DIPLOMARBEIT

Interactive Visualization of Time-Oriented Treatment Plans and Patient Data

ausgeführt am Institut für
Softwaretechnik und Interaktive Systeme
der Technischen Universität Wien

unter der Anleitung von
ao. Univ.Prof. Mag. Dr. Silvia Miksch

durch

Wolfgang Aigner

Greiner Straße 45
A-4320 Perg

Wien, am 24. Mai 2003

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

Abstract

This thesis presents methods to support protocol-based care in medicine. Time-oriented treatment plans and patient data are represented visually providing various interaction possibilities to aid execution and analysis of medical therapy plans formulated in the representation language *Asbru*.

We introduce a two-view approach consisting of a *Logical View* and a *Temporal View*. The *Logical View* depicts therapy plans using a flow-chart like representation based on "clinical algorithms". The *Temporal View* on the other hand depicts plans as well as patient data in form of parameters and variables over time. The plan visualization method within the *Temporal View* is based on the idea of *LifeLines*. For being able to depict hierarchical structures and temporal uncertainties, we extended this concept and a novel glyph called *PlanningLine* has been developed.

We present a number of existing visualization approaches addressing concepts that can be found in *Asbru* and discuss their usefulness for our purpose.

A user study with eight domain experts (physicians) was conducted at the beginning of this work to acquire requirements and goals for the development. Furthermore, a software prototype has been implemented and evaluated with five domain experts who judged it as usable and easy to understand. The findings of the user study and the evaluation are presented and discussed.

Zusammenfassung

Diese Diplomarbeit beschreibt Methoden zur Unterstützung von protokoll-basierter Behandlung in der Medizin. Zeitbezogene Behandlungspläne und Patientendaten werden visuell dargestellt und besitzen eine Reihe von Interaktionsmöglichkeiten für die Ausführung und Analyse von medizinischen Therapieplänen, die in der Planrepräsentationssprache *Asbru* verfasst sind.

Wir stellen einen Ansatz bestehend aus zwei Ansichten - einer *logischen* und einer *zeitbezogenen* - vor. In der *logischen Ansicht* werden Therapiepläne mit Hilfe einer flussdiagrammähnlichen Darstellung basierend auf "klinischen Algorithmen" repräsentiert. In der *zeitbezogenen Ansicht* werden auf der anderen Seite sowohl Pläne als auch Patientendaten in Form von Parametern und Variablen über der Zeit dargestellt. Die innerhalb der *zeitbezogenen Ansicht* verwendete Visualisierungsmethode basiert auf der Idee von *LifeLines*. Um sowohl hierarchische Strukturen als auch zeitliche Unsicherheiten darstellen zu können, haben wir dieses Konzept erweitert und eine neuartige Darstellungsweise, genannt *PlanningLine*, entwickelt.

Wir präsentieren eine Reihe von existierenden Visualisierungsmethoden, verschiedene *Asbru* bezogene Aspekte betreffend, und diskutieren ihre Brauchbarkeit für unsere Zwecke.

Eine Benutzerstudie mit acht Domänenexperten (Ärzten) wurde zu Beginn der Arbeit durchgeführt, um die Anforderungen und Ziele für die Entwicklung zu erarbeiten. Weiters wurde ein Software Prototyp implementiert und mit fünf Domänenexperten evaluiert, die ihn hinsichtlich Brauchbarkeit und Verständlichkeit sehr positiv beurteilten. Die Ergebnisse der Benutzerstudie und der Evaluation werden ebenfalls präsentiert und diskutiert.

Contents

Acknowledgements

First of all my advisor, Silvia Miksch, deserves severe thanks for supporting me so well throughout working on this thesis. She was giving valuable tips, help, inspiration, and motivation almost day and night and made working under extraordinarily good circumstances possible.

Thanks to my colleagues Wilfried Reinthaler, Kathi Kaiser, Monika Lanzenberger, Andreas Seyfang, and Peter Votruba for their contributions, discussion and criticism of my ideas as well as for the friendly atmosphere I could carry out my work.

Furthermore, I wish to thank the physicians who agreed in participating in our user study and prototype evaluation. The participants were (in alphabetical order using German titles): Dr. Anton Andonowski, Dr. Veronika Fabrizii, Prof. Dr. Paul Knöbl, Dr. Wolfgang Köstler, Prof. Dr. Christian Popow, Dr. Werner Rabitsch, Prof. Dr. Thomas Staudinger, and Dr. Friedrich Wimazal. They showed a lot of patience when giving deeper insights in the medical domain to a novice in that field. Special thanks to the main medical collaborator Christian Popow and Robert Kosara who both reviewed our design concept giving valuable tips and feedback.

I also want to thank Johannes Gärtner for his help when investigating the reasons for the difficulties in looking for user study participants as well as Christiane and Richard Ehrentraut for their assistance in finding user study participants. Special thanks to Sonja Nösterer, Raphael Schneeberger, and Wilfried Reinthaler for proofreading my thesis.

My flat mates Birgit Hörmanseder, Maria Pesendorfer, and Raphael Schneeberger also deserve thanks for their support and for providing a lovely, relaxing background.

Last but definitely most importantly, special thanks go to my grandmother Emma, my parents Franz and Maria, and my sister Elisabeth for their love and great support during all the years. It was my family who have given me the choice and opportunity to attend university, which I appreciate a lot and do not take for granted.

This thesis was supported by “Fonds zur Förderung der wissenschaftlichen Forschung - FWF” (Austrian Science Fund), grant P15467-INF.

Chapter 1

Introduction

"The purpose of computing is insight, not numbers."
R. W. Hamming, 1961.

1.1 Motivation

The use of clinical guidelines and treatment plans[1] has a profound history in the medical field and is widely accepted. The purpose of these documents is primarily to enable discussion and communication among physicians as well as improving the quality of care by setting up certain "standard procedures".

Most clinical guidelines are represented in plain text, rarely including tables and flow-charts [2]. Computer support is desirable for various reasons: First of all obviously to ease document exchange, editing and reuse. Furthermore, due to the semiformal structure of such documents, taking it a step further towards creation, runtime and analysis support introduces a much higher level of aid. Tool support in this sense helps to cope with ongoing information overflow, eases quality assessment, and facilitates information respectively knowledge exchange.

Planning techniques are well known in the field of Artificial Intelligence (AI). Various approaches and solutions to solve planning problems have been carried out by the AI-, respectively planning community since the 1960s. But most of these techniques can only be applied to well defined and limited problems. The real-world domain *medicine* requires much more flexibility: Well defined problems and processes are the exception rather than the rule, and uncertainty is an important factor. Traditional planning approaches are not able to support these requirements. Therefore, special techniques for medical plan management, like *Asgaard/Asbru* [Miksch, 1999] and others [Wang et al., 2001], have been carried out.

Most of those projects concentrate on the formal methods to describe plans, validate them and provide methods and tools for their execution or analysis. From

[1] Throughout this thesis, the expressions *clinical guideline*, *guideline*, *treatment plan* and *plan* will be used interchangeably

[2] Flow-Charts that are used to describe procedures are widely known as **algorithms**, **flow-chart algorithms** or **clinical algorithms** in the medical domain [Society for Medical Decision Making, 1992,Hadorn, 1995]. Due to the fact that the term *algorithm* has a different meaning in information technology, we use the notion *flow-chart* throughout this thesis.

a man-machine point of view, they focus almost entirely on the "machine end" of the *man-machine chain*. Very little work has been carried out to bridge the gap between the formal methods used "behind the scenes" and the end-user: physicians, nursing and other medical personnel. This work is intended to fill this gap. It aids physicians during plan execution by providing an easy-to-use, intuitive interface approach towards medical treatment planning, starting from the user's point of view.

1.2 Background

Before we start examining the prerequisites and requirements of this work, some words should be said about the project this work is part of, the *Asgaard* project.

The need to improve the quality of health care has led to a strong demand for clinical protocols and computer systems supporting both their creation and execution. This demand was the basis for the birth of the Asgaard project [Miksch, 1999]. The participants in this project were initially inspired by the planning community. Due to the shortcomings of the traditional approaches, the Asgaard/Asbru project has been trying to build the bridge between 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. The designers have developed a time-oriented, intention-based, skeletal plan-specification language, called *Asbru*, specific to the set of plan-management tasks [Miksch, 1999].

1.3 Overview of this Thesis

After presenting the motivation and background for this thesis, we analyze the requirements for our work in Chapter 2 along with stating the goals we want to achieve.

In Chapter 3, a State-of-the-Art report of visualization methods and medical software products addressing concepts that can be found in *Asbru* is given and their usefulness for our purpose is discussed. Furthermore, special *User Interface Design* and *Human Computer Interaction (HCI)* issues regarding the medical domain are presented.

The following chapter (Chapter 4) describes the user study we conducted in parallel to the problem analysis for acquiring requirements and imaginations from domain experts (physicians) as well as gaining deeper insights into the medical domain, work practices, and the use of guidelines in daily work. The results of this study are discussed and a user model along with use scenarios is resembled, describing tasks and goals of users.

The core part of this thesis is the "Conceptual Design" chapter (Chapter 5). It presents our design ideas, especially the elements of the two views along with their interaction possibilities.

After that, we describe the implementation of our software prototype in terms of architectural issues, certain implementation issues, and which parts of the design have been implemented or were left out. Then a report about our prototype

evaluation process (Chapter 7) follows, explaining how the evaluation was done and discussing its results.

Chapter 8 presents in detail how we implemented the *User Centered Design* approach in our development process. After that we recapitulate our work and present the lessons learned from carrying out this thesis in Chapter 9 and finally we present what has to be done in future in Chapter 10.

The appendices include definitions of basic terminology and concepts, all used interview guidelines and forms in their German original and English translated version, an example guideline modeled using the *Logical View* representation, and a listing of all used UML class diagram elements.

1.4 Conventions

The work presented here was done in context of the *Asgaard* project respectively the *Asgaard* project team. Most design and architectural ideas were carried out by me, but all that was formed on basis of discussion and input especially by my advisor and other colleagues. That is why I decided to use the pronoun “we” rather than “I” throughout this thesis.

Chapter 2

Problem Analysis

2.1 A Short Introduction to Asbru

Asbru [Seyfang et al., 2002, Miksch et al., 1997] is a time-oriented, intention-based, skeletal plan-specification representation language that is used in the *Asgaard* Project[1] to represent clinical guidelines and protocols. Asbru can be used to express clinical protocols as skeletal plans [Friedland and Iwasaki, 1985] that can be instantiated for every patient (for an example see Fig. 2.1). It was designed specific to the set of plan-management tasks [Miksch, 1999]. Asbru enables the designer to represent both the prescribed actions of a skeletal plan and the knowledge roles required by the various problem-solving methods performing the intertwined supporting subtasks. The major features of Asbru are that

- prescribed actions and states can be continuous;
- intentions, conditions, and world states are temporal patterns;
- uncertainty in both temporal scopes and parameters can be flexibly expressed by bounding intervals;
- plans might be executed in sequence, all plans or some plans in parallel, all plans or some plans in a particular order or unordered, or periodically;
- particular conditions are defined to monitor the plan's execution; and
- explicit intentions and preferences can be stated for each plan separately.

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

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

Intentions Intentions are high-level goals that should be achieved by a plan, or maintained or avoided during its execution. This information is very important not

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

only for selecting the right plan, but also for critiquing treatment plans as part of the ever ongoing process of improving the treatment. This makes intentions one of the key parts of Asbru.

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

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

Plan Body (Actions) The plan body contains plans or actions that are to be performed if the preconditions hold. A plan is composed of other plans, which must be performed according to the plan's type (Table 2.1): in *sequence*, in *any order*, in *parallel*, *unordered*, or *periodically* (as long as a condition holds, a maximum number of times, and with a minimum interval between retries).

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

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

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

2.2 Phases of Plan Management

In principle, *Asgaard* is aimed to aid plan management during the following three phases:

- Design Phase
- Execution Phase
- Analysis Phase

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

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

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

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

Figure 2.1: An example of Asbru code (part of a clinical treatment protocol for Infants' Respiratory Distress Syndrome (I-RDS)).

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

Table 2.1: Plan Types in Asbru.

Design Phase During *design time* of a clinical guideline, an author (or a committee) designs a guideline. The author prescribes *conditions*, that need to hold at particular plan steps (e.g., subject is female, blood gas has been above the target range for at least five minutes), *actions* (e.g., administer a certain drug in the morning and in the evening), an *intended plan* - the intended intermediate and overall pattern of actions, which might not be obvious from the description of the prescribed actions and is often more flexible than prescription of specific actions (e.g., use some drug from a certain class of drugs twice a day), and the *intended* intermediate and overall pattern of *patient states* (e.g., morning blood glucose should stay within a certain range).

At *design time*, the following issues are addressed:

- Plan Generation
- Advanced Plan Editing
- Plan Verification
- Plan Validation
- Plan Visualization
- Plan-Scenario Testing

Execution Phase During *execution time*, a physician/nurse applies the guideline by performing *actions*, which are recorded, observed, and abstracted over time into an *abstracted plan*. The *state* of the patient is also recorded, observed, and abstracted over time. Finally, the *intentions of the medical staff* might be recorded too - inferred from her actions or explicitly stated by the medical staff.

At *execution time*, the following issues are addressed:

- Plan Selection
- Plan Instantiation
- Data Abstraction
- Monitoring
- Execution Visualization
- Critiquing / Evaluation
- Plan Rationale / History

Analysis Phase After and during execution of a plan, recorded data is reviewed and analyzed. This includes data about the execution time of plan steps, patient state and data, parameter and variable values over time, as well as intentions and notes taken during the execution of a guideline. Furthermore, several executions of a guideline can be compared to detect i.e. similarities or differences between them.

During *analysis*, the following issues are addressed:

- Visualization
- Critiquing / Evaluation
- Plan Rationale / History
- Execution Comparison

2.3 AsbruView

AsbruView [Kosara and Miksch, 2001a, Kosara and Miksch, 1999] is a tool to make *Asbru* accessible to physicians, and to give any user an overview over a plan hierarchy. *AsbruView* is based on visual metaphors to make the underlying concepts easier to grasp. This was done because not only the notation is foreign to physicians, but also the underlying concepts.

AsbruView consists of three views: *Topological View*, *Temporal View*, and *SOPOView*. The *Topological View* mainly displays the relationships between plans, without a precise time scale. The *Temporal View* concentrates on the temporal dimension of plans and conditions. The *Topological View* uses graphical metaphors and the *Temporal View* uses glyphs to make the underlying concepts easier to understand. The *SOPOView* is a different view to capture the temporal dimensions of plans.

Topological View The basic metaphor in this view is the **running track** (see Fig. 2.2). Every plan is displayed as one such track, with metaphors to symbolize different parts. The time dimension (from left to right) is only a symbolic one, plans' sizes do not reflect their planned duration. The traffic signs symbolize the preconditions: The "no entrance with exception" sign stands for the *filter precondition* (which has to be true for the plan to be applicable at all); the barrier symbolizes the *setup precondition* (which has be true as well - but which can be achieved by other plans if it is not - for the plan to be applicable).

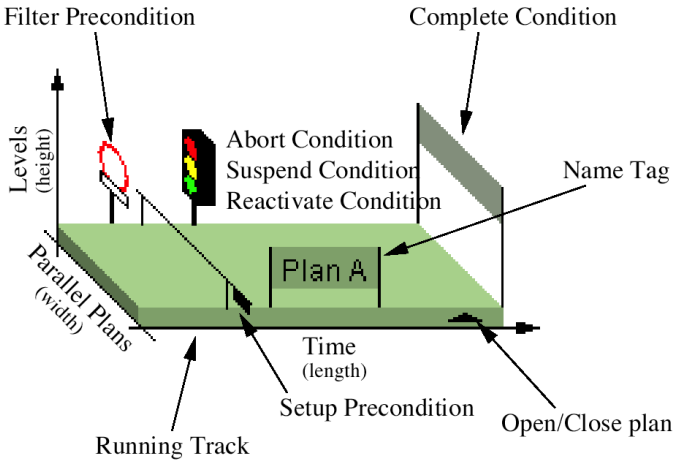


Figure 2.2: Anatomy of an AsbruView Plan in Topological View [Kosara and Miksch, 2001a].

The lights of the traffic light each stand for one further condition:

- **red**: the *abort condition* (which specifies when the plan has to be stopped and regarded as failed).
- **yellow**: the *suspend condition* (which defines when a plan has to be interrupted to treat an emergency, for example).
- **green**: the *reactivate condition* (which specifies when a suspended plan can be continued).

The finishing flag, finally, symbolizes the *complete condition*, which specifies when the plan has reached its goal and can be considered successful.

Plans can have *subplans*, which are then stacked on top of the containing plan (the so-called superplan).

Temporal View In addition to the topological information, physicians need to be able to see the details of the temporal extensions of plans. For this purpose, the *temporal view* is used.

It consists of a display that represents each plan with a so-called glyph, i.e., a graphical object whose features change with the values they depict (see Fig. 2.3). It

shows the four time points that mark the earliest start (ESS, earliest starting shift), latest start (LSS, latest starting shift), earliest end (EFS, earliest finishing shift), and latest end (LFS, latest finishing shift). Lying on these points (similar to parts of a bridge on their supports), are bars that represent the minimum and maximum duration of the plan (MinDu and MaxDu) – which are not 100% dependent on the start and end points, but are constrained by them (i.e., the MaxDu cannot be longer than the time between ESS and LFS, but can be shorter, so an action can take place during five hours, for example, but may not take longer than 15 minutes).

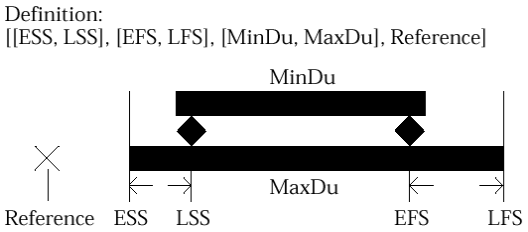


Figure 2.3: Time glyph of AsbruView [Kosara and Miksch, 2001a].

The *plan hierarchy* is also shown in this view in a way similar to tree views like those used in file managers. Tree branches can be folded, so that parts of the plan that are not currently of interest are hidden, and there is more space for the interesting parts. Symbols left of the list of plans show the plan type, e.g., sequential plans have a bullet next to every sub-plan, parallel plans have two parallel lines, and any-order plans have arrows pointing up and down (see figure below). The plans with a questionmark-texture are optional, i.e., they do not have to be finished for the containing plan to succeed, but can be dependending on their conditions.

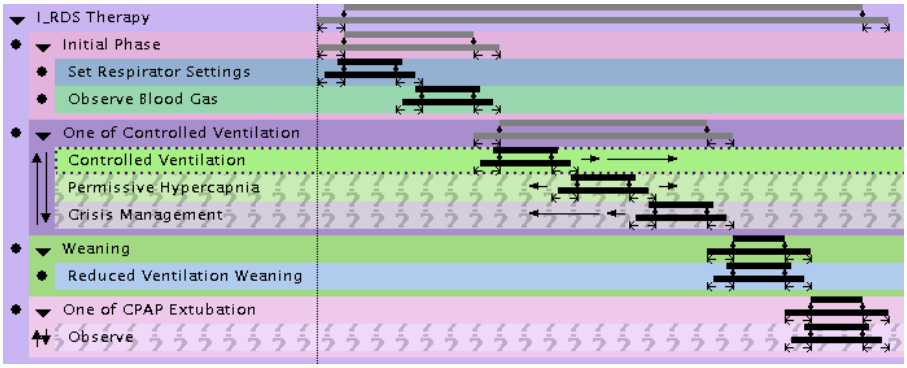


Figure 2.4: Temporal view [Kosara and Miksch, 2001b].

SOPOView Originally, sets of possible occurrences (SOPOs) were designed for the easy graphical propagation of temporal constraints [Rit, 1986], and not for making a complex notion of time easy to understand. Essentially, a diagram with two time axes is used that represent the begin and end times of an interval, respectively

(see Fig. 2.5). Any point in this diagram represents a whole interval, specified by its start (x-coordinate) and end time (y-coordinate). The area a SOPO covers (black in the figure) contains all intervals that fit the specification given by means of an earliest start, latest start, earliest end, latest end, minimum and maximum duration (thus enabling the representation of temporal uncertainty we were looking for).

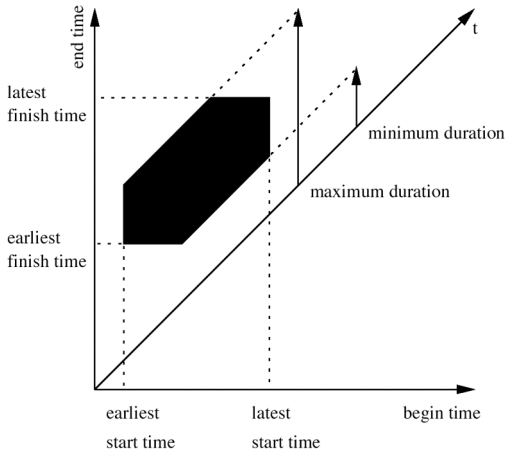


Figure 2.5: Sets of Possible Occurrences (SOPOs) [Messner, 2000].

In order to meet the given requirements of Asbru described above, the basic concept of SOPOs had to be extended in many ways [Messner, 2000]: The usage of colors for different plans' SOPOs as well as background colors indicating the hierarchical decomposition of plans was introduced. Furthermore, the position of axes have been changed for a better suitability when scrolling within the diagram becomes necessary. The prototype's user interface also provides means to change the diagram's resolution (e.g., from minutes to hours) as well as to zoom in and out the diagram. Also, a way to depict undefined parts by using zig-zagged edges was found. For an example of SOPOView, see Fig. 2.6.

To provide basic user interaction mechanisms, a direct manipulation of SOPOs has been implemented: a SOPO can be clicked, thus marking the underlying plan. To change the plan's time annotations, the SOPO can either be dragged as a whole and dropped within the diagram (with certain restrictions due to the language Asbru), or its edges can be dragged, thus changing only parts of the underlying time annotation (e.g., latest start time). Also, an editor can be used to change a plan's time annotations.

As plans may consist of subplans/actions (hierarchical decomposition), their corresponding SOPOs can be opened or closed, to reveal or hide their subplan's time annotations.

2.4 Requirements

AsbruView, the tool presented in the last section, is used for creating *Asbru* plans and **not** applicable for runtime visualization. In contrast to that, the task of this

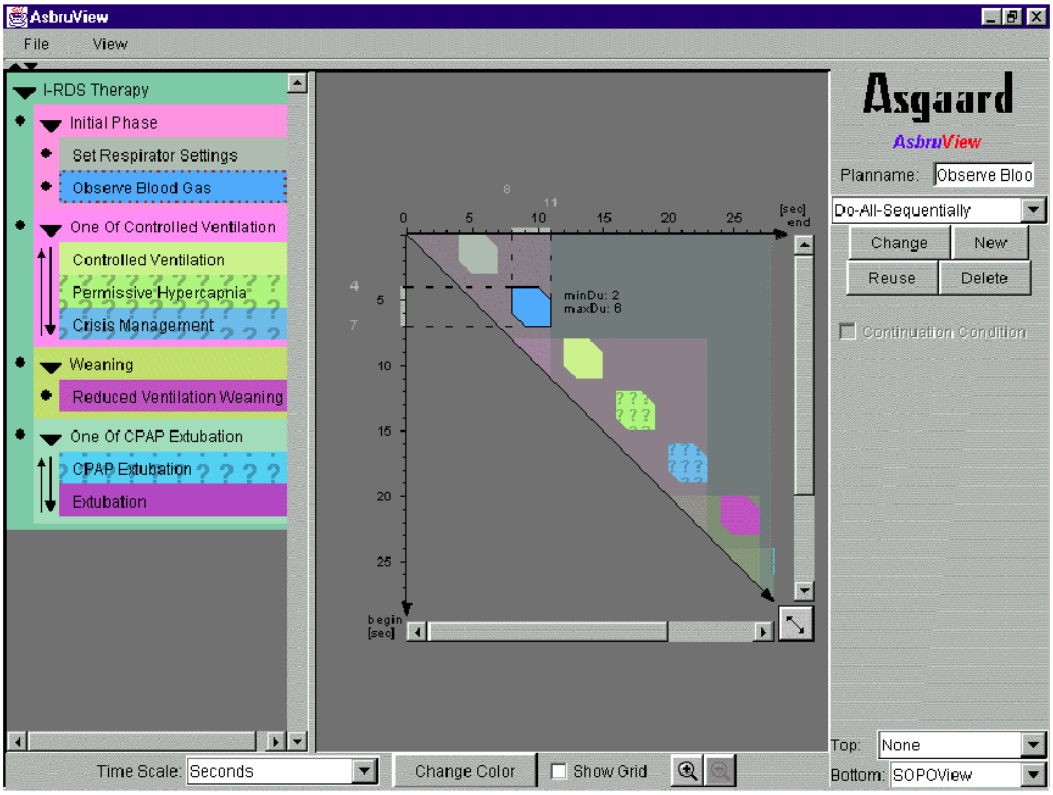


Figure 2.6: Screenshot from the AsbruView program showing SOPOView. [Messner, 2000].

work is to develop methods for visualization and interaction within time-oriented patient data and treatment plans at runtime. This includes:

- Graphical visualization of treatment plans defined in the plan modeling language *Asbru* for communicating the current position within a plan and depicting execution data for the past as well as the future.
- Visualization of patient data.
- Visualization of measured values (time-oriented).
- Analysis and comparison of measured values and plan execution over time.
- Display of decisions to come up and possible future directions as well as support for choosing upcoming actions respectively treatment steps.

