = the best, 5 = the worst rating). The numbers in the diagram represent the average ratings.

Questions	SOPO-Diagram	Temporal View	Equal	NA
Better representation of the structure (hierarchical decomposition) of plans	2	4	2	-
Better representation of temporal relations between plans (e.g., whether plans are sequential or parallel)	-	7	-	1
Better representation of undefined temporal information of plans	2	2	3	1
Better and faster reading of temporal information of plans (general view)	-	7	-	1
Better and faster reading of temporal information of a single plan	4	4	-	-
More clearly arranged representation of temporal information of plans	2	4	-	2
Which view do you rate better suited for the representation of temporal information of plans?	-	4	1	3

Figure B.2: Results of the comparison SOPOView vs. Temporal View, showing the number of participants who selected either the one or the other view as better, or both to be equal, regarding the given questions. NA = not applicable. (see also Part III of the Questionnaire II, Figures A.2 or A.4).

Bibliography

[Allen, 1983] J. F. Allen. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM,* 26(11), pages 832-843, 1983.

[Barnes and Barnett, 1995] M. Barnes and G. Barnett. An architecture for a distributed guideline server. In *Proceedings of the 19th Annual Symposium on Computer Applications in Medical Care (SCAMC-95),* R. A. Miller ed., Hanley & Belfus, Philadelphia, p. 233-237, 1995.

[Boehm, 1988] Barry Boehm. The spiral model of software development and enhancement. *IEEE Computer*, 21(5), 61-72, 1988.

[Bui et al., 1999] Alex A.T. Bui, Denise R. Aberle, Jonathan G. Goldin, Michael F. McNitt-Gray, Alfonso F. Cardenas, Eric Kleerup, and Osman Ratib. TimeLine: A Multimedia, Problem-centric Visualization of Patient Records. In *Proceedings of the 1999 American Medical Informatic Association Annual Symposium*, 1999.

[Checkland and Scholes, 1990] P. Checkland and J. Scholes. *Soft Systems Methodology in Action*. John Wiley & Sons, Chichester, 1990.

[Duftschmid, 1999] Georg Duftschmid. *Knowledge-based Verification of Clinical Guidelines by Detection of Anomalies*. PhD Thesis, Vienna University of Technology, Vienna, 1999.

[Eason and Harker, 1989] K. D. Eason and S. Harker. *An Open Systems Approach to Task Analysis*. Internal Report, HUSAT Research Centre, Loughborough University of Technology, 1989.

[Friedland and Iwasaki, 1985] Peter E. Friedland and Yumi Iwasaki. The Concept and Implementation of Skeletal Plans. *Journal of Automated Reasoning* 1(2):161-208, 1985.

[Hix and Hartson, 1993] Deborah Hix and H. Rex Hartson. *Developing User Interfaces: Ensuring Usability Through Product and Process*. John Wiley, New York, 1993.

[Hripcsak et al., 1994] G. Hripcsak, P. Ludemann, T. A. Pryor, O. B. Wigertz, and P. D. Clayton. Rationale for the Arden Syntax. *Computers and Biomedical Research*, 27, p. 291-324, 1994.

[Lindwarm et al., 1998] D. Lindwarm, A. Rose, C. Plaisant, K. Norman. Viewing personal history records: A comparison of tabular format and graphical presentation using LifeLines. *Behaviour and Information Technology*, 1998.

[Karam, 1994] Gerald M. Karam. Visualization using timelines. In *Proceedings of the 1994 International Symposium on Software Testing and Analysis*, Seattle, WA, 1994.

[Kosara, 1999] Robert Kosara. *Metaphors of Movement - A User Interface for Manipulating Time-Oriented, Skeletal Plans*. Master's Thesis, Vienna University of Technology, 1999.

[Kosara and Miksch, 1999] Robert Kosara and Silvia Miksch. Visualization Techniques for Time-Oriented, Skeletal Plans in Medical Therapy Planning. In Werner Horn, Yuval Shahar, Greger Lindberg, Steen Andreassen, and Jeremy Wyatt, editors, *Artificial Intelligence in Medicine: Proceedings of the Joint European Conference on Artificial Intelligence in Medicine and Medical Decision Making (AIMDM'99)*. Springer-Verlag, Berlin, 1999.

[Miksch, 1999] Silvia Miksch. Plan Management in the Medical Domain. *AI Communications*, 12(4), 1999.

[Miksch et al., 1997] Silvia Miksch, Yuval Shahar, and Peter Johnson. Asbru: A Task-Specific, Intention-Based, and Time-Oriented Language for Representing Skeletal Plans. In E. Motta, F. v. Harmelen, C. Pierret-Golbreich, I. Filby, and N. Wijngaards, editors, *7th Workshop on Knowledge Engineering: Methods & Languages (KEML-97)*. Milton Keynes, UK, 1997.