The methods we are going to develop are situated in the *execution* and *analysis phase* (see Section 2.2) and mainly aimed towards presentation rather than data entry.

From section 2.1 we can abstract the following visualization relevant data characteristics from *Asbru*:

- Time-oriented data from the past to future planning data including a rich set of time attributes allowing to represent uncertainties (execution time attributes and time annotations)
- Hierarchical data (subplans)
- Elements having execution sequence relationships (sequentially, parallel, any-order, unordered)
- Non-uniform element types (plans, if-then-else, ask, cyclical plans, variable assignments)
- Conditions having state characteristics (abort-condition, complete-condition)

These characteristics have to be represented visually in a clear, simple, and intuitive way for communicating them to the user.

In this work we basically focus on developing visualization and interaction methods to represent plans rather than parameters and variables.

2.5 Goals

The goals for developing such a tool are to provide a considerable treatment documentation, offer better analysis support by using appropriate visualization methods, merge various information sources, improve therapy, and help to concentrate on the essentials in the daily routine.

These goals represent vital improvements and advantages for the work process of physicians.

Chapter 3

State of the Art

Information Visualization in the medical domain using *Asbru* relates to several scientific fields and topics as identified in Chapter 2.

- Information Visualization
  - Information Visualization in Medicine
  - User Interface Design and Human Computer Interaction in Medicine
- Data
  - Hierarchical Data
  - Time-oriented Data[1]
  - Logical Sequence Data
- Tasks
  - Process Visualization
  - Medical Guideline Visualization
  - Visualization in Project Management
- Techniques
  - Focus + Context
  - Graph Visualization

These topics served as a starting point for the research of related work and the state of the art in science.

[1] *Asbru* includes a comprehensive time annotation consisting of the following attributes: Reference Point, Earliest Starting Shift (ESS), Latest Starting Shift (LSS), Earliest Finishing Shift (EFS), Latest Finishing Shift (LFS), Minimum Duration (MinDu) and Maximum Duration (MaxDu). Each part may or may not be defined in a time annotation.

3.1 General

3.1.1 Information Visualization

As technology improves steadily, especially in medicine, the amount of data is growing even faster and in huge steps. This growing amount of data is overwhelming and cannot be captured or analyzed without applying certain methods like Information Visualization.

Information Visualization, in contrast to Scientific Visualization, focuses on abstract data and information where a natural mapping to the physical world may not exists (i.e. Databases, Networks, Documents, Time, Hierarchies). Interaction is an important feature of Information Visualization, allowing the user to explore and work with the data interactively. Furthermore, Information Visualization proposes emphasizing techniques to ease cognition and tries to involve the user by making her an active element in pattern recognition [Chittaro, 2001]. Examples for these techniques are Focus + Context, Distortion, Highlighting, Overview + Detail, Filtering, or Brushing. *"[...] the purpose of information visualization is to use perception to amplify cognition."* [Card et al., 1998, p. 10].

3.1.2 User Interface Design and HCI in Medical Software Development

There is a lot of literature concerning User Interface Design and HCI (Human Computer Interaction) in general available, but when it comes to special problems and issues about software in medicine, only little research has been carried out [Gosbee and Ritchie, 1997, Holzman, 1999, Marcus et al., 2000].

Domain Analysis and Understanding the User

Basic insights about the medical domain and issues concerning developing medical software are given by Gosbee and Ritchie in [Gosbee and Ritchie, 1997]. Fundamental for their research is to gain knowledge about the structures in the medical domain on various levels such as the health system in general, laws and legal issues, administration, hospital internal hierarchies or distribution of competences. Another important step prior to a user study is to identify key personnel in various areas (i.e. head nurse). Because it is likely that the desired personnel cannot be included in a user study due to various reasons, one should think about feasible surrogates like for example medical students.

Gosbee and Ritchie [Gosbee and Ritchie, 1997] point out working conditions physicians have to face and include the example *"A day in the Life of an Internal Medicine Physician"* which leads to the observation that most physicians are stressed, have very little time and have to make a lot of decisions. Therefore, this issues have to be kept in mind and special measures have to be taken when dealing with physicians: i.e. appointments can possibly be postponed, cancelled or there might be disruptions due to emergencies or other important incidents. Hence, above all flexibility is important.

Knowledge of the medical terminology is a further issue of major importance. Only if the analyst knows and understands medical terms and basics, she is capable of talking to the user at the correct level and is taken seriously by physicians.

User Interface Design

Marcus et alċarried out a study [Marcus et al., 2000] for *Kaiser Permanente* where they pointed their spotlight onto User Interface (UI) Design issues for Medical Informatics. They examined development aspects of a Windows™ based clinical information system developed by *Kaiser Permanente*.

One of their design principles is to use structures and processes familiar to the end users. This includes metaphors, mental models, navigation, appearance as well as interaction issues. They also point out the importance of user interface guidelines to assure a uniform appearance and interaction model. These guidelines include details about layout, icons and symbols, typography and terminology, color, aesthetics as well as language and verbal style.

3.2 Data Visualization

3.2.1 Visualizing Hierarchical Data

Since data of all kinds often has hierarchical structure, hierarchical data and the need for its visualization are common problems in information technology.

Trees

The most often used technique for presenting hierarchical structures are *trees*. They are very well known and widely used by almost everyone who has ever worked with a computer.

Probably the most prominent examples for tree view applications are file system viewers such as the Microsoft™ *Windows Explorer* or the *Finder* application of the Macintosh™ system. Common characteristics among these tree view applications are their simple layout including expandable nodes which allow easy navigation within the hierarchy.

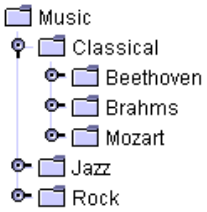


Figure 3.1: Tree View.

Problems arise if large trees have to be displayed, in this case, overview is lost very easily and clever layout becomes an important issue. Trees are basically a very

good and flexible way to visualize hierarchies but do not have a notion to include time.

Treemaps

Treemaps [Johnson and Shneiderman, 1991] are a rectangular, space-filling approach for visualizing hierarchical data. They use 2D visualization of trees where the tree nodes are encapsulated into the area of their parent node. The size of the single nodes is determined proportionally in relation to all other nodes of the hierarchy by an attribute of the node.

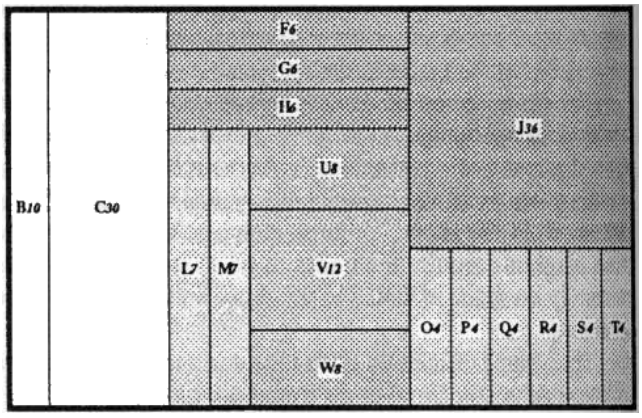


Figure 3.2: Treemap [Johnson and Shneiderman, 1991].

Summary

We have presented two of the most prominent methods for visualizing hierarchical data in 2D. The first one, *Trees*, focus on depicting the structure, whereas the second one, *Treemaps*, introduces a further dimension it is emphasizing by proportional space assignment.

But besides that, they have no notion for depicting time, flow, or concurrency.

3.2.2 Visualizing Time-Oriented Data

Time Lines and LifeLines

A simple and intuitive way of depicting incidents is by drawing a horizontal line for the time span the incident took. This form of visualization is called *Time Line* which is a very powerful visualization technique used long before computers even appeared [Tufte, 1983].

Ben Shneiderman, Catherine Plaisant et alöf the University of Maryland used Time Lines for Information Visualization and added some concepts as for example Facets. This further development is called LifeLine. They applied LifeLines for visualizing personal histories and patient information [Plaisant et al., 1996, Plaisant et al., 1998].

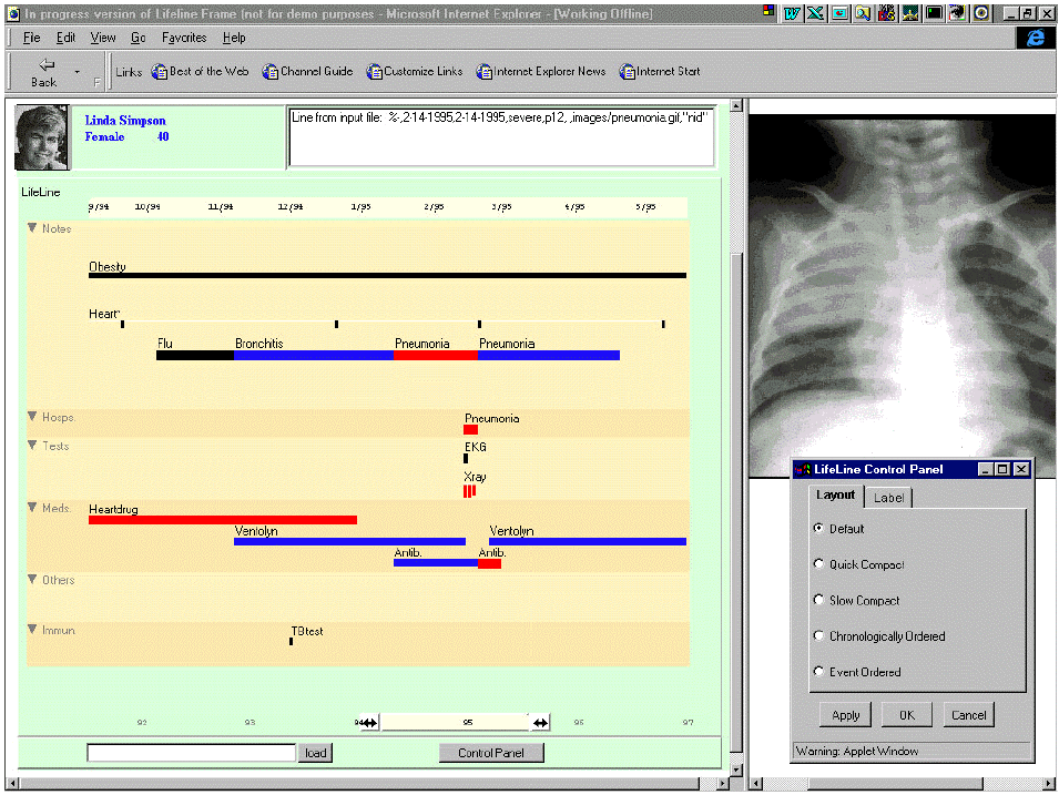


Figure 3.3: LifeLines used for visualizing Patient Records [Plaisant et al., 1998].

LifeLines are basically simple horizontal bars showing time-oriented data. Due to their simplicity they are easy to understand but some important features such as the ability of depicting hierarchical data are missing. Many new information visualization approaches are based on the idea of LifeLines, adding several extensions, as described later.

Temporal Objects

Temporal Objects [Combi et al., 1999] have been developed to depict complex time annotations and are similar to the Time Glyph used in *AsbruView* (see Section 3.5.1). Primarily, they are an extension of the familiar LifeLine method: two encapsulated bars with caps at each end present the attributes starting instant (ESS, LSS), ending instant (EFS, LFS), minimum and maximum duration (see Fig. 3.4).

The visual appearance of a *Temporal Object* emphasizes the extent of the glyph as a whole because the outer caps have a solid fill and seem to be solidly connected to the inner bars (also see gestalt principles [Card et al., 1998]). *Temporal Objects* cannot be as easy edited via direct manipulation but are probably easier to understand as the Time Glyph of *AsbruView*.

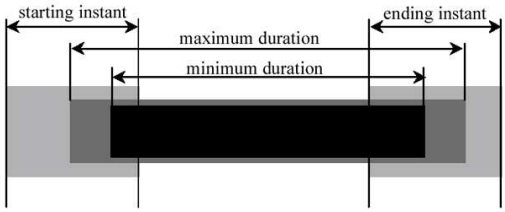


Figure 3.4: Temporal Object [Combi et al., 1999].

Paint Strips

Paint Strips [Chittaro and Combi, 2001] are basically also an extension to the well known LifeLines technique. The idea of LifeLines is enriched by a painting metaphor indicating that the displayed LifeLines are drawn by a paint roller. A paint roller at an end of a LifeLine means that this line can expand by moving the roller until the wall is reached. This way the maximum duration and earliest start or latest end, depending on which end of the painting strip they are attached to, are defined.

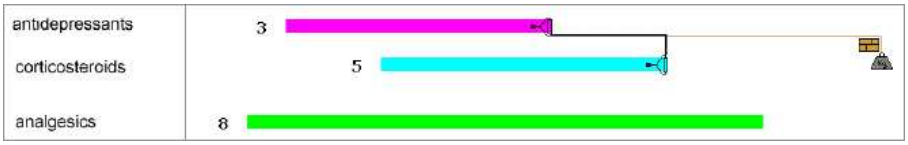


Figure 3.5: Paint strips [Chittaro and Combi, 2001].

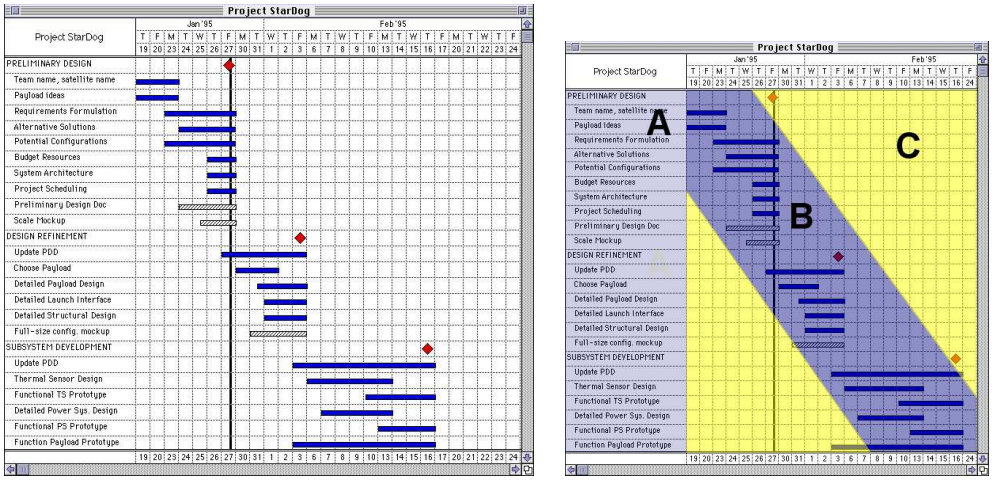
Another addition is the possibility to combine strips. The relationship of paint strips can be fixed, which means that if one strip moves, the other one moves in the same extent as well. This relationship is indicated graphically by connecting the involved paint rollers and attaching them to a weight at the end of a "rope", which is able to move the rollers (see Fig. 3.5).

Paint Strips were especially developed for medical applications but can be used elsewhere as well. Due to the simplicity of the paint strip metaphor, some time annotation attributes such as durations independently of the differences between start and end points, different time granularities, undefined values or a reference point cannot be visualized.

GANTT Charts

GANTT charts are a very well known diagraming technique mainly used in project management. They display both, hierarchy and time annotation graphically within a single view.

The tasks are displayed as a list at the left border of the diagram (see Fig. 3.6b, part A) and related tasks can be grouped to form a hierarchy. For displaying the extent of tasks in time, LifeLines, that are drawn at the respective lines of the task list, are used (see Fig. 3.6b, part B).



(a) (b)

Figure 3.6: GANTT charts. Example (a), Diagonal layout (b).

The fact that tasks are mostly ordered chronologically leads typically to a diagonal graphical layout from the upper left to the lower right corner of the display (see Fig. 3.6b). The overall space consumption of the diagram is rather high, but includes much unused space in the lower left and upper right corners (see Fig. 3.6b, part C).

SOPOs

An additional representation for *AsbruView* (see Section 3.5.1) is called *SOPO View* [Messner, 2000, Kosara et al., 2001] and utilizes **Rit's Sets of Possible Occurences (SOPOs)** (see Fig. 3.7). The axes of the diagram are used to depict start interval (x-axis) and end interval (y-axis). Minimum and maximum duration are the constraining borders parallel to the 45° time flow axis. Thus, the area a *SOPO* covers contains all intervals that fit the specification given by means of earliest start, latest start, earliest end, latest end, minimum, and maximum duration.

SOPOs were designed for the easy graphical propagation of temporal constraints, not for making a complex notion of time easy to understand. Specifically, parallel plans and hierarchical decomposition are very hard to depict and to work with and a notion of undefined parts is missing in the original design. [Kosara and Miksch, 2002]

Summary

Time is a very important data characteristic but methods for visualizing time other than in diagrams are not well known. The probably best known method among them are *GANTT charts* and their utilized *Time Lines*. An extension of *Time Lines* are *LifeLines* that have been used for example to visualize personal histories.

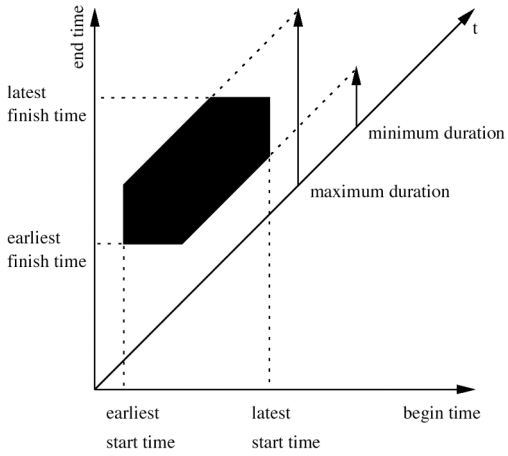


Figure 3.7: Sets of Possible Occurrences (SOPOs) [Messner, 2000].

A drawback of these methods is that they mostly work retrospectively, thus just accurately depicting temporal attributes in the past. To overcome this limitation, other visualization techniques like *Temporal Objects*, *Paint Strips*, and *SOPOs* were developed. These techniques are able to visualize complex notions of time like temporal uncertainties that can be utilized to depict future planning data.

The main flaw of the presented techniques is that except *GANTT charts* they cannot depict hierarchies, and logical sequences can only be represented implicitly.

3.2.3 Visualizing Logical Sequences

PERT Charts

PERT charts are another diagraming technique mainly used in project management. It consists of boxes and arrows whereas boxes are tasks and arrows depict their temporal and logical relationship. So basically they depict tasks with their predecessors and successors. Exact temporal information, like earliest start, latest start, earliest end, latest end, minimum duration and maximum duration is represented textually within the boxes but cannot be depicted graphically.

Compared to *GANTT charts*, relationship and order of tasks are visualized explicitly and more clearly. Therefore *PERT charts* are often used for determining "critical paths" of a project or depicting them visually. A flaw of *PERT charts* is that they do not provide a notion for displaying hierarchies.

In contrast to *GANTT charts*, *PERT charts* are generally not as well known in domains other than project management.

Flow Charts

Flow charts for representing algorithms were already designed back in 1947 [Goldstine and von Neumann, 1947]. *Flow charts* are a widely used and very well known diagraming technique for various applications such as for example software engineer-

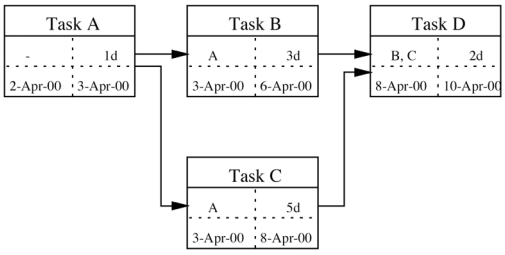


Figure 3.8: PERT chart.

ing. *Flow charts* are probably the prototype of so called *box and arrow diagrams* with a relative small number of different symbols used.

Due to their universal applicability, *flow charts* are also used for representing *medical algorithms*. The *Committee on Standardization of Clinical Algorithms* of the *Society for Medical Decision Making* proposed a standard for representing clinical algorithms graphically [Society for Medical Decision Making, 1992]. *"However, since algorithmic logic is wired implicitly into a protocol, it is difficult to learn an algorithm from a protocol. By contrast, flow-chart algorithms, or clinical algorithm maps, are uniquely suited for explicitly communicating conditional logic and have therefore become the main format for representing a clinical algorithm clearly and succinctly."* [Society for Medical Decision Making, 1992] The proposed standard includes a small number of different symbols and some rules on how to use them (see Fig. 3.9). One additional feature to standard *Flow charts* are *annotations*, that include further details i.e. citations to supporting literature, or clarifications for the rationale of decisions.

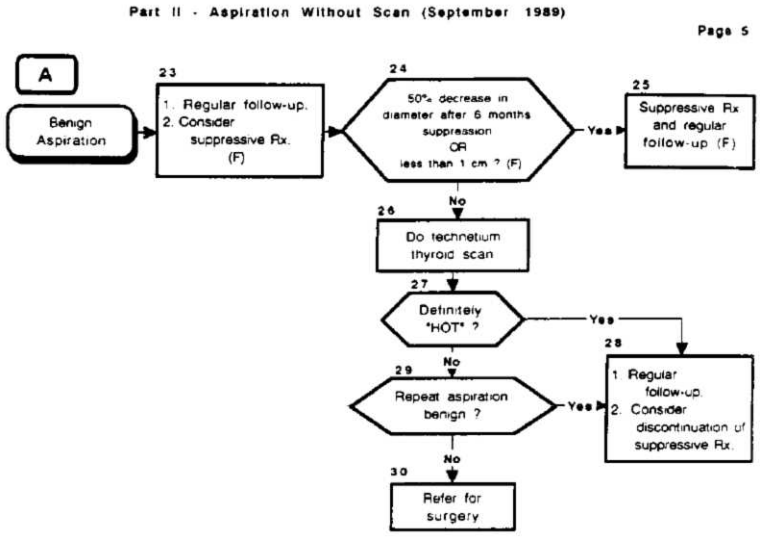


Figure 3.9: Clinical algorithm [Society for Medical Decision Making, 1992].

A big advantage of using *Flow charts* is that they are well known among physicians and require minimal additional learning effort. A drawback of *Flow charts* is their immense space consumption if more complicated situations are depicted. This leads to diagrams that are spread among various sheets of paper where overview is lost easily. Furthermore, *Flow charts* can not be used to represent time, concurrent tasks or complex conditions like the suspend or reactivate conditions used in *Asbru*.

Structogram (Nassi-Shneiderman-Diagram)

Structograms are a diagram type that has been designed especially for software development [Nassi and Shneiderman, 1973]. It tries to avoid problems present in *Flow chart diagrams* and generate a more uniform and clear way to depict program flows. *Structograms* are basically formed out of encapsulated rectangular boxes. The program syntax is partly implicitly contained within the diagram layout: Basic programming patterns like loops or conditions are visualized by special box styles (see Fig. 3.10). This means that for example a for-loop does not have to be modeled combining three boxes (*box 1* for setting the loop variable, *box 2* for checking the condition and *box 3* for manipulating the loop variable as necessary when using *Flow charts*) - it is modeled by a special for-loop box, just including one text line, thus presenting the for-loop flow implicitly. For keeping an overview, more detailed parts of the diagram can be collapsed.

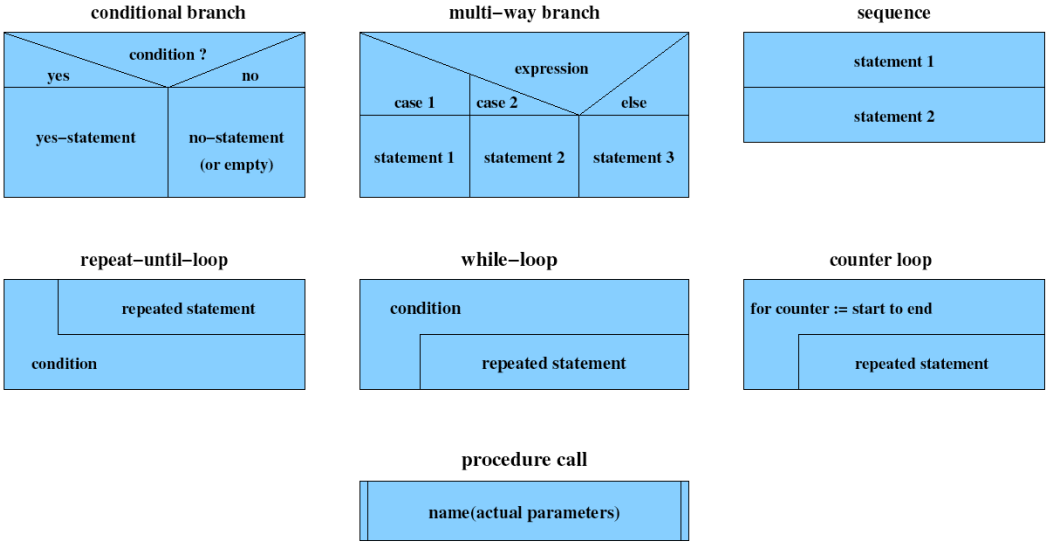


Figure 3.10: Structogram Elements.

Except of the software development and engineering community, *Structograms* are not well known. The method of packing more structural information into special diagram parts and disclaiming arrows leads to a strict rectangular and more cleaned up looking layout, but introduces a more costly step of learning the different diagram parts.

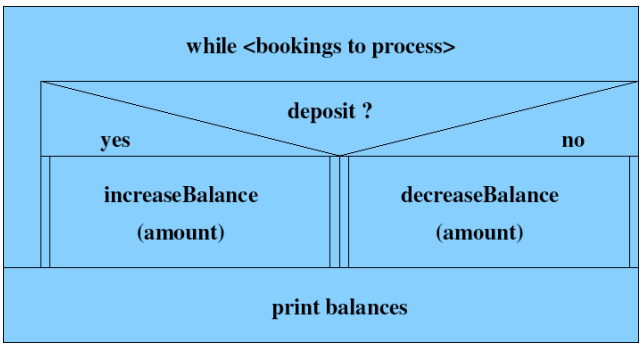


Figure 3.11: Structogram Example.

Process visualization

Most of the available products or research topics that have been examined in the field of process visualization are concerned either with production site visualization where a real world mapping is given by the geographical layout of the site or visualization techniques using measurement gauges [Rieger, 1999].

Summary

Depicting logical sequences is probably the most widely performed Information Visualization task. The most prominent representative of techniques for that matter are *Flow charts*. Due to their simplicity and versatility they are very well known and used in almost every possible application area.

Other techniques are *PERT charts* and *Structograms* which in contrast focus on special purposes, namely project management and software engineering.

The presented methods are perfectly suited to represent logical sequences, but have no notion for depicting time or hierarchy explicitly.

3.3 Focus + Context Methods

Information visualization has to deal with two limited resources [Card et al., 1998]:

- for the user: attention over time
- for the computer: space time (use of screen space over time)

Viewing detailed information (focus) while keeping overview (context) is the premise for Focus + Context techniques [Card et al., 1998]. In contrast to Overview + Detail techniques, this goal is accomplished within one single view applying selective aggregation and distortion. This transformation is formally determined by visual transformation functions which are either data transformations, view transformations, or change of representation (semantic scaling).

Leung and Apperley [Leung and Apperley, 1994] analyzed various distortion oriented presentation techniques trying to build a general taxonomy. They extracted two basic categories of Focus + Context techniques:

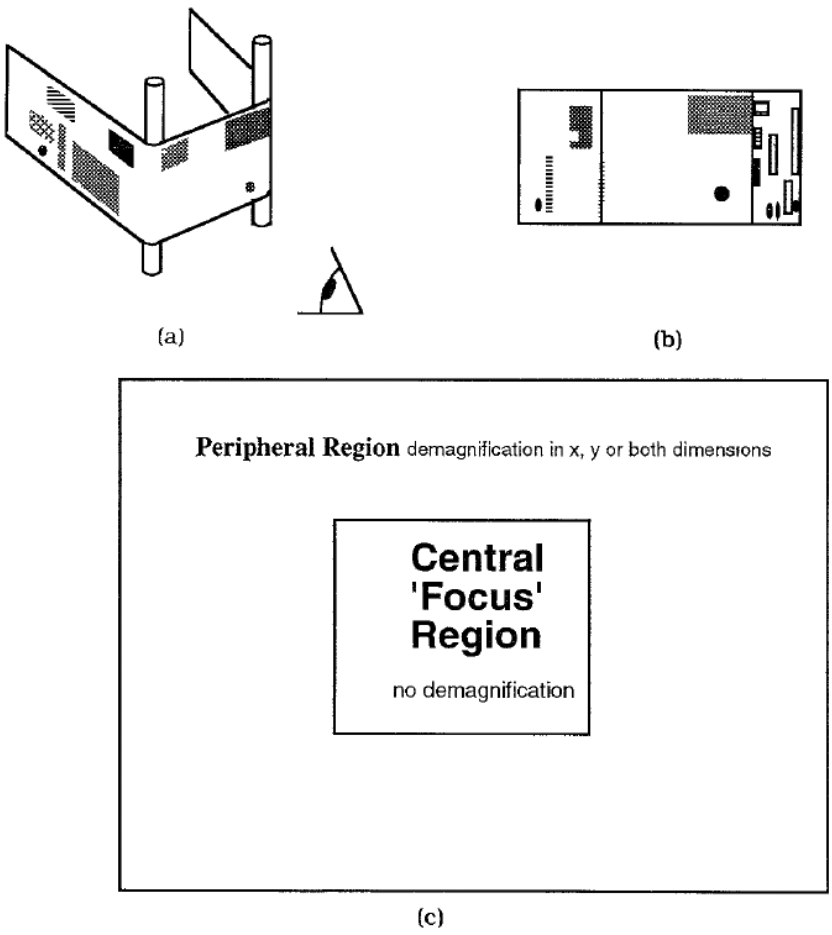


Figure 3.12: (a) A mechanical model of a distortion-oriented presentation technique. This model characterizes both the Perspective Wall and the Bifocal Display (b) the appearance of the data space transformed by a distortion-oriented presentation technique, in this case the Bifocal Display, obtained by viewing the model in (a) from infinity; (c) the presentation of a 2D distortion technique. [Leung and Apperley, 1994]

- techniques with continuous transformation functions (i.e. Polyfocal Display, Fisheye)
- techniques with non-continuous transformation functions (i.e. Bifocal Lens, Table Lens)

3.3.1 Fisheye View

The first one who came up with a Focus + Context method was Furnas with his work on *Fisheye Views* ("*A very wide angle, or fisheye, lens used at close distance shows things near the center of view in high magnification and detail. At the same time, however, it shows the whole structure – with decreasing magnification, less detail – as one gets further away from the center of view.*") [Furnas, 1981].

He defines two basic components of the *Fisheye View*: The first component is independent of the current interaction, consisting of parts of the structure which are of *a priori* global importance whereas the second one represents the contribution specific to the current focus of interaction (*a posteriori*). Based on this components he further defines the *DOI (degree of interest) metric*. This function controls the tradeoff between local detail and global context applying semantic scaling. It is resembled out of the three parts focal point, distance from focus and importance of nodes.

The *Fisheye View* is very well suited for hierarchical data structures, but problems arise if trees with an unequally distributed branch structure have to be displayed. Another characteristic of *Fisheye Views* is, that they are information suppressing techniques rather than distortion techniques [Leung and Apperley, 1994].

3.3.2 Bifocal Lens

A system based on Furnas' ideas that assists document retrieval was introduced by Spence and Apperley [Spence and Apperly, 1982]. They focused on developing a tool for locating a piece of information within a large database by facilitating the memory on where a particular document was located spatially as well as other clues like symbolic or textual labels. A good analogy for the problem characteristic is given by the authors: *"The difficulty has much in common with that of trying to pursue a newspaper through a 2-column-inch sized keyhole."* [Spence and Apperly, 1982].

The screen is divided vertically into three different viewports whereas the middle one shows the items of interest (focus) and the two outer ones contain context information. If an item is selected, it automatically moves to the center. Another issue is how to display documents in the context area. Spence and Apperley [Spence and Apperly, 1982] state that it is not sufficient to just display a demagnified version of its central region representation, but a representation more appropriate to the lower resolution has to be found.

A disadvantage of Bifocal Displays shown by Leung and Apperley [Leung and Apperley, 1994] is the discontinuity of magnification at the boundary between detailed and distorted view (see Fig. 3.13).

3.3.3 Table Lens

The *Table Lens* [Rao and Card, 1994] displays large data tables by distortion of spreadsheet's table cells (see Fig. 3.14). Rao and Card also present a general transformation function framework concept in their work. The transformation function used in the *Table Lens* is equivalent to the one used for the *Bifocal Lens* applied to two dimensions. Hence, the disadvantage of a discontinuous magnification is valid here as well.

Interaction and dynamic manipulation are emphasized in their work including actions like zoom, adjust, slide and the use of mouse gestures (i.e. "flicking").

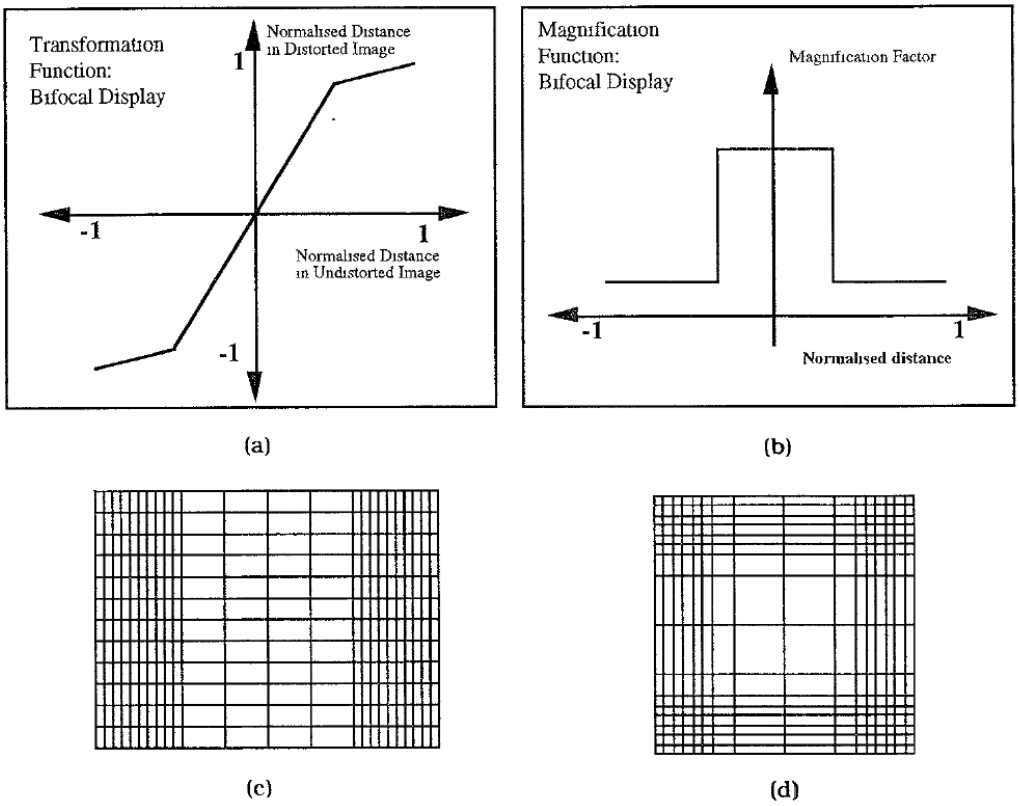


Figure 3.13: The Bifocal Display: (a) a typical transformation function; (b) the corresponding magnification function; (c) the application of the display in one dimension; (d) the application of the display in two dimensions. [Leung and Apperley, 1994]

3.3.4 Hierarchically Clustered Networks

Schaffer et al. [Schaffer et al., 1996] introduced a *Fisheye View* for hierarchically clustered networks, called *Variable Zoom*. They provide a method for zooming into clusters whereas zoomed content is magnified and the other nodes/clusters are shrunk (see Fig. 3.15). Zooming can also be applied to more than one cluster at a time (polyfocal). The network nodes have a fixed position in 2D space (x,y coordinates) and a calculation of position and size change is performed when zooming (geometrical problem).

Their work also includes a detailed description of the user study they conducted. This study shows that the *Fisheye View* is generally superior to a *Full Zoom View* but also concedes that the full zoom view has certain advantages in special situations like for example maintainence or looking for a specific characteristic within a single node.

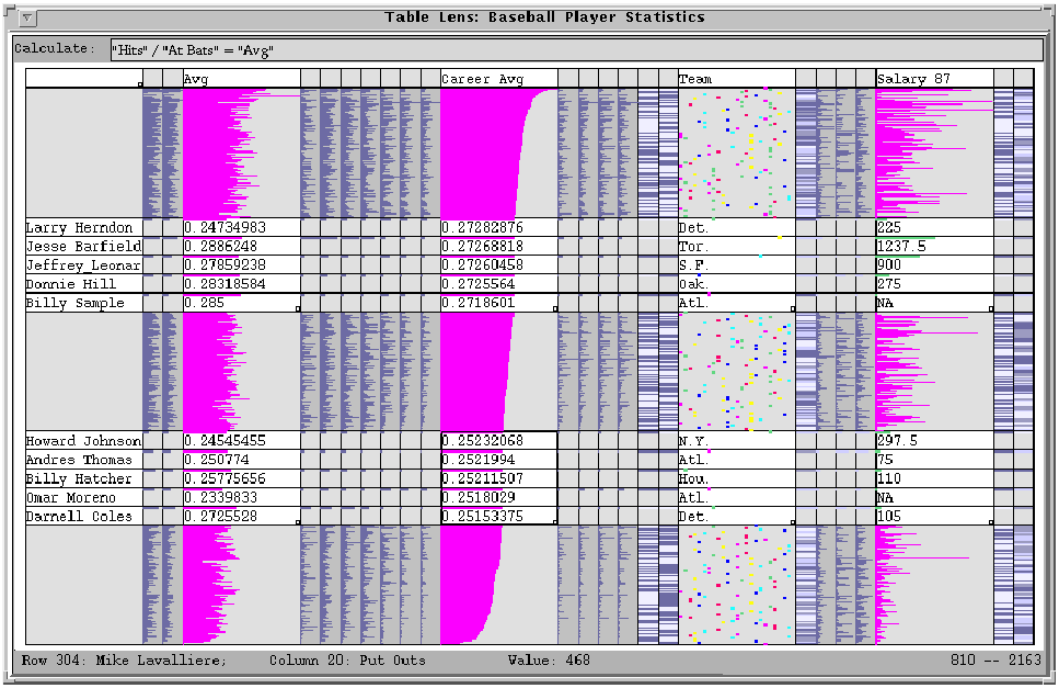
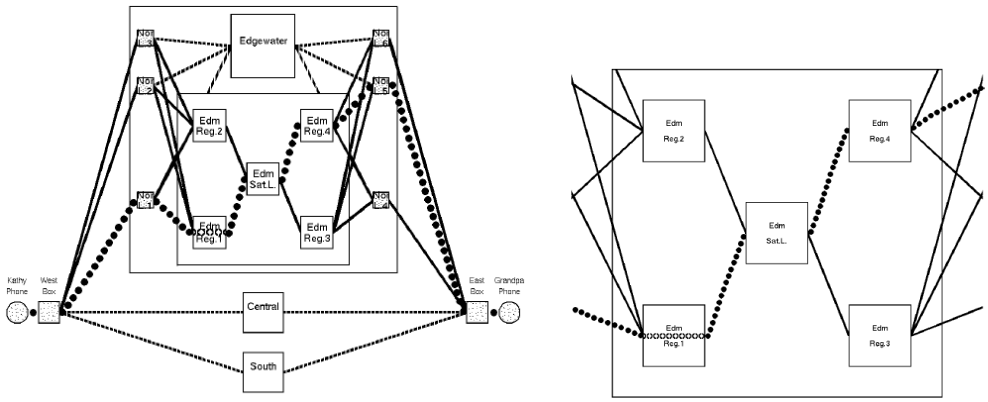


Figure 3.14: Table Lens [Rao and Card, 1994].



(a) (b)

Figure 3.15: Hierarchically clustered network. Fisheye (a), Full-Zoom (b) [Schaffer et al., 1996].

3.4 Medical Software Products

A very high portion of the offered commercial software products in medicine deal with administrative issues such as Patient Data Management or billing. Only very few include any visualization parts and even less offer a functionality for aiding treatment planning.

Patient Data Monitoring Systems (PDMS) face a variety of challenges including first of all the combination of low frequency and high frequency data, processing high dimensional data, and comparing these data.

Medical software products that use graphical representations and have sufficient product information available through their website have been examined.

3.4.1 IntelliVue / CareVue

A well known and wide spread application used mainly in intensive care units is *IntelliVue* [Philips Medical Systems, 2002] by *Philips Medical Systems* (formerly known as *CareVue* by *Hewlett-Packard*). *IntelliVue* is a highly customizable Patient Data Monitoring System (PDMS) that is used for a broad range of different medical units.

Its main presentational structure can be described as collection of spreadsheets with values in rows and time in columns. In terms of graphical representation, it offers mainly line diagrams depicting values over time as well as statistical diagrams like box-plots. More recent versions of *IntelliVue* have integrated a portal technology for displaying i.e. x-ray or other data.

3.4.2 Chart+ for Critical Care

Chart+ [Picis, 2002a] is a module of a comprehensive software package for critical care units offered by *Picis*. This module includes a charting application as well as a data collection engine dealing with vital patient information.

The largest part of the screen is dedicated to a tabular representation of collected values. The graphical representation at the bottom of the screen consists of line diagrams, visualizing certain selected parameters and values.

3.4.3 Visual Care

Another software module offered by *Picis* is *Visual Care* [Picis, 2002b] which mainly focuses on patient care and planning. It offers management of care protocols including also visual representations. This visualization is done by so-called "Flow-sheets" which are GANTT-like diagrams on spreadsheets additionally including icons that are used to represent planned incidents like for example certain examinations.

3.4.4 QCare

The *Critical Care Company* offers a palette of applications for critical care including *QCare* [Critical Care Company, 2002]. Many different kinds of graphical representations are used such as a GANTT-like representation for nursing planning, various forms of line diagrams, or an annotation system integrating hand drawn sketches.

Of particular interest is the nursing planning module. The visualization part uses a GANTT-like representation including LifeLines for depicting incidents like the duration of a medication. Furthermore, it uses icons depicting different types of medicaments and annotating text within TimeLines.

3.4.5 Others

Other medical software products that have been examined like *Coronary Risk Profile (CRP)* (Wellsource) [Wellsource, 2002], *SOAPware* (Docs, Inc.) [Docs Inc., 2002] and *Clicks Medical Information System* [Roshtov Software Ind. Ltd., 2002] (Roshtov Software Ind. Ltd.) only include classical diagram types as line diagrams and bar charts.

3.4.6 Summary

The most popular form of data representation among the examined products is using tables where numerical respectively textual data is organized in spreadsheets. When displaying time-oriented data, measured values respectively parameters are placed in rows and the points of measurement in columns.

The *Flow Sheets* used in some applications are a combination of spreadsheet and LifeLine representation whereas the displayed temporal granularity is determined by the width of table cells. Table cells are either filled (as part of a LifeLine), contain symbols, textual data, or a combination of these items.

Most of the table based applications allow depicting the data as diagrams whereas the use of line diagrams is dominant. Some of the products provide the possibility to include image based data such as x-ray images or ultrasonic images.

3.5 Medical Treatment Planning

3.5.1 AsbruView

AsbruView [Kosara and Miksch, 2001a, Miksch and Kosara, 1999, Kosara et al., 2001, Kosara and Miksch, 2001b] is a graphical tool that supports creation and manipulation of time-oriented, skeletal plans, clinical protocols and guidelines can be seen as. It is part of the *Asgaard* project, thus based on the plan representation language *Asbru*.

AsbruView utilizes metaphors of running tracks and traffic control to communicate important concepts and uses glyphs to depict the complex time annotations used in *Asbru*. The interface consists basically of two major parts, respectively views: One captures the topology of plans, whereas the second one shows the temporal dimension of plans [Kosara, 1999] (see Section 2.3 for further details).

The hierarchical decomposition of plans can be explored via expandable and collapsable blocks. The basic structure of the temporal view has some similarities with structograms: It uses building blocks whereas the plan type (i.e. parallel, sequential) is indicated by simple symbols.

As stated before, the complex time annotation is depicted by a novel glyph (see Fig. 3.17). All attributes of the time annotation as well as not defined values and

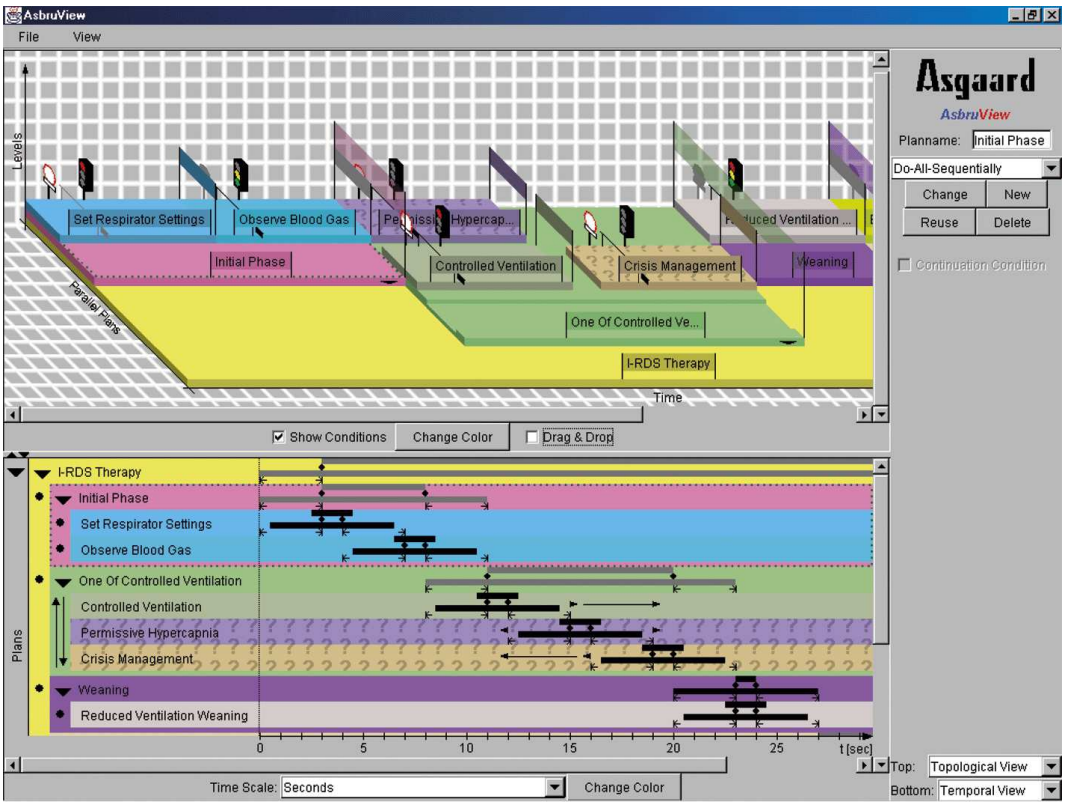


Figure 3.16: AsbruView [Kosara and Miksch, 2001a]

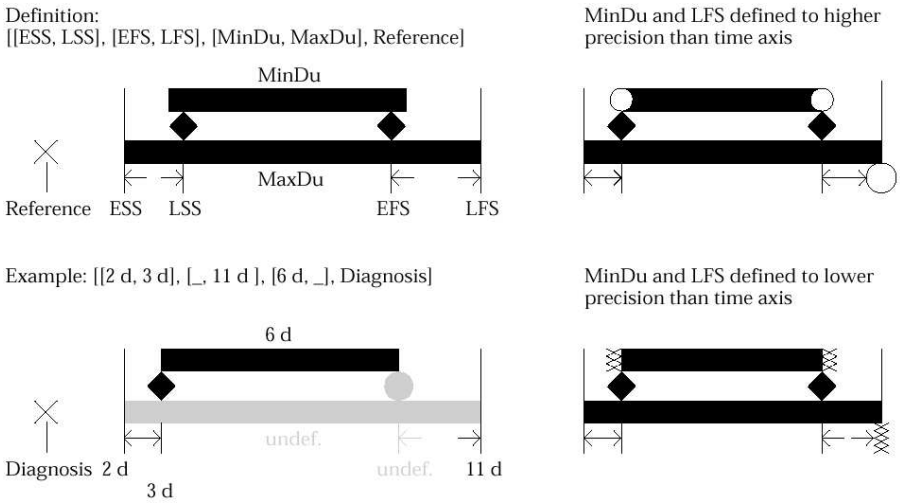


Figure 3.17: Time glyph of AsbruView [Kosara and Miksch, 2001a]

different time scale granularities can be shown and manipulated in a natural and easily understandable way, which has been proved by user tests [Kosara and Miksch,

2001b].

The topological view uses 3D graphics to depict a running track and various traffic control objects. This 3D view helps understanding the metaphor more easily, but also introduces some problems: The first problem is occlusion - if several plans are expanded, some parts of other plans are hidden. The ability of disabling some objects improves the situation but is not a thorough solution for this problem. Viewing more expanded plans at once, which means that lots of items are displayed, introduces the problem of a cluttered interface and leads to a decrease or even loss of overview. Another shortcoming, which has probably the biggest overall impact, is runtime performance. Due to the used 3D view and the implementation in Java, rendering graphics uses a significant portion of processing power thus decreasing the overall performance drastically.