[Mullet and Sano, 1995] Kevin Mullet and Darrell Sano. *Designing Visual Interfaces. Communication Oriented Techniques*. SunSoft Press, Prentice Hall, 1995.

[Musen et al., 1992] M. Musen, C. Carlson, L. Fagan, S. Deresinskim, E. Shortliffe. T-HELPER: Automated Support for Community-based Clinical Research. In *Proceedings of the 16th Annual Symposium on Computer Applications in Medical Care (SCAMC-92),* pages 719-723, 1992.

[Musen et al., 1996] Mark A. Musen, Samson W. Tu, Amar K. Das, and Yuval Shahar. EON: A component-based approach to automation of protocol-directed therapy. In *Journal of the American Medical Informatics Association (JAMIA)*, pages 367-388, 1996.

[Negroponte, 1995] Nicholas Negroponte. *Being Digital*. Alfred A. Knopf, New York, 1995.

[Nielsen, 1993] Jakob Nielsen. *Usability Engineering*. Academic Press, Inc., 1993.

[Ohno-Machado et al., 1998] L. Ohno-Machado, J. Gennari, S. Murphy, N. Jain, S. Tu, D. Oliver, E. Pattison-Gordon, R. Greenes, E. Shortliffe, G. Barnett. The GuideLine Interchange Format: A Model for Representing Guidelines. *Journal of the American Medical Informatics Association,* 5(4), p. 357-372, 1998.

[Plaisant and Shneiderman, 1997] Catherine Plaisant and Ben Shneiderman. *An Information Architecture to Support the Visualization of Personal Histories.* Technical Research Report. University of Maryland, 1997.

[Plaisant et al., 1996] Catherine Plaisant, Brett Milash, Anne Rose, Seth Widoff, and Ben Shneiderman. LifeLines: Visualizing Personal Histories. In *Proceedings of CHI '96 Conference: Human Factors in Computing Systems*, ACM, pages 221-227, New York, 1996.

[Plaisant et al., 1998] Catherine Plaisant, Richard Mushlin, Aaron Snyder, Jia Li, Dan Heller and Ben 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, 1998.

[Preece, 1994] Jenny Preece, editor. *Human-computer interaction*. Addison-Wesley, 1994.

[Ravden and Johnson, 1989] Susannah Ravden and Graham Johnson. *Evaluating Usability of Human-Computer Interfaces: a practical method*. Ellis Horwood Books in Information Technology, 1989.

[Rit, 1986] Jean-Francois Rit. Propagating temporal constraints for scheduling. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 383-388. Morgan Kaufman Publishers, Inc., 1986.

[Shahar, 1994] Yuval Shahar. *A Knowledge-based Method for Temporal Abstraction of Clinical Data*. PhD Thesis, Stanford University, Stanford, 1994.

[Shahar and Musen, 1993] Yuval Shahar and Mark A. Musen. RÉSUMÉ: A temporal-abstraction system for patient monitoring. *Computers and Biomedical Research* 26(3): 255-273, 1993.

[Shahar et al., 1998] Yuval Shahar, Silvia Miksch, and Peter Johnson. The Asgaard Project: A Task-Specific Framework for the Application and Critiquing of Time-Oriented Clinical Guidelines. *Artificial Intelligence in Medicine*, 14:29-51, 1998.

[Sherman et al., 1995] E. Sherman, G. Hripcsak, J. Starren, R. Jender, P. Clayton. Using Intermediate States to Improve the Ability of the Arden Syntax to Implement Care Plans and Reuse Knowledge. In *Proceedings of the Annual Symposium on Computer Applications in Medical Care (SCAMC-95)*, R. M. Gardner ed., Hanley & Belfus, New Orleans, Louisiana, p. 238-242, 1995.

[Shneiderman, 1998] Ben Shneiderman. *Designing the User Interface: Strategies for Effective Human-Computer-Interaction*. Addison-Wesley Longman, 3rd edition, 1998.

[Sommerville, 1992] Ian Sommerville. *Software Engineering*. 4th Edition. Addison-Wesley, 1992.

[Stoufflet et al., 1996] P. Stoufflet, L. Ohno-Machado, S. Deibel, D. Lee, R. Greenes. GEODE-CM: A state-transition framework for clinical management. In *Proceedings of the 20th Annual Symposium on Computer Applications in Medical Care (SCAMC-96)*, Hanley & Belfus, Philadelphia, 924, 1996.

[Thimbleby, 1990] Harold Thimbleby. *User Interface Design*. ACM Press, New York, 1990.

[Tu and Musen, 1996] S. Tu and M. Musen. The EON Model of Intervention Protocols and Guidelines. In *Proceedings of the 1996 AMIA Annual Fall Symposium (formerly SCAMC)*, J. J. Cimino, ed., Washington DC., Hanley & Belfus, Inc., Medical Publishers, Philadelphia, pp. 587-591, 1996.

[Tufte, 1983] Edward R. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, Cheshire, Connecticut, 1983.