3.5.2 Guideline Overview Tool (GOT)

The *Guideline Overview Tool* [Aigner, 2001] has been designed as part of the *Asgaard project*. Its purpose is to provide a user interface for comparing and analyzing treatment plans as well as patient data of several patients on a single screen. LifeLines with several extensions are used to display treatment plans. Extensions include special bars and signs for different plan states and events (i.e. suspend, reactivate, abort) as included in *Asbru* as well as the possibility to collapse LifeLines which provides a very compact view for plans.

Uncertainty or undefined values can not be depicted using this prototype. A way for interactively exploring the plan hierarchy is also not supported.

Two forms of diagrams are used for displaying patient data (see Fig. 3.18): LifeLines are used for depicting qualitative values and line diagrams for quantitative values. Both diagram types can also be presented in a compacted form to minimize vertical space consumption without the need of disabling the display of a value completely.

3.5.3 Midgaard

A parallel work to this thesis is carried out by B. Ragnar [Bade, 2002], where LifeLines are used to display plans in time as well. The LifeLine extension also includes the possibility to depict uncertainty and undefined values in the time annotation of plans ("L-notation", see Fig. 3.19).

A problem of using the "L-notation" for displaying uncertainty is that if many bars are displayed, the screen gets overloaded and cluttered quickly by a lot of thin vertical lines.

3.5.4 Related Work: Protocol Based Care

A comprehensive overview of related protocol-based care projects can be found at [Duftschmid, 1999] and [www.openclinical.org, 2003]. Some of the projects have graphical tools which are mainly used for plan creation:

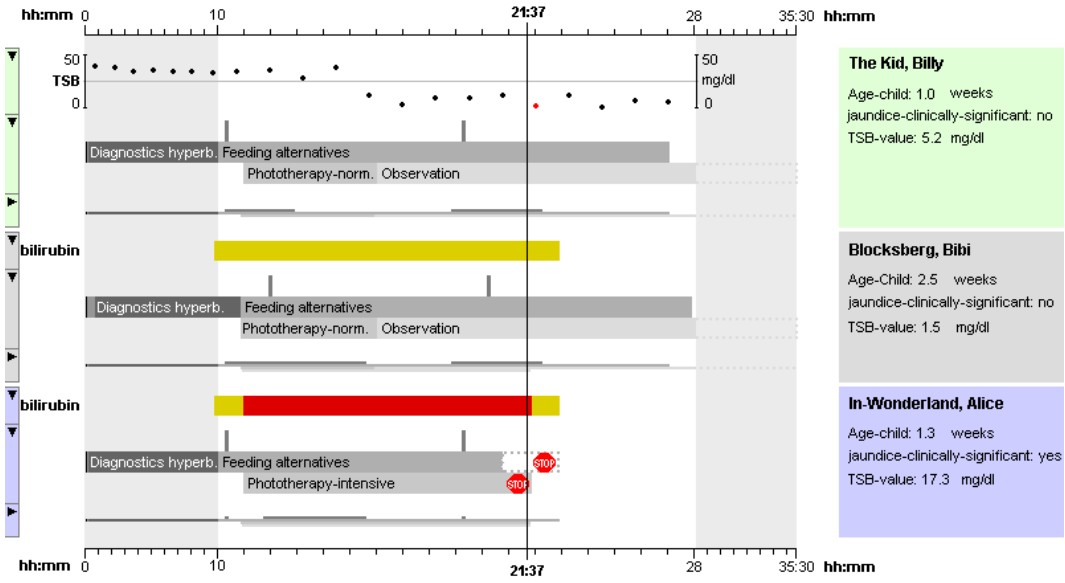


Figure 3.18: Guideline Overview Tool [Aigner, 2001].

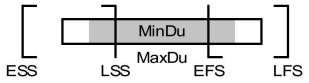


Figure 3.19: L-notation for depicting plans over time [Bade, 2002].

GLARE

The *GLARE* [Guarnero et al., 1998] *knowledge acquisition tool* (see Fig. 3.20) is an "intelligent" guideline acquisition interface for authoring guidelines. Beneath the possibility to create plans graphically, it includes modules to verify the well-formedness of a guideline via syntactic and semantic tests as well as consistency checking of temporal constraints.

GUIDE

GUIDE [Quaglini et al., 2001] is a workflow-based approach using *Petri nets* that have been extended. For creating these workflows, the graphical tool *Oracle Workflow*^{TM} is used (see Fig. 3.21).

Protégé

Protégé [Shankar et al., 2002] provides a set of customizable knowledge based tools. These tools are used by a variety of other projects like *PROforma*, *PRODIGY*, or *GLIF*.

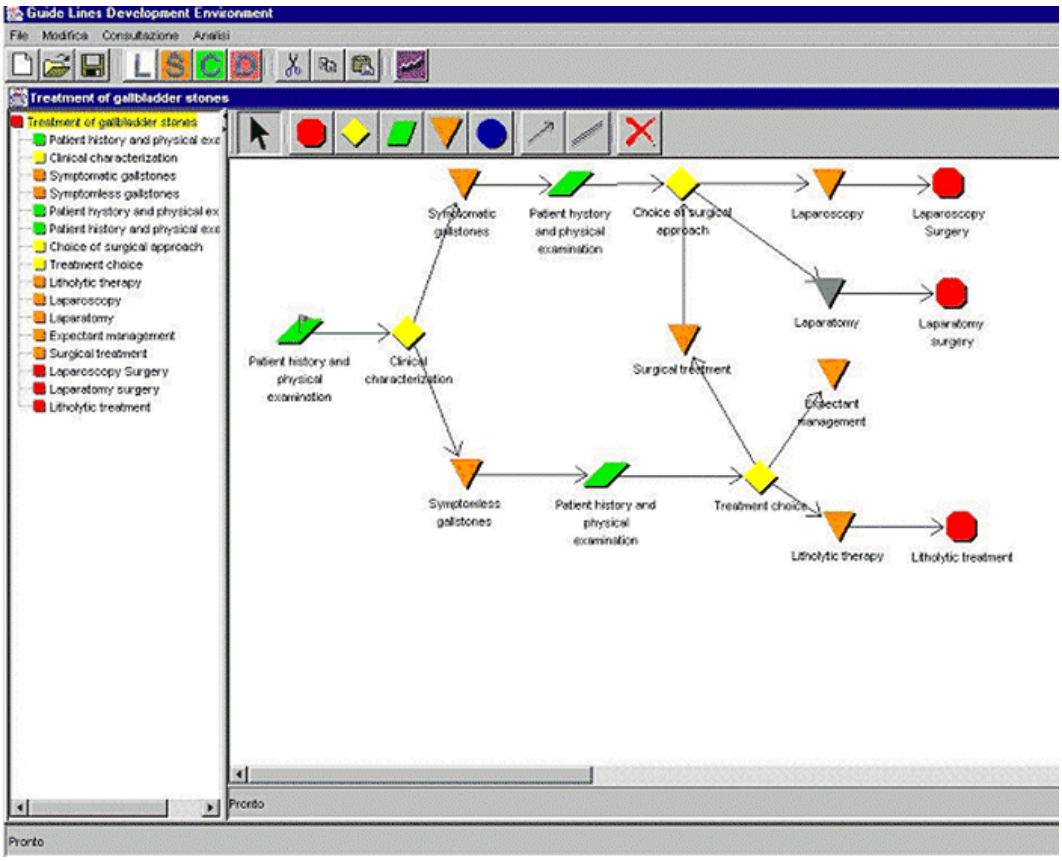


Figure 3.20: *GLARE knowledge acquisition tool* [www.openclinical.org, 2003].

GLIF *GLIF* [Peleg et al., 2000] supports guideline modeling as flowchart of structured steps that represent clinical actions and decisions. Guidelines can be viewed as state transition graphs using a flowchart representation (see Fig. 3.22) of a temporal sequence of clinical steps (*Action Step, Decision Step, Branch Step, Synchronization Step, Patient State Step*). The authoring tool supports plan creation at a conceptual level and also provides nesting of elements.

PROforma *PROforma* [Fox and Thomson, 1998] also uses a flowchart-like authoring tool to depict the four possible *PROforma* tasks *plan*, *decision*, *action*, and *enquiry* (see Fig. 3.23 and Fig. 3.24).

Summary

Only some of the available projects dealing with protocol-based care provide graphical tools at all. The just listed ones include such graphical tools, but most of them only for authoring plans. They use a flowchart- or workflow-like presentation depicting the elements used in their formal representation.

These tools make plan creation easier especially for non computer scientists but they use a not very familiar graphical representation and mix state and flowchart

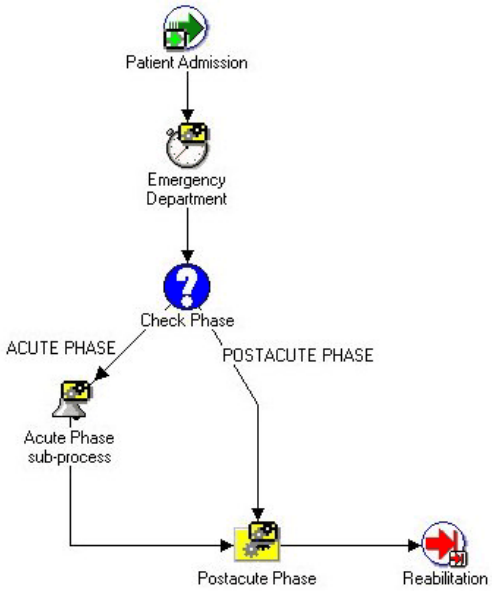


Figure 3.21: *GUIDE* using *Oracle Workflow*^{TM} [www.openclinical.org, 2003].

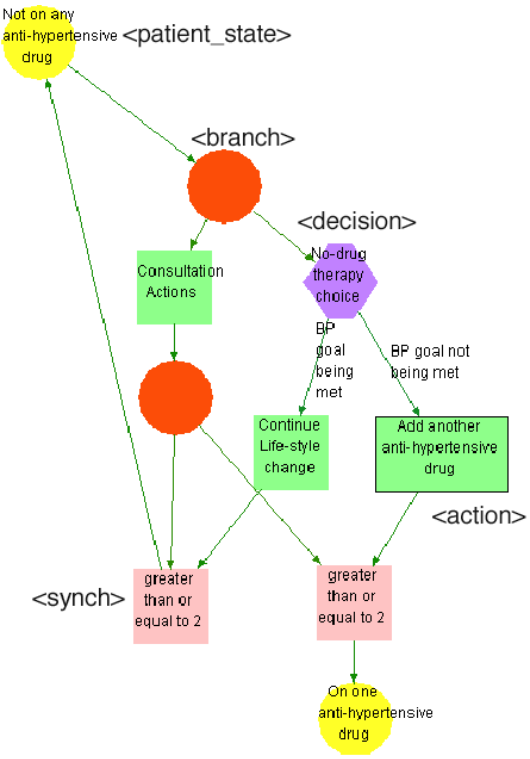


Figure 3.22: *GLIF* authoring with *Protégé* [www.openclinical.org, 2003].

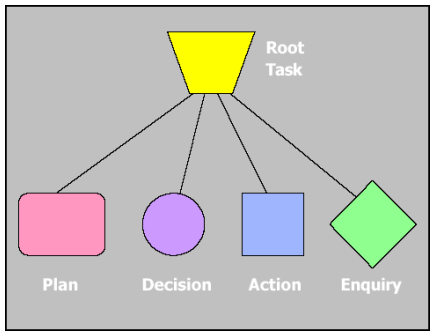


Figure 3.23: *PROforma* tasks [www.openclinical.org, 2003].

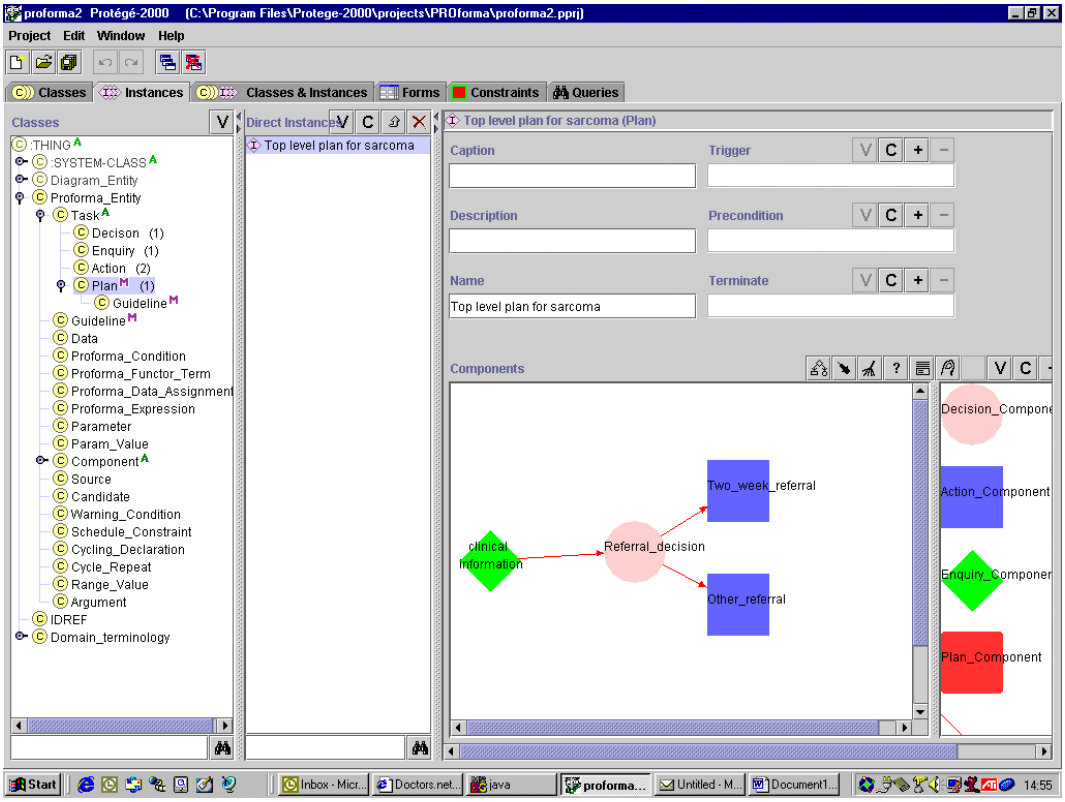


Figure 3.24: *PROforma* authoring with *Protégé* [www.openclinical.org, 2003].

characteristics within a single diagram. Thus, understanding this representation and using it for plan authoring requires a considerable amount of learning effort. Furthermore, none of the presented tools included a notion for depicting time graphically. They work on a conceptual respectively logical level containing time just implicitly.

Moreover, these tools support plan authoring only, but none of the projects includes visualizations at runtime. Often, this is left out for commercial products using a certain representation, but we were not able to find a single product with integrated runtime plan visualization.

3.6 Discussion

Important Information Visualization techniques and approaches as well as User Interface Design and Human Computer Interaction (HCI) issues related to our particular problem and medical software development in general have been presented in this chapter. Furthermore, various commercial software products that use graphical representations or diagraming techniques were examined.

The presented Information Visualization techniques are elegant solutions to familiar problems in this field. But all of them are only related partially or can only be applied to parts of our problem characteristic. None of them provides a notion including all the visualization characteristics needed for depicting *Asbru* plans (see Chapter 2):

- Time-oriented data from the past to future planning data including a rich set of time attributes allowing to depict uncertainties (execution time attributes and time annotations),
- Hierarchical data (subplans),
- Elements having execution sequence relationships (sequentially, parallel, any-order, unordered),
- Non-uniform element types (plans, if-then-else, ask, cyclical plans, variable assignments),
- Conditions having state characteristics (abort-condition, complete-condition), and
- Focus + Context display and interaction.

	Time-oriented data	Hierarchical data	Non-uniform element types	Logical sequence data	Concurrency	Conditions having state characteristics	Focus + Context functionality
Trees		●	●				○
Treemaps		●					
LifeLines	●			○	●		
Temporal Objects	●			○	●		
Paint Strips	●			○	●		
GANTT charts	●	●		○	●		○
SOPOs	●				●		
PERT charts				●	●		
Flow charts		○	●	●			
Structogram		○	●	●	●		○
AsbruView	●	●	●	●	●	●	○
Guideline Overview Tool	●	●		●	●		○
Midgaard	●	●		●	●		●

● ...completely fulfilled, ○ ...partly or implicitly fulfilled

Table 3.1: Summary of investigated visualization methods and tools.

Chapter 4

User Study

4.1 Objectives, Method

Due to often fundamental differences in knowledge background, requirements and needs among different user groups it is next to impossible to develop a tool that fits all. In our special case it is not possible to design software optimal for physicians, nursing personnel, and other medical staff at once because of their different needs and objectives. We believe that designing software for a clearly defined group of users leads to better results in both structure and interaction than trying to fulfill requirements for everyone. Different views would be a solution for enabling a software product being well suited for different user groups. In this case the individual views would also be designed independently to achieve best possible results. Due to these reasons we decided to develop our software for one clearly defined group of users: **physicians**. Because of other prerequisites like available data sets and treatment plans we restricted this group further to physicians working in hospitals at intensive care units dealing with critically ill patients.

One of the first steps for requirement analysis in our development was conducting a user study [Nielsen, 1993]. The objectives for carrying this study were:

- gaining more domain knowledge
- investigating work practices and flows
- getting information about user experience on working with computers and certain software products
- determining user requirements

In order to obtain these goals we decided to carry out a qualitative survey in form of a *semi-structured interview* [Preece et al., 2002]. Hereby, user interviews lead by an interview guideline were conducted. We chose this type of user study for various reasons: First of all it is important to keep in mind that the beginning of development is characterized by many unclear issues, uncertainties and only very vague concepts. Additionally, the user study was mainly focused onto the "WHAT" questions rather than "HOW" issues [1].

[1] "WHAT" questions are basically more general, emphasizing WHAT is important, WHAT should

An alternative to the semi-structured interview would be carrying out a questionnaire that is filled out by users. But in the light of the above stated prerequisites this is not the best choice for gaining as much information as possible. The questions would have to be more detailed and limited in order to get answers because it is not very likely that someone would write lengthy text answering a rather general question i.e. on how a certain workflow is performed. Furthermore, the freedom for the interviewed person is much higher in case of an interview situation. In contrast to the strict structure of a questionnaire she can get into more details at certain points or talk about issues not included in the guideline as well. This means that the level of flexibility is very high in a semi-structured interview in terms of the opportunity to shape it onto the special situation of every interview partner.

The goal for this user study besides gaining more knowledge about the domain, user requirements, and desires, was the creation of a user model containing representative user profiles, their goals, and the construction of interaction scenarios based on this user model.

This method is based on Coopers *Interaction Design* technique [Cooper, 1999] also known as *Personas*, *Goals* and *Scenarios*. Whereas *Personas* are a created cast of characters representing real persons along with both their knowledge in the computer and the application domain. These persons have certain *Goals* they want to achieve when using a product. A *Scenario* is basically a detailed story about a person performing a certain task to achieve her goals. Cooper also shows that designing for only a few main characters and their goals helps to enhance the usability of a product and avoids feature driven products.

4.2 Interview Guideline

The guideline used for conducting the user study was structured as follows (see Appendix B for the complete guideline):

Firstly, some general comments on the interview situation followed by a short presentation about the *Asgaard* project in general (organization, goals, parts, phases) and the role of this work within the project were given. After that, knowledge and experiences in working with computers, the medical domain and medical planning tools were examined.

In the computer related part, the interviewed physicians had to judge their own computer knowledge and were asked which tasks they carry out using computers, what software products and operating systems they use and how often they work with computers, both in their job as well as in private. Furthermore, they were asked if they have experiences with project management techniques or project management software and known diagraming and visualization techniques were investigated by showing a set of example diagrams to the interviewees (see Appendix B.3).

Examining medical domain issues contained questions about the position and responsibilities, prior career steps, and experiences of the physician. Beyond that, the handling of treatment plans and medical guidelines was explored in detail. This

be visualized, WHAT visualization techniques should be used rather than "HOW" questions that focus on details as HOW certain parts should be implemented or HOW desired features can be realized. Generally spoken, "WHAT" questions exclude implementation issues.

included questions on "if", "how", and "what for" treatment plans respectively medical guidelines are used and how they are represented. Additionally, the workflow starting from diagnosis to choosing a certain therapy and carrying out the therapy was discussed. If there is data and certain measurements that are important and needed independently of the present case was tried to filter out as well.

Afterwards the basic opinion towards computer aided execution of guidelines and therapy plans was asked as well as experiences and contentment in case the physician ever worked with similar systems before.

A goal and task analysis was conducted in the next section of the interview guideline. This section included rather general questions as whether such a system would be used at all if one would have the opportunity to, which tasks one would like to perform with such a system, and how one imagines daily work with the system. More detailed questions were asked about desired important characteristics of the system and what has to be visualized necessarily. Further on, issues on the design and interaction with such a system were studied: If she has a concrete picture of the tool in her mind or if she could imagine that it works similar to a known software product. If she imagines working with the system on a laptop, at the bedside, or at a workstation and how she would like to interact with the system (via i.e. mouse or stylus). What visualization techniques one could imagine being used for representing plans was another very important question for future design steps of the software.

The last, rather relaxing section tried to find out more about the desired, more general characteristic of the system. It contained psychological questions about system analogies in the sense of: If the system would be a color/car/house, what color/car/house would it be?

4.3 Results

Eight physicians of the General Hospital of Vienna (AKH Wien) agreed on participating in this user study. Most of them work at departments for critically ill patients. The General Hospital of Vienna (AKH Wien) is a university clinic which means that employed physicians also work scientifically. Conducting an interview took on average about 45 minutes and lead to interesting, but not too surprising results and insights.

Fundamental issues for the interviewed physicians were rather practical ones. Most importantly the system has to save time – no one would use a system if it would take more time as working without it. Furthermore, as much input data as possible should be integrated automatically, minimizing the need for manual data entry. Another major issue is that learning effort for using the system has to be minimal. The system should be intuitive, simple and clearly structured without complex menu structures or functions.

Computer Skills and Used Software. Most of the interviewed physicians judged themselves in the mid region of a scale from 1 (novice) to 5 (expert). Only two of them declared themselves as experts having substantial knowledge even in software development.

Computers are mostly used for word processing, literature research and communication by the interviewed doctors. Significantly less also work with statistics packages, image processing software or self developed tools for scientific projects. All interviewed persons stated that they use Microsoft™ Standard Software (Office) for most of their work. In terms of operating systems, Windows™ is used by everyone and half of the interviewed persons knew the Macintosh™ system as well.

Only two physicians knew project management techniques or have already used project management software.

Known Visualizations. All physicians were familiar with standard diagram types (Line diagrams, Bar charts, Pie charts) and Flow charts. That flow charts are known by all is comprehensible to the fact that the widely known *clinical algorithms* are generally depicted by a special form of Flow charts as proposed in [Society for Medical Decision Making, 1992]. GANTT charts were also relatively well known among our interview partners. Half of the interviewed physicians knew LifeLines and PERT charts whereas LifeLines were understood much more easily when asking about the possible meaning of an example in contrast to PERT charts, where this was not the case. Relatively unknown were Structograms, and Glyphs as for example Chernhoff Faces.

Plan Visualization. When asked for the preferred form of visualization for plans, LifeLines turned out to be most popular, followed by Flow charts and GANTT charts. Important to note is, that no other form of visualization was mentioned.

Experiences with Used Medical Software. This section examined experiences with medical software whereas most comments were given having *CareVue* [Philips Medical Systems, 2002] in mind. It has been complained about the used software that it is often troublesome to work with, complex, inflexible, slow and that one has to deal with many completely different systems in terms of structure, interaction, and design. Positive comments included mainly the advantage of a quickly ascertainable overview and a good documentation in contrast to thick and complicated paper based records, that visualizations are included, and data is imported automatically.

Hardware and Interaction. As much as half of the interviewed physicians imagine having one workstation at the department for working with the software, whereas only a fourth desire having TabletPCs and another fourth would like to have it included into bedside monitor systems. The mouse is the most wanted input device closely followed by a stylus as used for handheld PCs.

Characteristic. The most important characteristics for the potential users of the system are simple and intuitive interaction, a simple menu structure, a clear system that is fast, and very crucial – time saving. Other specified characteristics included: interactivity, not overloaded display, no use of 3D visualizations, freely configurable, uniform, many links to external sources (hyperlinks), related to practice, valid information and up to date. The refusal of 3D visualization was argued by getting overloaded and overwhelmed.

The very interesting system analogy questions drew a diverse picture. The most often stated colors that represent the system characteristic were blue and grey which leads to a rather cold, clear, and somehow rejecting impression. The car comparison brought a quite uniform result among the interview contestants: Upper class cars with a high level of safety, a certain amount of luxury and speed like BMW, Audi, or VW were named. When comparing the system characteristics with houses, a two fold result was brought up: One part of physicians stated one or more family houses whereas the other part imagined modern buildings having a modern, ecological, and transparent architecture. The first impression leads to a convenient, familiarly characteristic where one feels well and which invites to stay. In contrast to that, the latter impression leads to a clearly structured, modern characteristic without ballast and hidden features enabling quick and effective work.

When summarizing and evaluating all issues the following fundamental system characteristics can be recognized:

- simple and transparent structure
- intuitive interaction (easy to learn and comprehend)
- cleaned up interface
- high level of application safety (undo where possible)
- time saving (allowing quick and effective work)
- fast
- flexible

4.4 User Model

Analyzing the interview results and categorizing users led to three different characteristic user groups. Based on these user groups three fictious personalities (*personas* [Cooper, 1999]) were resembled uniting the main characteristics of each group:

Name:	Markus Zolte
Age:	29
Position:	Assistant doctor in training (internal medicine)
Computer knowledge:	good user knowledge; mainly word processing, documentation, literature research and communication; uses mainly Microsoft^{TM} standard software products, and works with *IntelliVue*
Visualization knowledge:	is familiar with standard diagrams for data visualization and flow charts
Medicine:	has been assistant doctor for 2 years, uses standardized workflows for diagnosis, uses guidelines besides that rarely
Treatment planning tool:	did not hear of it yet; thinks that it is a good idea, but not applicable universally; expects above all a reduction of the huge amount of paper used now
System use:	wants to use the system for therapy conduction; wants to explore the state of the art of treatment algorithms through the system
Desires:	intuitive and simple, visually appealing, related to practice, fast, interactive, integration of diagnostic findings, validity of information must be assured, quick comprehensibility

Name:	Andrea Habacher
Age:	38
Position:	Assistant medical director (internal medicine)
Computer knowledge:	good user knowledge; mainly word processing, documentation, scientific work; uses miscellaneous software tools at her department; uses mainly Microsoft™ standard software products and statistics packages; works with Windows™, rarely with Macintosh™ systems
Visualization knowledge:	is familiar with standard diagrams for data visualization, Flow charts, GANTT charts and LifeLines
Medicine:	works in the field of inner medicine since 10 years; works with guidelines and therapy plans. (not directly, but they are internalized)
Treatment planning tool:	knows such tools (department internal software); thinks that such systems make sense if several criteria are fulfilled
System use:	wants to use the system for documentation, therapy planning and analysis
Desires:	wants a simple, clear system, no 3D, uniform for different departments, integrated search system, up to date, modification possibility, many links to external sources (hyperlinks), integration of miscellaneous data sources

Name:	Heinrich Kovanic
Age:	43
Position:	Assistant medical director (intensive care unit)
Computer knowledge:	very good computer knowledge; mainly Microsoft™ Office applications, statistical analysis, image processing, scientific work, literature research and communication; uses mainly Microsoft™ standard software products and statistics packages, works with *IntelliVue*, knows Macintosh™ systems and project management techniques as well as software tools for project management
Visualization knowledge:	is familiar with standard diagrams for data and flow visualization (flow charts, structograms, PERT charts, GANTT charts and LifeLines) and various statistical diagrams
Medicine:	has many years of experience in intensive care, uses guidelines and therapy plans; workflows/plans are internalized, does not use aids (guidelines on paper)
Treatment planning tool:	thinks that computer support is good and makes sense; uses internal, self programmed software tools; is not satisfied with *IntelliVue*
System use:	wants to use the complete workflow (planning, data, documentation, analysis, medication, ...)
Desires:	wants a fast, clear system, no complex graphical elements, simple interaction, freely configurable, no complex menu structures; the system should have an open architecture for integration of other data sources

4.5 Scenarios

4.5.1 Scenario 1

Markus Zolte is going to work in the pediatrics department for the next few months and is exploring various treatment methods for new born infants. He informs himself about hyperbilirubinemia by walking through the treatment protocol. He is interested in the logical workflow and explores the flow chart representation of the treatment plan.

After the first walkthrough of the hyperbilirubinemia protocol, Markus Zolte goes back to the intensive phototherapy part and wants to know in which cases this plan is aborting. He is also interested which part of the complete treatment plan he is viewing right now.

Furthermore, he wants to see all other parameters and variables that are getting used in this treatment plan.

4.5.2 Scenario 2

Andrea Habacher just completed the treatment of a patient using the controlled ventilation plan. Now she wants to analyze different parts of the treatment along with measured patient data. She starts in the temporal view part by examining how long different phases of the plan took in relation to others. Therefore, she disables the Fisheye view.

The "handle PCO2 plan" is of particular interest for her. Thus she zooms onto that plan and uses Fisheye view for displaying context information. She also wants to display the PCO2 value for examining relations between plan execution and PCO2 values.

Because there is a significant discontinuity of the PCO2 value within this plan, she recalls the sub-steps taken in the "handle PCO2 plan" by displaying it within the corresponding Logical View. Furthermore, she expands the plan in the Temporal View to see when the particular steps were conducted.

After that she is interested in if and how the PCO2 values influenced the "patient-state" parameter. Therefore, she displays this parameter and collapses the temporal plan display for getting more vertical space.

4.5.3 Scenario 3

Heinrich Kovanic is currently treating a patient suffering from hyperbilirubinemia. He displays the "TSB" and "TSB-change" values additionally in the value observation panel.

Now he wants to review the patient record for getting basic patient information. After that he investigates all incoming parameters and collapses the time oriented plan view. He encounters a rapid increase of the TSB value two hours ago and wants to find out which plan or action took place at that time. Therefore, he expands the temporal plan view and adjusts the time scale to a portion +/- 5 hours of the point of interest. He navigates down the hierarchy of plans and identifies the related one.

Furthermore, he examines the parameter constraints defined by the conditions presented in the Logical View. After encountering the reason for the value change, he wants to go back to the actual position of plan execution.

4.6 The Ambivalent Relationship of Medicine and Technology

Carrying out a user study is probably never an easy task and especially the medical sector has a somehow ambivalent relation towards technology. We decided to include this section into this thesis because we encountered several serious problems at the beginning of our study. We also included literature research when digging for the reasons of this difficulties.

One of the first steps when conducting a user study is to look for interview partners that match the defined user group. As easy it might look like, it turned out to be the hardest part of all. At the beginning we were in good faith that physicians would be interested very much in participating in our study because the developed tool tries to ease their work and brings significant improvements. But simply spoken, this imagination was false. Many physicians we contacted did not seem to be interested or were even reluctant to participate. It seemed as they interpreted the short introduction to our project in a wrong way, came up with many technical problems, and did not believe that it would work at all. Only after longer conversations and defending our ideas we were able to eliminate these false imaginations.

Faced with this problems we took a deeper look at why things are how they are. We began with examining the short letter that introduced our project and asked the particular physician for her participation in the user study. This examination led to the conclusion that our letter was too much focused on the scientific issues, did not state why we wanted the particular doctor to participate, and most importantly, what benefits the physician encounters when participating. Metaphorically speaking we placed ourselves in this letter as opponent to the physician. We were placed on the one side along with our technology and the physician was placed on the other side of the "line". This led to the situation that we were representing the technology that was saturated by many prejudices and had to defend it without getting any information and sympathy from the user. It would have been better to place ourselves at a position along with the physician both exploring and evaluating the technology.

Another document we took a closer look at was the short presentation of the *Asgaard* project. The original version was emphasizing the scientific aspects as well as the structure and parts of the project. We decided to improve this presentation in a way that it emphasized the physician's situation and pointed out the advantages and opportunities more clearly. Hence, we shifted the bias from a scientific form of presentation to a rather product- and user-centered one.

Along with this improvements we were exploring the reasons for the rejecting attitude of some physicians. One of our impressions was that the technical issues and problems stated by some doctors are not the only reasons for their refusal. We

considered possible organizational, personal and social aspects:

- anxiety of automation
- not willing to spend time
- not interested in computers
- need for more precise documentation that would list all taken steps minutely and leads to precise retracing possibility
- loss of individuality by standardization

Literature research brought up some interesting papers that validated our observations and presented some more facts other scientists encountered when dealing with technology in medicine [Gershon et al., 1994,Smith, 1996,Gorman and Helfand, 1995].

Interestingly, Hans-Peter Meinzer reported in the panel discussion "Is Visualization REALLY Necessary" [Gershon et al., 1994] that *"Radiologists have been quite hesitant to accept the new 3-D volume visualization, claiming that "We don't need it." "*.

Richard Smith examined various user studies in the medical field in his work "What clinical information do doctors need?" [Smith, 1996] that listed only 13 studies about the information need of physicians that have been carried out until that. Most of them had low participation and response rates and all but one were American. Furthermore, he points out that doctors in developed countries seem to be overwhelmed by the information provided to them and the amount of information is enormous and disorganized. But *"[...] computer systems that have been developed are not widely used – perhaps because they have not been developed to meet doctors' information needs"* [Smith, 1996].

It is important to state that medicine does basically not work like other natural sciences. Strict rules or clear cause-effect chains like for example in physics or chemistry can only be found very rarely in medicine. This also applies to work methods and treatment approaches which are no strict step-by-step instructions. There is no well defined methodology like a comprehensive analysis of the situation at the beginning based on which a treatment method can be chosen conclusively that results in a defined effect. *"Yet most of the information doctors use when seeing patients they keep in their heads in what has been called "a constantly expanding and reinterpreted database."* [Smith, 1996]

Keeping these issues in mind is very important when dealing with physicians. A lot of the refusing attitude doctors evince towards technology and technicians is found on this misunderstandings. Technicians tend to build systems that work analogous to how they are used to work themselves: clear foundations with strictly defined basics and methods. Technicians are looking from a different point of view at medicine than doctors do. They try to understand medicine in a technological way applying their own mental models and categories which leads to frictions between medicine and technology. Physicians feel misunderstood, one of the interviewed doctors stated: *"What makes me angry is that technicians come, look at some parts,*

and think that they understand how medicine works."[2] On the other hand, it is also true that physicians do often have wrong impressions and prejudices towards information technology blaming technicians without any further consideration.

And last but not least, working conditions that doctors have to face, have to be considered. Most physicians are stressed, have very little time and have to make a lot of decisions. (See [Gosbee and Ritchie, 1997] and their example *"A day in the Life of an Internal Medicine Physician"*.)

Recapitulating, the conclusion can be drawn that too less effort has been dedicated in conducting user studies and trying to capture systems from a physician's rather than a technological point of view in the past.

"New medical information systems, no matter how fast, inexpensive, and easy to use, will not be used more widely until it has been demonstrated to practitioners that these systems provide answers that help solve the problems of patient care." [Gorman and Helfand, 1995].

[2]Original German statement: "Was mich ärgert ist, dass Techniker kommen, sich ein paar Teile ansehen und dann glauben zu verstehen wie die Medizin funktioniert."

Chapter 5

Conceptual Design

5.1 General

In Chapter 2 we abstracted the following visualization relevant data characteristics from *Asbru*:

- Time-oriented data from the past to future planning data including a rich set of time attributes allowing to depict uncertainties (execution time attributes and time annotations)
- Hierarchical data (subplans)
- Elements having execution sequence relationships (sequentially, parallel, any-order, unordered)
- Non-uniform element types (plans, if-then-else, ask, cyclical plans, variable assignments)
- Conditions having state characteristics (abort-condition, complete-condition)

As the *State of the Art Report* (Chapter 3) showed, neither one of the existing visualization methods is able to capture those characteristics, nor one of the examined medical software products (Section 3.4) included a visualization technique that could be used here. Based on the investigated visualization techniques and the results of the user study we conducted (Chapter 4), we worked on the creation of a new form of visualization supporting all the characteristics we need.

That a single form of visualization supporting all characteristics would be too complicated, overloaded, hard to understand, and probably not usable was one of the first findings during this process. Therefore we decided to take a **multi-view approach**. Such an approach was also taken by Q. Zeng and J. J. Cimino in their web-based project "Providing Multiple Views to Meet Physician Information Needs" [Zeng and Cimino, 2000] and showed very promising results.

Different views may support different user tasks, different intentions and emphasizes different aspects of the investigated object. By using different views, the single view can be kept simple and clearly focused onto the selected set of characteristics.

Back to our concrete application: The main goal is to visualize treatment plans at runtime (how they were and going to be executed) as well as parameters and variables used within the system. This led to the introduction of two views:

- Logical View
- Temporal View

Whereas the *Logical View* focuses on the logical sequence of plans along with their subelements and the *Temporal View* focuses on visualizing time-oriented aspects of plan execution as well as displaying parameters and variables. Rather than using a single representation including all aspects of *Asbru*, we use two familiar forms of visualization which reduces learning effort and are easier to understand.

Generally, we focused our work on the visualization of plans rather than parameters and variables. This was done due to the fact, that there are already many forms of visualization available for representing quantitative as well as qualitative data over time. See [Aigner, 2001] or [Bade, 2002] for more on time-oriented data visualization in the medical domain.

The following table summarizes which data characteristics are visualized by the two views:

	Logical View	Temporal View[1]
Time-oriented		•
Hierarchical	•	•
Non-uniform element types	•	○
Conditions	•	

Table 5.1: Data characteristics in views.

5.2 Logical View

5.2.1 Asbru Prerequisites

Based on the description in 2.1, the following simplified description of the structure of *Asbru* plans can be extracted:

- An *Asbru* plan may contain the following conditions:
  - **filter precondition:** Only if this condition evaluates to TRUE, the plan gets executed.
  - **abort condition:** If this condition evaluates to TRUE, the whole plan aborts. This condition is valid and checked all throughout plan execution and is getting forwarded to subplans.

[1] There is just a distinction between plans and single plan steps in the *Temporal View*. Single plan steps are only displayed on demand (when selected in the *Logical View*). See section 5.4 for more details.

  - **complete condition:** If and only if the elements within the plan body are completed as intended and the complete condition evaluates to TRUE, the plan can complete successfully.

- An *Asbru* plan has a plan-body containing *single-steps* that are executed in one of the following *execution sequences*:
  - **sequentially:** The contained steps are processed one after the other in the given order.
  - **parallel:** All steps get initialized at the beginning and are processed in parallel.
  - **any-order:** Same as *sequentially* except that the execution order is arbitrary.
  - **unordered:** The contained steps can be executed in an arbitrary way.
- A *single-step* is one of the following:
  - **Variable assignment:** An expression is getting assigned to a plan variable.
  - **If-Then-Else:** If the condition of the construct evaluates to TRUE, the *then-branch* otherwise the *else-branch* gets executed if present.
  - **Ask:** An external, typically user entered value is assigned to the specified parameter.
  - **Plan activation:** The specified plan gets activated.

5.2.2 Plan Step Elements

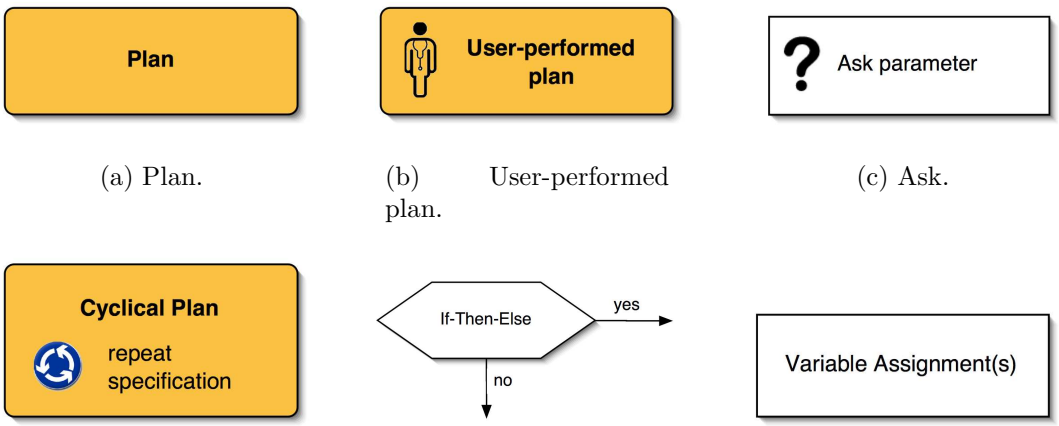
The used visual plan step elements are based on the elements of the flow-chart like representation of the *Proposal for clinical algorithm standards* by the *Society for Medical Decision Making, Committee on Standardization of Clinical Algorithms* [Society for Medical Decision Making, 1992].

We added one *plan element* and a number of symbols for depicting parts of the *Asbru* language that could not be visualized by the elements of the proposal (see Fig. 5.1 for an overview):

- **Plans** respectively **plan activations** are represented by a rounded rectangle filled with the plan color[1] (see Fig. 5.1(a)). In case of being a **cyclical plan**[2], an additional roundabout icon as well as the repeat specification in textual form are presented within the rectangle (see Fig. 5.1(d)). Furthermore, a physician icon appears within the element if the plan is **user performed** (see Fig. 5.1(b)).
- **Variable assignments** are represented by a rectangle containing the assignment textually (see Fig. 5.1(f)).

[1] A distinct color is assigned to each plan, making it easier to distinguish plans from other elements and helping to recognize them in other parts of the representation.

[2] See the section on *Asbru* (2.1) for more information about cyclical plans.



(a) Plan. (b) User-performed plan. (c) Ask.

(d) Cyclical plan. (e) If-Then-Else. (f) Variable assignment.

Figure 5.1: Plan step elements.

- **If-Then-Else** constructs are shown as hexagons having the condition displayed textually (see Fig. 5.1(e)). The *then-branch* of the construct is always connected via an arrow originating at the right top of the element, and the *else-branch* via an arrow originating at the bottom of the element. The branches are labelled by the word "yes" (*then-branch*) respectively "no" (*else-branch*) right next to their connecting arrow lines.
- **Ask** steps of a plan are represented by a rectangle including a question mark ("?") symbol and the text "Ask" followed by the parameter to be entered into the system (see Fig. 5.1(c)).

5.2.3 Anatomy of a Plan

Using the elements just presented, we are able to visualize the single steps within the plan body of an *Asbru* plan. For depicting the conditions and the execution order of the plan steps, an enclosing frame was created, containing the following parts (see Fig. 5.2):

The topmost bar is filled with the plan color and contains the **title of the plan**.

Below the plan title, the **abort condition** is shown. It is represented by a red bar having a *stop sign icon* at the left side. Right besides this icon, the *abort condition* is printed textually. This condition has the following semantics: If the condition evaluates to TRUE, the current plan gets aborted. Furthermore, this condition is valid during the entire execution of all steps in the *plan body*.

The green bar at the bottom of the plan represents the **complete condition**. It has a *checked finish flag icon* at its left and contains the *complete condition* textually. The semantics of this condition is: If and only if this condition evaluates to TRUE, the plan can complete successfully.

The biggest part of the representation is dedicated to the *plan body* of the de-

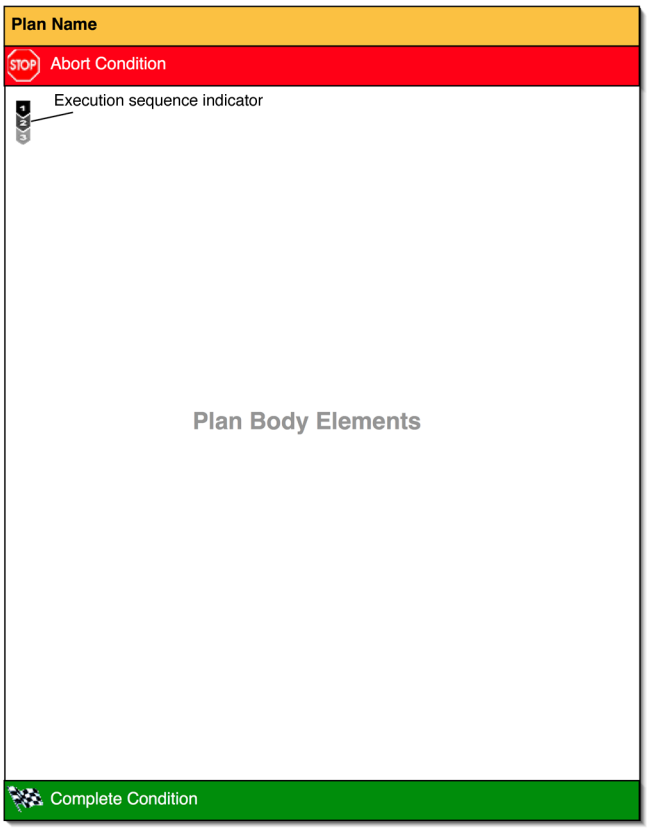


Figure 5.2: Structure of the *Logical View*.

picted plan along with an icon on top showing the execution order of the elements enclosed. The **execution sequence indicator** has four possible icons:

(a) Sequentially. (b) Parallel. (c) Any-order. (d) Unordered.

Figure 5.3: Execution sequence indicator icons.

- The first one showing three arrows on top of each other having ascending numbers on their top stands for **sequential** order (see Fig. 5.3(a)).
- The second one showing the three icons side by side at the same height level

represents **parallel** execution order (see Fig. 5.3(b)).

- The third icon showing three arrows again on top of each other but having arbitrarily sequenced numbers stands for **any-order** execution (see Fig. 5.3(c)).

  Note: Additionally, every arrow has a different shade of gray making it easier to distinguish between the *sequential* and *any-order* icon.

- The last one of the execution order symbols shows arrows besides and on top of each other beginning at different height levels. This icon represents **unordered** execution sequence (see Fig. 5.3(d)).

The rest of the plan body area contains plan elements as described in the last section. If the execution order of the elements is *sequentially*, the elements are additionally connected by arrows.

5.2.4 Used Symbols

Many of the used symbols within the *Logical View* are taken from the *traffic metaphor* (stop sign, roundabout sign, checkered finish flag). These symbols have been chosen due to the fact that they are well known in large parts of the world, the semantics of this symbols are quite unambiguous, and they were also used in *AsbruView*, the *Asbru* plan creation tool [Kosara, 1999, Kosara and Miksch, 2001a, Miksch and Kosara, 1999, Kosara and Miksch, 2001b].

The rest of the icons represent semantics that could not be covered by symbols of the traffic metaphor. Therefore new ones were created: *Execution sequence icons*, *Ask*, *User-performed plan* (see Fig. 5.3 and Fig. 5.1)). These icons were designed focusing mainly on simplicity, ease of understanding, clearness, discernibility, and gender independence (physician icon).

5.2.5 Current Position

During execution of a plan, already executed and currently running parts are marked by using bolder lines for element borders and arrows. This way the current execution position can be determined.

5.2.6 Interaction

Regardless of the fact that the static form of the *Logical View* as described so far contains all information needed and may also be useful in a printed form, adding user interaction increases the usefulness much more.

One used symbol for that purpose not mentioned so far is a **small gray triangle** at plan elements and plan titles (see Fig. 5.4). This triangle indicates if an element has subelements (triangle pointing to the right) and if the subelements are currently expanded (triangle pointing to the bottom). In case an element has no subelements, no triangle is shown at all.

By clicking a triangle pointing to the right, the element is getting expanded, which means navigating down in the hierarchy. When clicking a triangle pointing to the bottom, the element is getting collapsed, which means navigating up in the

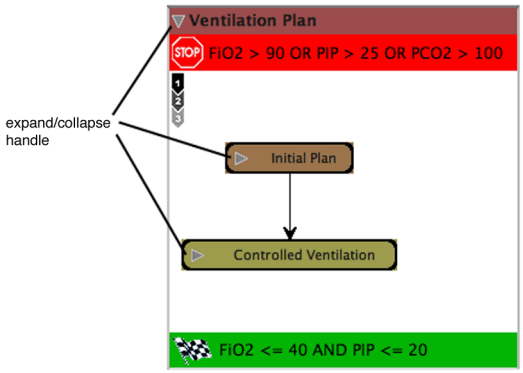


Figure 5.4: Expand/collapse handle.

hierarchy. The use of those triangles is intuitive and based on their application in file system viewers as for example the *Finder* of the Macintosh™ system.

Furthermore, the elements of the *Logical View* can be dragged and resized in case the applied automatic layout is not delivering the desired results.

Other forms of interaction are described in the section *Coupling of Views* (5.4).

5.2.7 Focus + Context

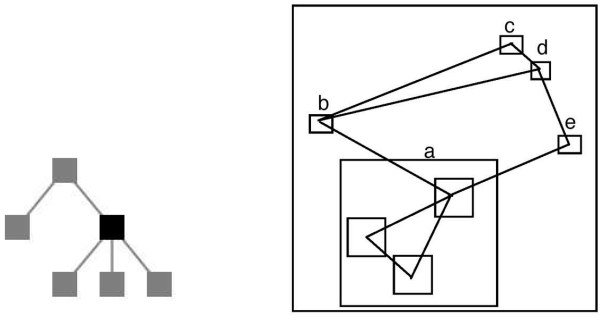
Losing track of the actual position within a plan is quite easy when just using the visualization presented so far.

Overview + Detail The first utility overcoming this problem is the *Overview + Detail display*. It shows a small tree like representation of the whole plan, marking the current view position (see Fig. 5.5(a)). This *Overview display* is only shown on demand (triggered by the user) for not overloading or cluttering the screen.

Fisheye View The second utility avoiding to get lost within a plan is the *Fisheye display* whereas the current (sub)plan represents the focus. In principle, *Asbru* plans can be seen as *hierarchically clustered networks*. Schaffer et alėxamined visualization techniques for that kind of systems in their work on "Navigating hierarchically clustered networks through fisheye and full-zoom methods" [Schaffer et al., 1996] (see Fig. 5.5(b)). The authors show that the *Fisheye display* is particularly useful but for certain purposes (i.e. examining a specific problem within a selected node), *full zoom* is more appropriate. Therefore we use a button for toggling the *Fisheye* vs. *Full zoom* display.

5.2.8 Discussion

The flowchart-like representation of so-called *clinical algorithms* [Society for Medical Decision Making, 1992] is well known among physicians, because it is used frequently in literature and is part of the education of physicians as our user study proved.



(a) Overview display.

(b) Fisheye display [Schaffer et al., 1996].

Figure 5.5: Focus + Context displays.

Asbru is too powerful to be translated completely into a Flow chart representation. The main difficulty in that sense is the state machine characteristic regarding plan conditions. Therefore, the most accurate visualization for *Asbru* plans would probably be State Transition Diagrams. But this type of visualization is not well known, requires high learning effort and might not be accepted by physicians.

Furthermore, our user study showed that minimal learning effort and ease of understanding are essential and most important, given that the tool should not be limited to specialists or academic purposes only.

Based on these arguments we decided to use a flowchart-like representation. We are fully aware that the used visualization is not accurately representing how an *Asbru* plan is going to be executed. But we think that the mental model we are trying to create by this visualization is close enough to the actual execution model being at the same time familiar and easy to understand. An absolutely accurate representation would require a much less simple and clear visualization but only show subtle differences in the used model.

5.3 Temporal View

The *Temporal View* of our tool focuses on the time-oriented aspects of *Asbru* plans as well as displaying parameters and variables. The time interval covered by this view can span from a point in time in the past to a point in time in the future. Whereas only plans having time annotations (temporal planning attributes) can be displayed accurately in the future. Parameters and variables can only be displayed for past and present because they are only known and valid starting right at the point they appear.

As already mentioned, we focused on designing a visualization for displaying plans rather than parameters and variables.

5.3.1 Facets

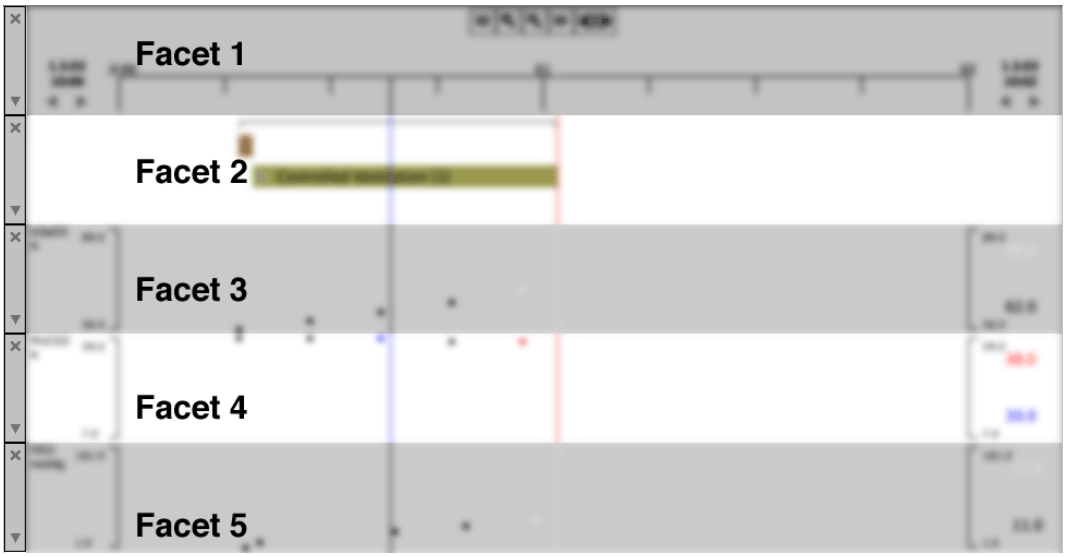


Figure 5.6: Facets.

Spatially, the *Temporal View* is divided into horizontal display areas, so-called **Facets** (see Fig. 5.6). These *Facets* have a handle for manipulation at their left border. This handle contains an icon for *closing* the *Facet* and another one for *expanding* respectively *collapsing* it.

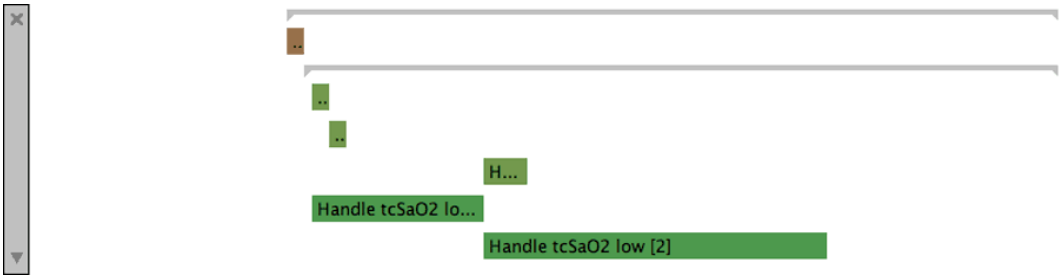


Figure 5.7: Ventilation plan in expanded view.

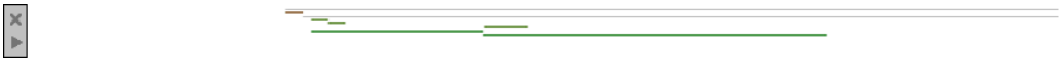


Figure 5.8: Ventilation plan in collapsed view.

In collapsed display, a vertically very small, downscaled (geometrically as well as semantically) version of the Facet's content is displayed (see Fig. 5.7 and Fig. 5.8). This interactive feature is particularly of use when more vertical space for other

Facets is needed by showing main information of the collapsed Facet without hiding it completely. Hence, this is a form of **Focus + Context** visualization in the vertical dimension. (Focus + Context view for the horizontal dimension is mentioned below in Section 5.3.2.)

Furthermore, the height of the individual Facets can be adjusted by mouse manipulation (dragging the lower border of a Facet up or down).

5.3.2 Time Scale

The topmost element within the *Temporal View* is the *time scale*. It is the most important element controlling all others by defining the time interval that is displayed. The time scale display basically consists of the following parts (see Fig. 5.9):

- start and end label,
- the scale including its captions and
- several buttons for manipulating the scale

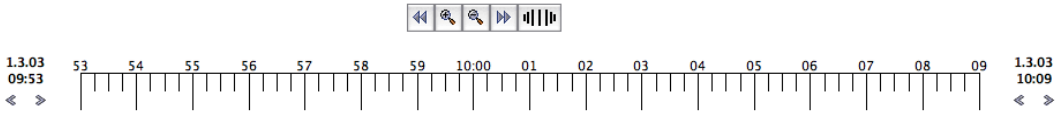


Figure 5.9: Time scale.

In principle, the *time scale* is defined by two points in time representing start and end of the display interval. These points can be set arbitrarily to points in time in the past, present as well as the future. Furthermore, they can be manipulated independently from each other by clicking the two small arrows below the labels for start and end instant.

Due to the fact that the duration of *Asbru* plans can span from milliseconds to years, the time scale has to be extremely versatile in terms of label formatting and setting up a clever strategy for drawing scale division ticks and their granularity.

Focus + Context

A valuable feature offered by the *time scale* is the **Fisheye display**. It magnifies a part of the scale interval (*focus*) and at the same time demagnifies the areas to the left and right of the focus (*context*) (see Fig. 5.10). This way, the area of interest is emphasized for detailed examination without hiding information before and after that area, thus preserving the "full picture".

The Focus + Context technique applied here is one having a non-continuous transformation function based on the *Bifocal Lens* [Spence and Apperly, 1982,Leung and Apperley, 1994].

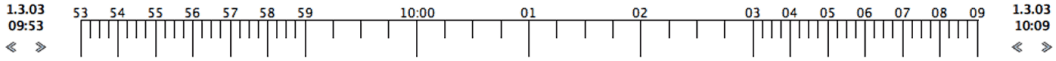


Figure 5.10: Fisheye time scale.

Interaction and Manipulation

The used *time scale* offers many ways of user interaction to manipulate the time scale:

Scale Shift The leftmost of the buttons on top of the element shows two arrows pointing to the left. By clicking this button, the whole scale interval is getting shifted back in time (to the left). This is done by decreasing the value of the begin and end instant by the width of the small interval divisions.

The third button is offering the same functionality for the opposite direction. It is shifting the scale forward in time (to the right).

Zoom The two buttons in between the shift buttons show a magnification glass having a "+" or "-" on them. These buttons are used for zooming in (+), respectively out (-). The zoom is done by applying a configurable factor (i.e. 50 %) to the interval in the center of the scale.

Fisheye Manipulation The rightmost button is used for toggling between Fisheye and Normal display. When switching to Fisheye display, the focus and context intervals can be manipulated by mouse clicking and dragging.

Furthermore, the zoom in and out buttons are now applied to the focus interval only, thus adjusting the magnification factor of the focus area.

Begin/End Time Manipulation As already mentioned, the begin and the end points of the scale can be manipulated independently by using the two small arrows below each label.

5.3.3 Visualization of Plans

For visualizing plans within the *Temporal View* we created two novel glyphs, namely two extensions of LifeLines: **LifeLines+** and **PlanningLines**.

LifeLines+

Our *LifeLines+* enable visualizing a rich set of data aspects:

- temporal attributes
- element characteristics
- hierarchical structure

Figure 5.11: LifeLine+ having a property symbol at the right.

A *LifeLine+* has a defined beginning and a defined end enable drawing a bar connecting those two points in time. The bar includes the title of the depicted incident plus a number of optional elements:

Exceed Indicators If the *LifeLine+* exceeds the specified display interval (and the line can not be drawn completely), small gray arrows at the exceeding borders indicate that the line continues in that direction (see Fig. 5.12). Additionally, the caption is omitted in case the *LifeLine+* exceeds the display interval at the beginning.

Figure 5.12: A LifeLine+ exceeding the end of the scale.

Property Symbols Furthermore, custom symbols can be added to the *LifeLine+* that are displayed within the bar aligned to the right (see Fig. 5.11). These symbols can be used to indicate simple properties of the depicted incident. For visualizing *Asbru* plans we use symbols to indicate cyclical plans or plans that have been aborted.

Visualizing Hierarchy All aspects mentioned so far concerned time oriented issues and simple element properties. An important feature added as well is the capability to visualize hierarchies. The fact that an element contains subelements respectively has child elements, is indicated by a small triangle pointing to the right in front of the bars caption (analog to the *Logical View*). By clicking this triangle, the element gets expanded. Here, the child elements are getting displayed as *LifeLines+* and the expanded element itself is reduced to a gray, so-called *summary line* (see Fig. 5.13). By clicking onto this *summary line*, the reverse effect is triggered and the element gets collapsed into a *LifeLine+*.

Figure 5.13: Expanded LifeLine+.

The basic idea for this representation was inspired by the work of C. Plaisant et al(see section 3.2.2 and [Plaisant et al., 1996, Plaisant et al., 1998]). We modified this representation by adding symbols and - most important - introduced a notion to depict hierarchies.

By reducing expanded *LifeLines+* to *summary lines* we minimize visual clutter and keep the display simple and clear. The basic idea of the visual representation of those *summary lines* was taken from *GANTT charts* that provide similar functionality.

This novel visualization depicts a combination of data aspects that can be found frequently in various kinds of applications, helping to gain new insights into the underlying data (structure).

PlanningLines

PlanningLines are, as the name indicates, intended for depicting planning data afflicted with temporal uncertainties. Beneath all aspects visualized by the just presented *LifeLines+*, *PlanningLines* offer additional support for the following rich set of time attributes:

- start interval (earliest starting shift + latest starting shift)
- finish interval (earliest finishing shift + latest finishing shift)
- minimum duration
- maximum duration

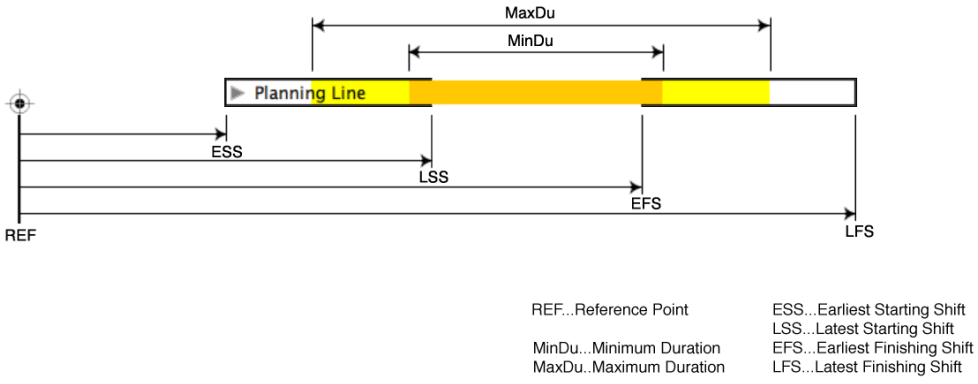


Figure 5.14: PlanningLine.

The glyph itself consists of three main parts: The **start cap** at the left, the **end cap** at the right, and the **duration bars** in between (see Fig. 5.14).

The *mental model* used when designing the glyph is the following:

The two caps to the left and right are holding the bars in between. The caps are fixed mounted but the duration bars can be moved to the left and right as much as the caps allow them to (see Fig. 5.15). More precisely this means that the inner bars can not be shifted beyond the vertical borders of the caps and in the opposite direction just as much as the bars are held by the caps. If shifted further, the bars would fall down. Furthermore, the inner bar can grow as much as the outer bar allows.

Figure 5.15: PlanningLine flexibility.

The caps are drawn in black to emphasize their fixed position. The bars in contrary are colored whereas the color of the maximum duration bar has equal hue and saturation but higher brightness as the minimum duration bar.

All other aspects of the glyph are the same as introduced for the *LifeLine+* (title, exceed indicators, property symbols, hierarchy).

Figure 5.16: Expanded PlanningLine.

In case some of the attributes are not present, the following display rules apply:

- If the missing attributes follow from the others uniquely, they are displayed as they would be present.
- If the "earliest starting shift" is missing, but the "latest starting shift" is present or the "latest finishing shift" is missing, but the "earliest finishing shift is present", "latest starting shift" respectively "earliest finishing shift" are displayed as diamonds (see Fig. 5.16).
- In all other cases, parts are not drawn if their associated attribute is not present.

Several other visualization techniques for depicting temporal uncertainties are available as presented in Chapter 3 (State of the Art Report). We were looking for a representation that can be combined with traditional *LifeLines* seamlessly, at the same time being clearly and intuitive. Therefore we created this new glyph, called *PlanningLine*, mainly based on the *Time Glyph* of *AsbruView* [Kosara, 1999, Kosara and Miksch, 2001a, Miksch and Kosara, 1999, Kosara and Miksch, 2001b] and *Temporal Objects* [Combi et al., 1999], overcoming most of the drawbacks of these visualization techniques (see Chapter 3).

Our glyph is based on the simple mental model of a bar held by two mounted caps, making it intuitive and easy to understand. The graphic representation is kept very plain and simple associating an exact meaning to each line respectively element without adding any other additional items.

Discussion

The use of *Time Lines* and *LifeLines* for representing incidents over time is a well known technique that turned out to be clear, straight forward, and intuitive. Additionally, our user study showed, that *LifeLines* are the preferred visualization technique for depicting plans over time (see Chapter 4).

Therefore we based our design closely on *LifeLines* as presented by C. Plaisant et alând the various modifications as presented in the *State of the Art Report* (see Chapter 3). We added and modified certain elements to get two visual representations able to fulfill our requirements for depicting *Asbru* plans.

But these representations are not limited to visualize *Asbru* plans only: They can be applied for various purposes, depicting hierarchical, time oriented incidents including temporal uncertainties for planning data. One further possible field of application is project planning. Using the visualization technique presented here, GANTT and PERT diagram facilities can be combined plus a rich set of user interaction possibilities is offered.

5.3.4 Time Dimensions

As already mentioned, the *Temporal View* is able to cover the three basic tenses past, present, and future. Whereas we can neglect the present for displaying plans because it is a single instant, therefore having no horizontal extent. The present respectively current time represents the borderline between past and future.

For displaying plans that have started and ended in the past, therefore being fully defined, we use *LifeLines+*. On the other side, we use *PlanningLines* to depict plans that (might) be executed in the future being afflicted with temporal uncertainties. For plans that have started already but are still executing, the already executed part is depicted using *LifeLines+* and the part to be executed in future by using *PlanningLines* (see Fig. 5.17).

Furthermore, a color palette is used for coloring plans or parts of plans that have already been executed and a grayscale palette for plans in the future. The colors used for plans correspond to the ones used in the *Logical View*.

5.3.5 Plan vs. Plan Instance

Basically, an *Asbru* plan can be seen as a template. This template gets instantiated whenever the plan gets executed. Furthermore, more than one instance might be created for a single plan. More precisely that implies that the *Temporal View* contains plan instances (already executed or still running plans) as well as plans (templates to be instantiated in future).

This pattern can be seen analog to the Class - Object relationship in Object Oriented Programming.

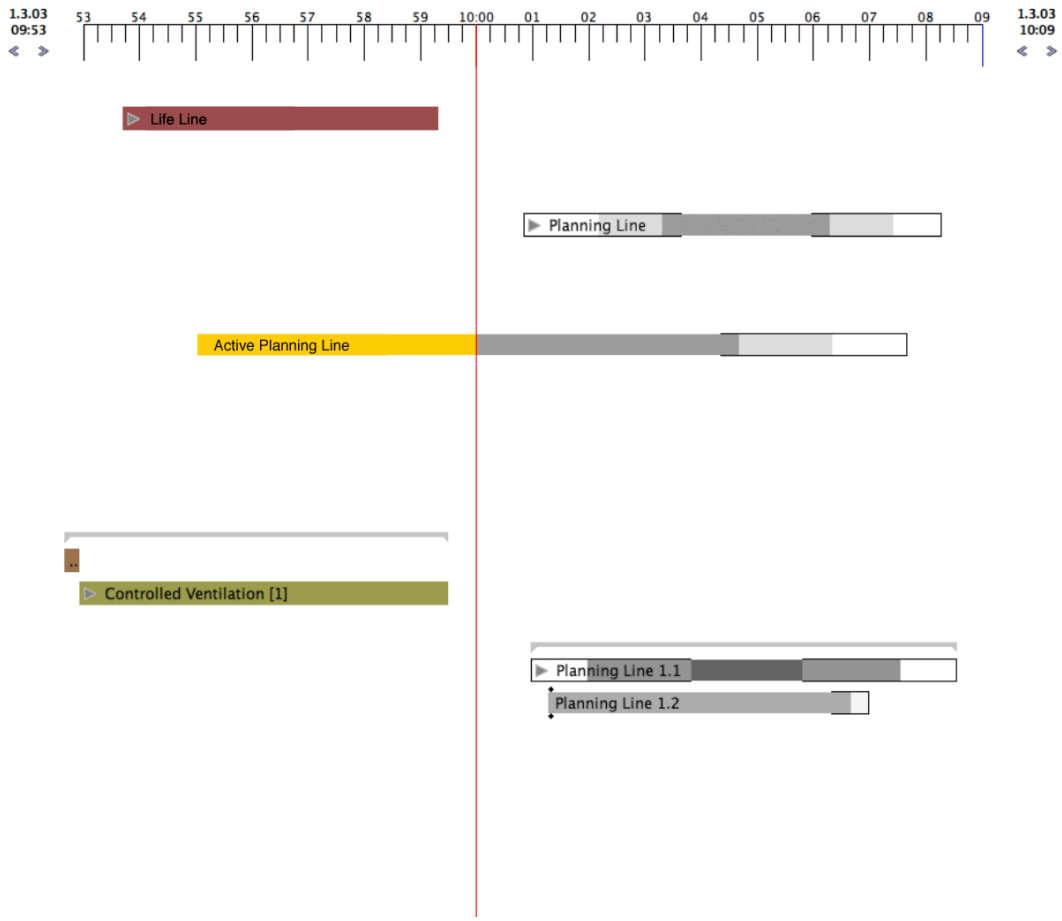


Figure 5.17: Temporal View elements.

5.3.6 Current Time Indicator

An item shared by all elements of the *Temporal View* is the *current time indicator*. It is a simple, vertical (in our case red) line, marking the current time. Furthermore, the current time is displayed precisely in the upper right corner of the application window (see Fig. 5.18).

5.3.7 Time Cursor

Another shared item is the *Time Cursor*. This particularly useful element marks an arbitrary point on the time scale. It is represented by a simple (in our case blue) vertical line and can be manipulated by mouse clicking + dragging (see Fig. 5.18). The precise value of the *Time Cursor* is displayed at the bottom of the application window right below the vertical time cursor line.

The *Time Cursor* can be used for example to inspect variable and parameter values at certain points in time, measure beginning and ending of plans, or compare various elements.

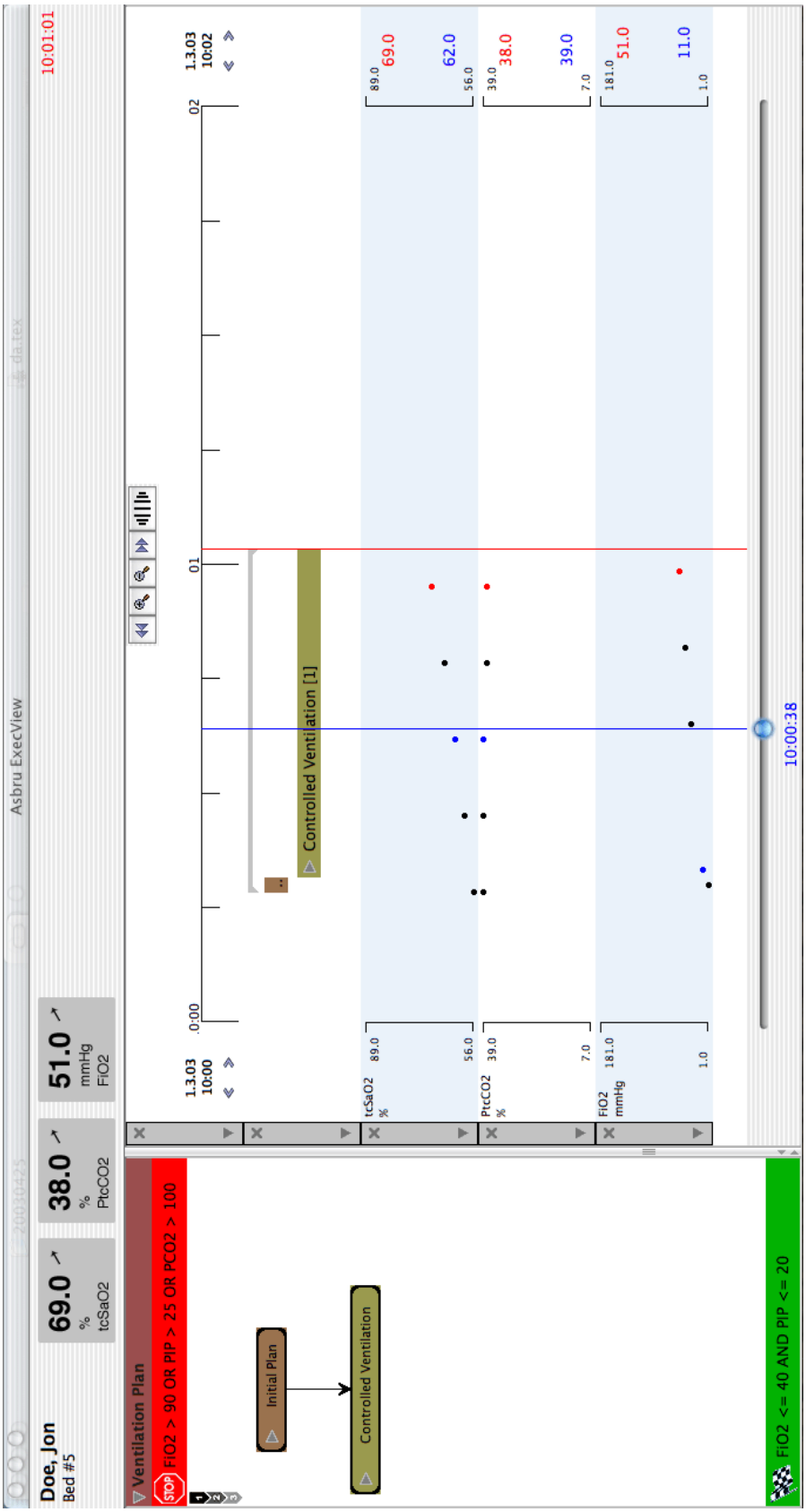


Figure 5.18: Application window showing the execution of a plan (Mockup).

5.3.8 Visualization of Variables and Parameters

As already mentioned, this work focuses on the visualization of plans and provides just very limited support for the visualization of parameters and variables. We only included the visualization of quantitative data as simple diagrams. More design ideas for that matter can be found in the related projects [Aigner, 2001] and [Bade, 2002].

The diagram form we used is based on E. Tuftes work "Graphical summary of patient status" [Tufte and Powsner, 1994]. The data points are plotted as dots along the time axis. The currently valid point is marked using the color of the *Current Time Indicator* (in our case red) and also presented precisely in numerical form to the right of the diagram (see Fig. 5.18). Furthermore, the data points valid at the instant of the *Time Cursor* are marked using the *Time Cursor* color (in our case blue) and also presented numerically to the right of the diagram using the same color.

The y-axis is shown both at the right and left side of the diagram including its begin and end value numerically. Moreover the diagram is labeled by the variable/parameter name and its unit at the left side of the diagram.

5.3.9 QuickView Panel

A separate possibility to display currently valid variable and parameter values is the so-called *QuickView Panel* in the top part of the application window (see Fig. 5.18). The panel consists of rectangular areas that can be assigned to the available parameters respectively variables.

A single item shows the current value along with its title, unit, and a trend indicator.

Thus, the *QuickView Panel* allows to monitor the most important values by putting them at a prominent position, enlarged in size and without the need of displaying the complete history in an additional *Facet*.

5.4 Coupling of Views

Using different views for visualizing a complex system unfolds its power only if those views are coupled and work together hand in hand:

Color Usage Every plan is getting a distinct color associated. This color scheme is shared among the views allowing quick recognition of plans across views. Moreover, plan instances use the same color as their plan (template) indicating that they belong together.

Selection Selecting an element in one view (indicated by changing the caption color from black to white), also selects the corresponding element(s) in the other view. The usage of the plural "elements" originates from the plan vs. plan instance relationship (see Section 5.3.5). This means that selecting a plan in the *Logical*

View selects all instances and plans (templates) corresponding to that plan in the *Temporal View*.

Furthermore, selection only takes place if the correspondent elements are currently visible, hence no automatic expansion of elements is done.

Propagation Basically, navigation is done independently in the two views, which means that if an element gets expanded in one view, the correspondent element(s) in the other view is/are **not** expanded as well. But the actual selection can be propagated to the other view on demand. By double clicking, the actual selection gets propagated, which means automatic expansion of elements to make the actual selection visible in case it is hidden.

5.5 Design Technique

During all stages of the conceptual design process, paper and pencil mockups were used to visualize ideas and communicate them. This technique is cheap, time, and resource saving and at the same time giving a good impression of the design and serves as basis for its discussion.

More detailed mockups created using graphic programs were only produced sparsely at advanced stages of the design to show a more detailed picture more closely to the intended appearance.

5.6 Design Evaluation

When having completed the first "release" version of the conceptual design, we conducted an evaluation session for getting early feedback regarding our design. This early evaluation process was very valuable and reduced the risk of investing time and effort for might going in the wrong direction.

The evaluation was done by two experts: one person is a visualization expert having experience with medical software development and the other one is a physician (medical expert) having visualization knowledge.

The result of the evaluation was very positive, validated our concept, and showed that we were working in the right direction. Only some minor issues of the design were objected which led to an improvement of the design.

Chapter 6

Prototype Implementation

6.1 General

The next logical step after having finished the conceptual design process was to proof our concept by implementing a software prototype. This prototype should demonstrate the main characteristics of the design to proof that our concept works at all, to get a better impression of the look and feel, see how the interaction patterns are working, and play around with. Furthermore, it should act as basis for further evaluation by potential users.

The prototype was implemented using the programming language *Java* (JDK 1.4.1). The visual elements of the prototype are based on Java's *Swing* standard component library.

Throughout this chapter we use *UML*[1] *class diagrams* for explaining our prototype implementation. See Appendix E for a description of the used UML elements.

The Java project has the following package structure (see Fig. 6.1):

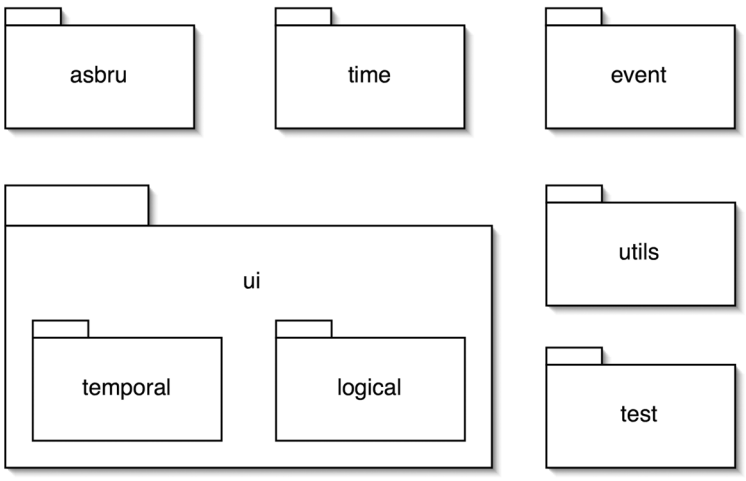


Figure 6.1: Package structure.

[1]UML = Unified Modeling Language

- **asbru**: Classes that model parts of *Asbru* or deal directly with *Asbru* related features.
- **time**: Classes and interfaces modeling general temporal elements and their relationships.
- **event**: Classes and interfaces implementing the event model of the prototype's user interface.
- **ui**: User Interface classes.
- **ui.temporal**: User Interface elements for the *Temporal View*.
- **ui.logical**: User Interface elements for the *Logical View*.
- **utils**: Various utility classes.
- **test**: Test classes for unit testing of the core classes.

6.2 Approach

For implementing the prototype we applied a *rapid prototyping* approach with small development cycles (about two weeks). Hence, the prototype evolved step by step whereas the analysis and design steps were done only for the next development cycle and existing parts were getting constantly refactored.

This approach has been taken from the *Extreme Programming (XP)* technique [Beck, 2000,Astels et al., 2002,ExtremeProgramming.org, 2002,XProgramming.com, 2002]. Furthermore, another XP technique, namely "unit testing", was applied for the classes implementing core functionality of our prototype. This means writing tests first and doing the actual implementation after that, which has a lot of advantages. First of all one can see instantly if the implemented part works by running the test and is forced to work out the external behavior clearly upfront by writing the test.

6.3 Architectural Issues

6.3.1 MVC Paradigm

The basic structure of the prototype resembles the *MVC* paradigm. *MVC* is the acronym for the three components: **Model**, **View**, and **Controller**. Whereas the *model* is the core element building up the system structure. The *model* can have one or more *views* associated to it. *Views* are passive, meaning that they do not change or manipulate the *model*. *Views* represent the *model* or parts of it (mostly visually). *Controllers* in contrary are associated elements that are responsible for manipulating (parts of) the model. In return, changes caused by *controllers* are reflected by the representing *views*.

Furthermore, the coupling of these three parts is very loose: The *model* does not and should not know anything about its associated *views* or *controllers*. Often, the

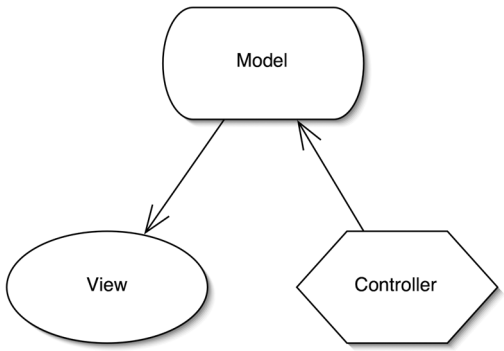


Figure 6.2: Model View Controller paradigm.

separation between *views* and *controllers* is not that strict and elements act both, as *views* and *controllers*.

In our case, *Asbru* plans as well as parameters and variables are the *models*, the *Logical View* and the elements of the *Temporal View* are *views* on them:

	Logical View	Temporal View
Asbru plans	•	•
Asbru plan instances		•
Parameters		•
Variables		•

Table 6.1: Models and views.

Models

Asbru Plans and Plan Instances *Asbru* plans and plan instances are modeled by the classes of the **asbru** package as depicted in Fig. 6.3, Fig. 6.4, and Fig. 6.5 (see Chapter 2 for a description of *Asbru*).

Parameters and Variables Parameters as well as variables are simply modeled as data points over time. They are represented by the *DataRow* class as a *java.util.Vector* of *java.awt.geom.Point2D.Double* points (see Fig. 6.6).

Additionally, methods for determining the minimal and maximal data value as well as getting the value valid at a specified instant were added.

Views

In principle, our prototype contains two basic views of plans: the *Logical View* and the *Temporal View*. For coupling of these views another entity managing them is

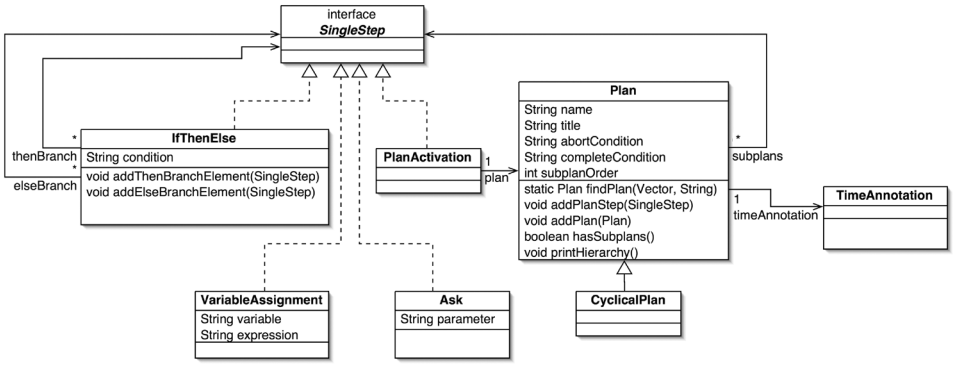


Figure 6.3: Asbru plan model.

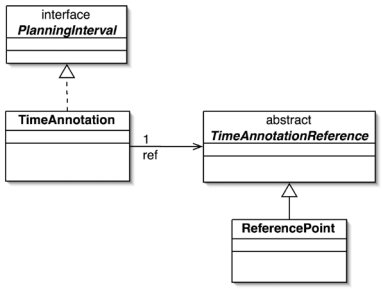


Figure 6.4: Time Annotation for *Asbru*.

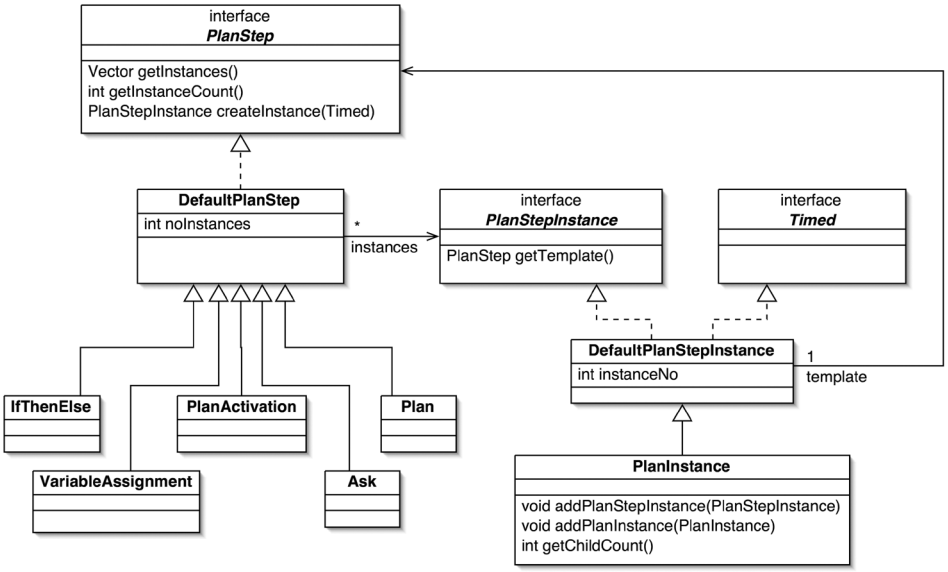


Figure 6.5: Plan step instance classes.

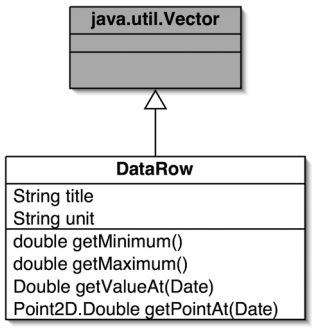


Figure 6.6: DataRow class used for Parameters and Variables.

needed. This task is performed by the *PlanViewManager* class that holds references to all views of *Asbru* plans in the system (see Fig. 6.7).

Furthermore, these views do not represent plans themselves visually but use other view elements for that matter (*LifeLines+*, *PlanningLines*, *PlanGraph* - see Section 6.4 and 6.5 for detailed descriptions).

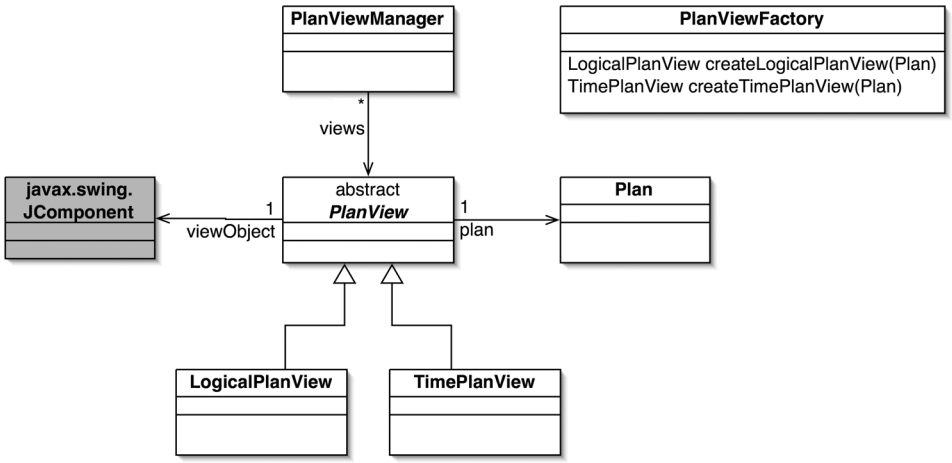


Figure 6.7: Plan views.

6.3.2 Observer Pattern

A heavily used architectural element is the *Observer pattern* [Gamma et al., 1994]. In our case it is particularly applied for interaction event notification.

The user interface (UI) event model used in our application has the following event types:

- **select**: A UI element was getting selected.
- **expand**: A UI element was getting expanded.

- **collapse**: A UI element was getting collapsed.
- **propagate**: The propagation of the current selection has been triggered.

The event sent to all registered listeners is implemented by the *ViewSelectionEvent* class. It holds references to the object sending or resending the event and one to the object originally firing the event.

Classes that are interested in receiving *ViewSelectionEvents* have to implement the *ViewSelectionListener* interface. This interface defines a set of listener methods called when the associated event types are fired.

Furthermore, the *ViewSelectionModel* interface has to be implemented by classes that wish to fire such events and manage the listener list. A default implementation of this interface was created to be used for simple delegation (*DefaultViewSelectionModel*).

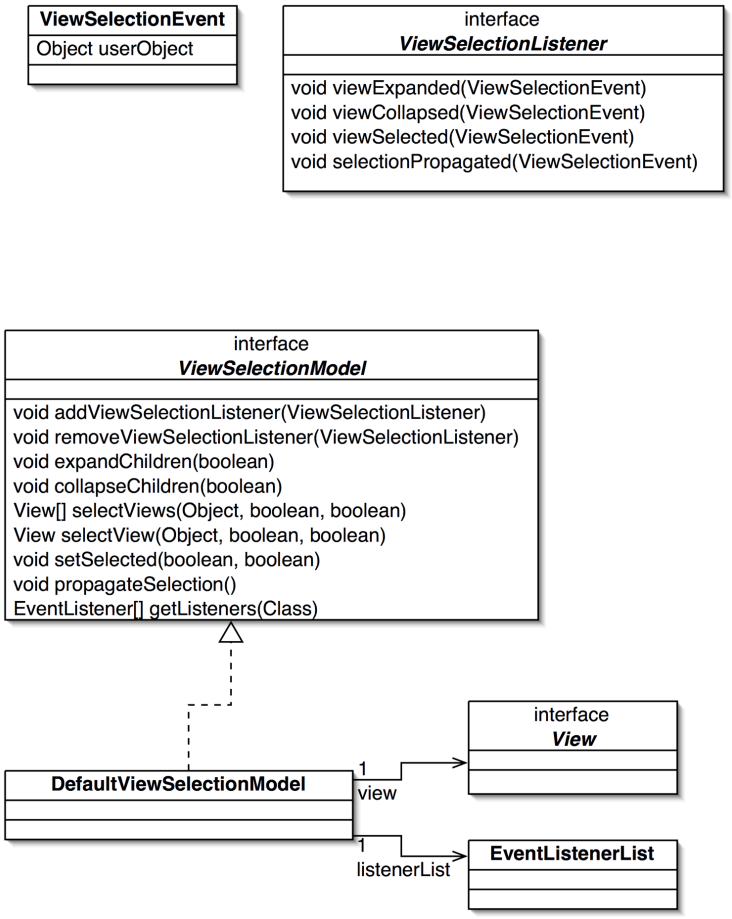


Figure 6.8: View selection event handling classes.

Due to the structure of views, a layered dispatch is used for delivering events. This means that events are passed up in the hierarchy as long as they affect the next

level. When the *PlanViewManager* is reached, it passes the event down to all other plan views (see Fig. 6.9).

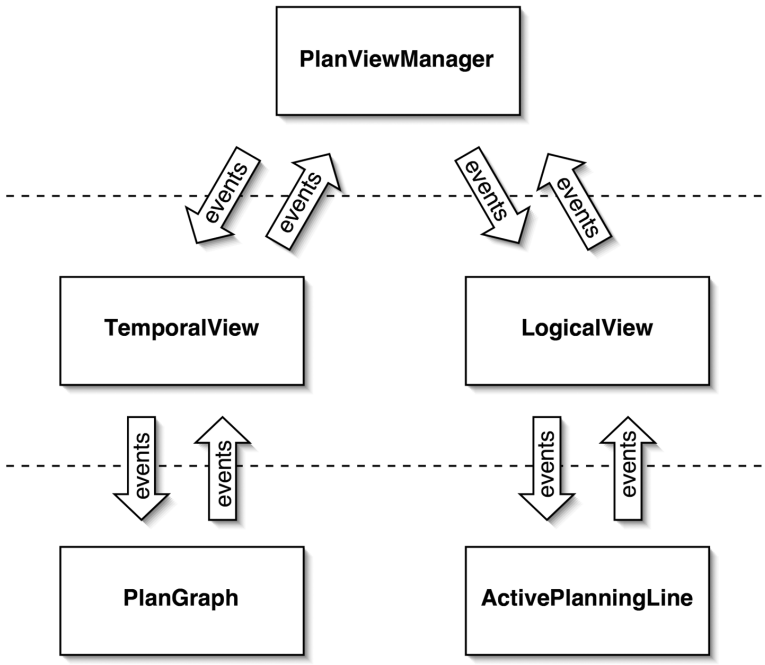


Figure 6.9: Layered dispatch for views.

6.3.3 Factory Pattern

Another software pattern getting used is the *Simple Factory*. We use a factory for creating views out of a given *Asbru* plan (*PlanViewFactory* - see Fig. 6.7).

Firstly, this shifts lengthy and complex object creation code from the objects to the factory and secondly, it strictly separates *Asbru* related code from the used general purpose views as you can see in Section 6.5.

6.4 Logical View

6.4.1 JGraph

The *Logical View* of our tool is used to visualize the logical structure of an *Asbru* plan. For displaying the flowchart-like part of our representation to depict plan step elements, we use the graph drawing framework **JGraph** [Alder, 2002a, Alder, 2002b].

JGraph is a general purpose graph drawing framework developed by Gaudenz Alder. It is a flexible, small, and powerful package using the MVC model (see 6.3.1). It is structured analogous to the standard *Swing* component *javax.swing.JTree*.

JGraph uses a general model of graphs consisting of **nodes** and **edges** connecting them. Views are defined for representing graph cells (nodes and edges) which in

turn utilize renderers that do the actual screen painting work (using the *Flyweight* pattern [Gamma et al., 1994]).

We used the *JGraph* package Version 2.1 (Geneva) which is distributed under the terms of the *GNU Lesser General Public License*. The framework is very well documented and even more important, profound and fast support is provided by message boards of the project at [Alder, 2002b].

6.4.2 PlanGraph

To use *JGraph* for our purpose, we had to build application dependent graph cells, views, and renderers by subclassing the present default implementations creating a *PlanGraph* component including them. Furthermore, creating layout algorithms for positioning cells and drawing edge paths was necessary.

6.5 Temporal View

6.5.1 Time Model

Time Interfaces

The skeleton for time oriented objects is resembled by a set of interfaces along with their relations (see Fig. 6.10).

The basic interface characterizing an object to be time oriented in general is *Timed*. The interfaces *Event*, *Interval*, and *PlanningInterval* on the next level represent the three basic types of time oriented objects:

Name	Description	Attributes
Event	An incident happening at a single point in time.	time
Interval	An incident representing a process starting at a specific instant spanning to an ending point in time.	start, end
PlanningInterval	A planning incident having temporal uncertainties specified by a set of temporal attributes.	Earliest Starting Time (EST), Latest Starting Time (LST), Earliest Finishing Time (EFT), Latest Finishing Time (LFT), Minimum Duration (MinDu), Maximum Duration (MaxDu)

Table 6.2: Time model.

A further interface combining *Interval* and *PlanningInterval* is the *ActivePlanningInterval*. This is used for objects that used to have planning attributes (*Plan-*

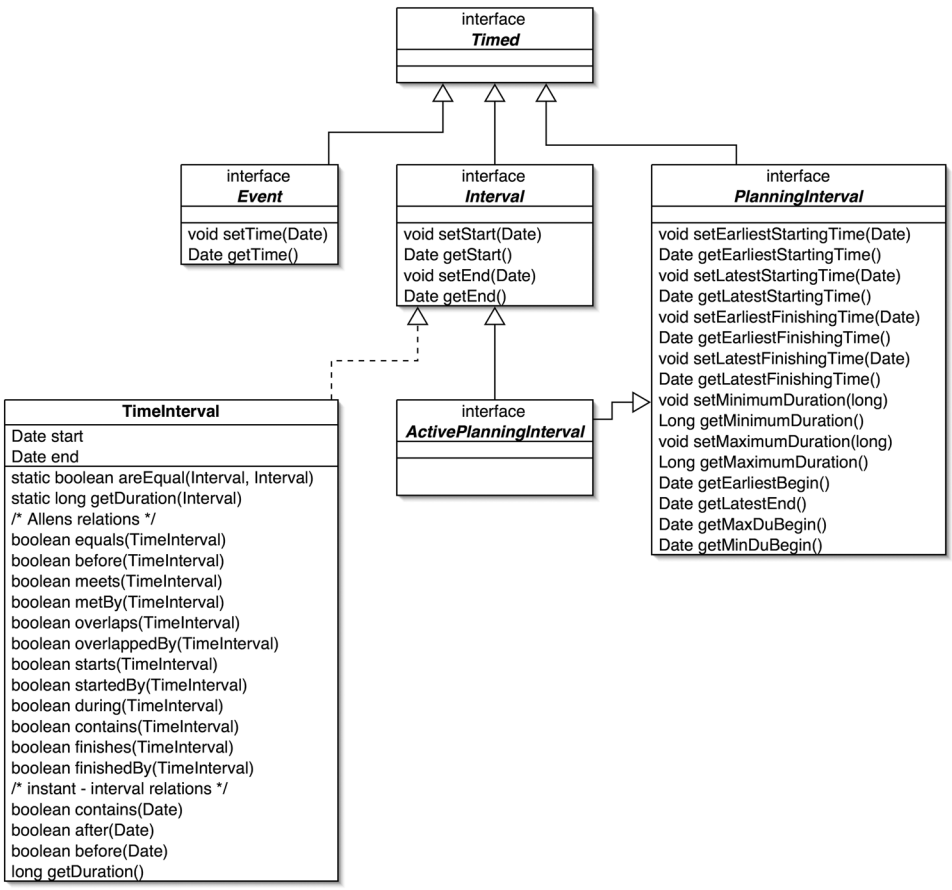


Figure 6.10: Time interfaces.

ningInterval), got executed then, thus additionally have a defined start and end attributes (*Interval*).

TimeInterval Class

The general purpose class *TimeInterval* implements the *Interval* interface along with various methods for comparing and testing intervals (see Fig. 6.10). These methods include tests for all interval relations defined by James F. Allen (*Allen's relations* [Allen, 1983]), instant to interval relations, and methods for calculating the duration of an interval.

6.5.2 Architecture

All elements of the *Temporal View*, except diagrams, are generally views of temporal objects (objects implementing the *Timed* interface or one of its sub-interfaces) (see Fig. 6.12).

This way, an application independent implementation is ensured because views are coupled to *Asbru* plan model classes via time interfaces rather than directly (see

Fig. 6.11).

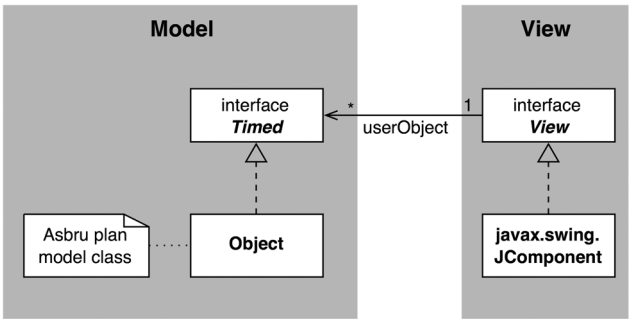


Figure 6.11: Temporal View architecture.

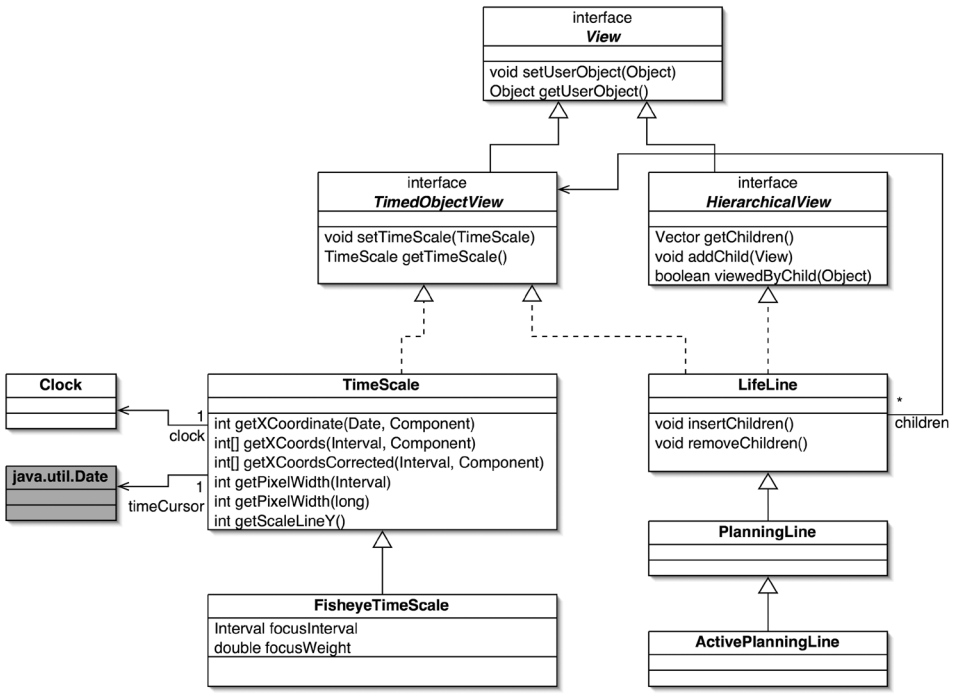


Figure 6.12: Temporal View classes.

6.5.3 Time Scale

The *TimeScale* class plays a central role in the *Temporal View*. It determines the display of all other time oriented elements within the view. Therefore it provides methods for determining screen coordinates for given instants or intervals (see Fig. 6.12).

Furthermore, the class implements a set of manipulation possibilities, namely zooming, shifting, and altering begin and end time.

A more complex part of the implementation is the included "auto granularity" functionality. This contains determining begin and end label formatting as well as setting the granularity for the displayed scale ticks for intervals possibly spanning from milliseconds to years.

Fisheye Scale

A time scale implementing Fisheye functionality was created by subclassing the *TimeScale* class (see Fig. 6.12).

Focus + Context are determined by two parameters:

- focus time interval
- percentage of scale width that is covered by the focus interval

Based on these parameters, the context interval can be calculated:

$$ci = si - (si * scale\,percentage\,focus)$$

ci...context interval pixel width, si... scale interval pixel width

The context portion left of the focus interval and the portion right to it have the same magnification and are divided according to their durations in relation to each other.

Furthermore, an additional button is presented in the *FisheyeTimeScale* object for toggling Fisheye and Normal display.

6.5.4 Layout Manager

Java AWT respectively *Swing* containers have a very clever layout mechanism: A so-called *Layout Manager* is associated to a container being responsible for laying out the components within the container (setting position and size).

There is already a set of predefined layout managers included in the JDK that can be used for a lot of purposes. But none of them is able to perform our special case of laying out components with respect to a time scale. Therefore we implemented our own *layout manager* (*TimeViewLayout*) which is able to set the position and size of a single *Timed* object within a container.

6.5.5 LifeLine+

In principle, a *LifeLine+* is a view of an *Interval*. It therefore has a defined beginning as well as a defined end. The position and size of the *LifeLine+* within the enclosing container is determined by the layout manager *TimeViewLayout* as described in the previous section. To get the temporal attributes translated into screen coordinates, the *LifeLine+* holds a reference to the *TimeScale* object.

The *LifeLine+* is a *Swing* component subclassing *javax.swing.JPanel* and can therefore be integrated into all kinds of applications using *Swing* UI components seamlessly.

If a *LifeLine+* has children, they are hold inside the component within a child container. Hence, *LifeLines+* and their sub-elements are encapsulated.

6.5.6 PlanningLine

A *PlanningLine* is a view on a *PlanningInterval* and subclasses *LifeLine*. It extends the simple drawing capabilities of *LifeLines+* by displaying the set of temporal planning attributes: EST, LST, EFT, LFT, MinDu, and MaxDu[2] as described in the *Conceptual Design* chapter.

6.5.7 ActivePlanningLine

Combining *LifeLines+* and *PlanningLines* is done by the *ActivePlanningLine* class which is a subclass of *PlanningLine*. This element is used to be able to display currently running incidents. Depicting this, *LifeLine+* display functionality is used to show the already executed portion and *PlanningLine* display functionality to depict the portion that is going to be executed in future.

6.5.8 Current Time Indicator and Time Cursor

Both, the *Current Time Indicator* as well as the *Time Cursor* are implemented using a *javax.swing.JLayeredPane* for the *Temporal View*. This means that layers are added to the view:

- **Main layer**: Contains all Facets and their UI elements.
- **Current time layer**: Contains the *Current Time Indicator* represented by a vertical line.
- **Time cursor layer**: Contains the vertical line representing the current time cursor value and a manipulation handle.

Current Time Indicator

The *Current Time Indicator* itself is a view of a *Clock* holding the current time value. The *Clock* object is a thread that notifies the registered listeners regularly in a customizable cycle by subclassing *javax.swing.Timer*.

Time Cursor

The *Time Cursor* is not only displaying the current time cursor value (a *Date* object), but can also be manipulated by a handle. This manipulation handle is implemented in the prototype using a *javax.swing.JSlider*.

6.5.9 Diagram

A *Diagram* is a view for a *DataRow* object that contains (value, time stamp) pairs. To determine the horizontal screen coordinates for the data points, a reference to the *time scale* is held by the diagram.

[2]EST = Earliest Starting Time, LST = Latest Starting Time, EFT = Earliest Finishing Time, LFT = Latest Finishing Time, MinDu = Minimum Duration, MaxDu = Maximum Duration

A speciality of the *Diagram* element is its capability to mark and display the value of the data points valid currently and at the time cursor instant. For this reason a reference to the *time cursor* has to be held additionally.

6.6 Color Manager

An important utility for the coupling of views is the *ColorManager* class. It is a Hashtable that simply associates colors to objects using a custom color palette. This way it is possible to set up a uniform color scheme for all views.

6.7 Not Implemented Features

As mentioned at the beginning of this chapter, the implemented prototype is intended as proof of concept and should implement the main features of our conceptual design. Furthermore, the main focus of this work is directed towards visualization issues rather than a working direct coupling with *Asbru* files or other existing parts of the *Asgaard* project.

Taking this and the fact that the time reserved for implementing the prototype was very limited into account, one can comprehend that certain features mentioned in the conceptual design chapter could not be implemented:

- QuickView Panel (see Section 5.3.9)
- Facet functionality (see Section 5.3.1): Expand/collapse, close, resize are not implemented
- Manipulation of Fisheye parameters (see Section 5.3.2): Focus portion and magnification factor of focus interval are fixed currently.
- Overview in the *Logical View* (see Section 5.2.7)
- Fisheye view in the *Logical View* (see Section 5.2.7)
- Indication of the current position within the *Logical View* (see Section 5.2.5)
- Display of all available parameters and variables for an *Asbru* plan: Only a predefined set of parameters respectively variables can be displayed currently. There is no way to see which parameters are available for a given plan and display them.
- Data abstraction directly from *Asbru* plans: Currently plans are getting generated hardcoded within test classes rather than reading them from *Asbru* files.
- Automatic calculation of plan durations if they are not defined: Now, time attributes have to be defined for plans, even if they possibly could be calculated through time annotations of surrounding plans.

- Direct coupling to the *execution unit*: The tool is currently not directly coupled with the *execution unit*, thus simulating test data sets and plan execution sequences has to be done by test classes.
- Time annotations that reference plan state transitions: *Asbru* offers the possibility of defining time attributes not only relative to fixed reference points but also relative to plan state transitions of other plan instances (see Chapter 2). This form of referencing plan state transitions is currently not supported by the plan model used in the prototype.

Chapter 7

Prototype Evaluation

7.1 Objectives

The next step after having implemented our software prototype as described in the last chapter, was showing this prototype to real users. This was done due to the reason that we did not want to create a tool respectively representation methods that we "believe" meets the needs of physicians, but proving this by conducting a prototype evaluation.

Often users do not know what they want, but they soon see what they do not want if they see it [Preece et al., 2002]. That is why having a prototype was important to show the ideas of visualization techniques more closely to the intended final product as well as interaction issues which cannot be envisioned by paper based mockups.

At this stage, the goal of this process was to evaluate our design by using a prototype for getting a more realistic picture on interaction and look & feel issues rather than a usability study of the prototype itself. Furthermore, a user test focusing on usability issues would not have been appropriate using the created prototype because it implements just core visualization and interaction functionality.

7.2 Method

The evaluation was carried out by conducting interviews with physicians working in intensive care units. Five of the eight physicians who already participated in the user study at the beginning of this work (see Chapter 4) took part in the evaluation. The interviews were performed at their workplaces using a laptop for prototype presentation and testing.

The interviews consisted of the four main parts: Introduction, Prototype Presentation, Prototype Testing, and Feedback / Questionnaire (see Appendix D for the complete evaluation form).

Introduction At first, a general description of the problem was given in order to recall what has been presented in the user study four months earlier. After this, our design approach has been introduced. This included information about the steps we

already took in the development and showing our visualization concepts consisting of the two views: *Logical View* and *Temporal View*. All diagram elements, symbols, and icons used within the *Logical View* were introduced along with their semantics. Furthermore, the parts of the *Temporal View* have been presented along with their meanings. Especially our new *LifeLine* concept including hierarchy was described and the novel *PlanningLine* glyph was explained. Explaining the *PlanningLine* glyph was done by the means of the mental model of encapsulated bars held by two caps as presented in Section 5.3.3.

Especially the status of the prototype was pointed out in order to view it not from a "product ready to use" perspective.

Moreover, general information about the structure of the interview as well as our objectives were given at this point.

Prototype Presentation The next part of the interview was showing the functionality of the prototype via the test data set of a *ventilation guideline*. Here the main focus was directed towards showing how the design was implemented from a user's point of view as well as interaction patterns. This included demonstrating navigational features, the functionality of the *time scale* and the *time cursor* along with the marking of data points in diagrams, and the coupling of views.

Prototype Testing After the presentation, the test persons were able to try the prototype on their own by playing around with it. During testing, the physicians had also the opportunity to ask questions about the design or functionality of the prototype.

Feedback / Questionnaire After the introduction, presentation, and trial phases, the test persons were asked to give feedback. This was done orally along a prepared questionnaire. Questions were asked about

- the prototype in general,
- the concept respectively the used graphical representations,
- the Logical and Temporal View in particular,
- interaction and view coupling issues,
- the applicability of such a tool, and
- if the physicians would be interested to be involved in further development of this software.

(See Appendix D for the complete set of questions.)

Remark: Depending on time constraints of the interviewed persons, prototype testing or certain questions of the interview form were left out.

7.3 Results

7.3.1 General

The feedback regarding our design and prototype given by the interviewed physicians was generally very positive. All of them considered the overall structure clear, simple and not overloaded. The graphical representations, glyphs, and symbols have been judged to be intuitive and clear, keeping the learning effort relatively low. Especially the use of the two views was considered to be useful and appropriate.

The issue being criticized mostly was the somewhat irritating and difficult navigation and zoom behavior of the *Temporal View*. Another generally negative point was the sometimes too low contrast between font and background color for plans and furthermore, one physician pointed out that the individual Facets should be marked off more clearly.

7.3.2 Logical View

All asked physicians considered the *Logical View* representation to be clear, intuitive, and easy to understand. It was appreciated mostly due to its familiar diagram form that is well known from *clinical algorithms* which are widely used in medicine. Furthermore, the used colors were found to be appropriate and the navigation to be intuitive.

Negative points addressed were first of all the missing overview not implemented in the prototype and secondly the rather poor layout within the plan body. One test person pointed out that the diagram elements should not be moveable and resizable.

7.3.3 Temporal View

Generally the *Temporal View* was judged to be clear and simple, clearly depicting when something happened. Above all, the provided *Time Cursor* was appreciated and considered to be very useful. Furthermore, it was easily recognizable for the physicians where the *Current Time Indicator* and *Time Cursor* are placed and which value they represented. Moreover, the *Fisheye view* functionality was found to be useful by all but one interviewed doctors.

Another feature being pointed out positively was the marking of data points within diagrams.

All physicians found the *PlanningLine* glyph simple and easy to understand, especially by the means of the provided metaphor (see Section 5.3.3). Furthermore, the navigation within the plan hierarchy was considered to be intuitive.

On the negative side, the not optimal behavior of time shift and zoom functions of the time scale was pointed out as already mentioned above and the scale captions were found to be too imprecise.

One physician mentioned that he would prefer using the *PlanningLine* glyph with its caps for depicting plans in the past as well to provide a completely uniform representation.

Furthermore, the limited functionality of the provided diagrams was objected.

7.3.4 Interaction and Coupling of Views

Interaction with the tool was considered to be easy to understand especially due to its step-by-step nature without any "magic key press or click features".

Asked about the opinion of direct vs. independent navigation of the two views, the answers were twofold, but all asked physicians would like that to be user configurable.

7.3.5 Applicability and Future Involvement

Basically, all interviewed doctors can imagine to use a software like this for daily work in dealing with treatment plans. Of course they objected the missing features and small bugs of the prototype, but in general they were pleased about the outcome of this work.

Moreover, all physicians were interested in being involved in further development of such a tool. In particular they would appreciate having plans used in their specific work area as test data sets to be able to give better feedback on the usability and applicability issues.

7.3.6 Recommendations

Some physicians expressed recommendations on additional features for the developed tool apart from the ones already included in the conceptual design.

First of all, an online help was addressed in form of either a standard help function, explanations directly on the screen, or having a legend. Secondly, one physician desired to have a plan creation respectively editing possibility included in the tool.

Furthermore, the need for user interface state personalization was pointed out by another interviewed doctor. Due to the fact that such a tool would be used by more than one person, problems could arise from individual settings preferred by the particular persons. This might lead to confusion or irritation which could be avoided by allowing to save and recall user profiles. Another possibility would be having a default view setting that can be restored if one is lost in the actual view.

7.4 Bugs and Shortcomings

During testing and evaluation, several bugs and shortcomings were spotted:

- The tooltips showing plan titles do not work on *Windows*$^{\text{TM}}$.
- The white font for marking the selection of a plan is sometimes not readable (particularly in case of *PlanningLines*).
- Selection and deselection of plans show anomalies in some cases.
- Resizing of views does not work if the window is getting smaller.
- The arrow signaling that a plan is going beyond the time scale invites to interact, but is just signalizing that fact.

- Logical view elements do only provide single line captions.

7.5 Proposed Improvements

Possible changes and additions that were found out during testing and evaluation are proposed in the following:

- Interaction with plans and coupling of views should be more intuitive. This could be achieved via the following interaction pattern:
  - **click**: Selection of a plan.
  - **double-click**: Zooming to the width of the currently selected element within the *Temporal View* and expanding the selected plan within the *Logical View*.
  - **drag & drop**: Dragging & dropping elements between views propagates the dragged element to the target view.
- Direct vs. independent navigation of views should be user configurable.
- The *Time Cursor* should stay at the current position when scrolling the time scale.
- The sub-plans of a selected plan should be marked in case it is expanded rather than highlighting the summary line representing the plan.
- The position of the plan conditions should be user configurable.
- There should be a way to jump to the current time in the *Temporal View* (i.e. by clicking the clock value).
- The formatting of time scale captions should be improved.

7.6 Discussion

Conducting this prototype evaluation provided useful information about the appropriateness of our graphical representations along with their interaction methods. The feedback was very positive and we saw that our representations are clear, intuitive, and easy to understand. Furthermore, the strong and weak points of our prototype were brought up clearly by the users (and not by assumptions of ourselves) which enables us to improve them specifically.

An issue worth to be pointed out is the success of the mental model respectively metaphor used for explaining the *PlanningLine* glyph (see Section 5.3.3). The metaphor of encapsulated bars held by two caps, whereas the caps are fixed and the bars can be moved, was a very good mean for explaining the depicted attributes.

Most interesting was probably the increased interest towards the topic by the physicians in contrast to the first interview in the user study. Having more concrete concepts and a prototype to play around with using a real data set seemed to boost the interest a lot. Additionally, the fact that all interviewed doctors are interested in being involved in further development of this software confirms that impression.

Chapter 8

User Involvement

"New medical information systems, no matter how fast, inexpensive, and easy to use, will not be used more widely until it has been demonstrated to practitioners that these systems provide answers that help solve the problems of patient care."
[Gorman and Helfand, 1995]

The development approach we took for this work is called **User Centered Design** [Preece et al., 2002]. As the name implies, this means putting the user (her needs, tasks, imaginations, input, feedback) in the center of the development process and involving her from the very beginning.

We have involved potential users and domain as well as visualization experts in our project. The first step was gathering as much domain and user information as possible by conducting a *user study* (see Chapter 4). This lead to a better understanding of the medical domain in general as well as it was giving valuable insights on work practices, user tasks, and expectations from physicians.

The next step of user involvement was doing an evaluation of the conceptual design (see Section 5.6). For this purpose, two domain experts (one medical expert and one visualization expert) were asked to review the concept and give feedback on the design. This evaluation showed that we were working in the right direction and pointed out some minor weaknesses of the design we were able to correct quickly. We decided to do this step as *expert review* rather than by *user testing* because cost and effort are much lower and at the same time receiving a lot of valuable information is ensured by choosing the right experts. In our opinion, involving actual users is more appropriate having a running prototype. Furthermore, our user study showed that it is very hard to get appointments respectively time from physicians. And because our test users were the same as the user study participants, we did not want to stress this too much by asking them for an interview twice within three month.

Having programmed the prototype, we were again involving actual users by conducting a *prototype evaluation* (see Chapter 7). The response of our test users was very positive and provided important information on what issues are particularly crucial for practical use of such an application.

Figure 8.1 shows our development process graphically as a set of interconnected tasks around the central entity of design, the user. Points of information exchange with the user in the development are signalized by arrows to and from the user.

All in all, the *User Centered Design* approach turned out to be well suited and "keeping its promises". Involving users from the very beginning increases the quality of the development a lot by focussing onto real users' needs rather than speculations about them often ending in so-called "featurism" [Cooper, 1999].

User Centered Design seems to require more effort and higher cost at first sight, but it pays off by spotting problems early, at stages were correcting them is a lot easier and cheaper. Furthermore, applying the appropriate methods at the various stages of the development is important - it is not always the best thing to conduct a user study with 100 test persons.

Through carrying out this work we experienced that the user centered development approach led to solutions of higher quality.

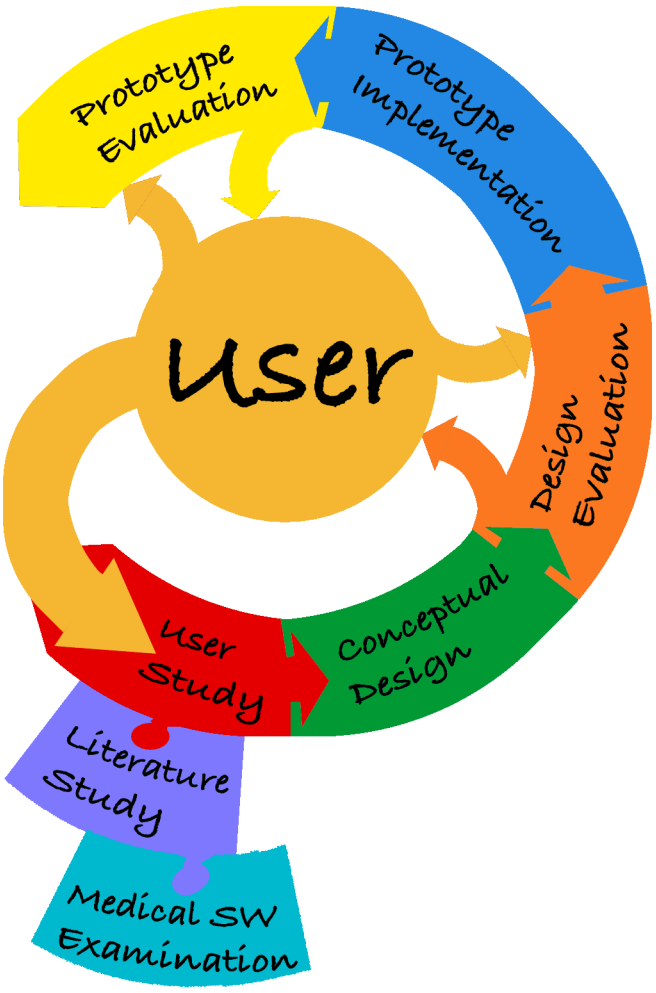


Figure 8.1: User Centered Design.

Chapter 9

Conclusion

At the beginning of this thesis we gave a brief introduction of protocol-based care in general and more specifically of the *Asgaard/Asbru* approach this work is part of. We worked out the requirements for our visualization determined by *Asbru* and listed the goals we wanted to achieve by the development of these techniques.

Next, we studied related literature in the fields of medical software development, Information Visualization, User Interface Design, and Usability. Furthermore, we examined graphical visualization methods used in commercial medical software products and presented visualizations included in related scientific projects. These tasks were performed in order to find out whether existing methods or tools fit our requirements.

In parallel to these steps we conducted a user study with eight physicians working in intensive care departments to gain deeper insights into the medical domain, work practices, application of guidelines in daily work, users' needs, expectations, and imaginations. The results of this study formed the basis for resembling a user model (typical user groups represented by fictitious *Personas*) along with *Scenarios* describing tasks and goals in analogy to the *Interaction Design* technique proposed by A. Cooper [Cooper, 1999]. These *Personas* and *Scenarios* have been used for cognitive walkthroughs of the design. Using them for evaluating the software prototype was intended as well but could only be done partially due to the limited functionality of the prototype.

Because none of the examined visualization methods fulfilled all the requirements defined, we framed out a new conceptual design. We took a **two-view approach** introducing a *Logical View* and a *Temporal View* for representing plans, parameters, and variables. For the *Logical View* we created a flowchart-like representation based on *clinical algorithms* [Society for Medical Decision Making, 1992]. For depicting plans within the *Temporal View*, we extended the *LifeLine* concept and developed a novel glyph called *PlanningLine*. A *PlanningLine* is able to represent incidents having temporal uncertainties. Moreover, this glyph is not limited to the medical domain, but can be used for a variety of applications.

The conceptual design has been evaluated by two experts (one medical expert and one visualization expert) afterwards and showed that we were working in the right direction. Only minor issues were addressed by the reviewers we were able to correct quickly.

During the next phase we implemented a software prototype using *Java* for proofing our concept and giving a better impression especially of interaction issues.

This prototype was used in the following evaluation, again performed by physicians dealing with critically ill patients. In the evaluation sessions we presented our concept, the implemented prototype, and asked for feedback. The evaluation showed that the graphical representations we developed are easy to understand and well suited for this field of application. We received mostly positive feedback and all interviewed physicians can imagine using a tool having these visualizations themselves for daily work.

In Section 2.5 we formulated goals to achieve by our work. We accomplished these goals by introducing our visualization methods which offer better analysis support in protocol-based care. Various information sources are integrated in our software, namely medical guidelines, their execution trace, patient data in form of measured parameters, and plan variables. These advantages help to concentrate on the essentials in the daily routine and improve therapy. Furthermore, recording this data provides a comprehensive treatment documentation.

9.1 Lessons Learned

Our work showed that Information Visualization techniques can contribute a lot to improve treatment planning software in the medical domain. A crucial issue when introducing new visualization methods or software is involving the user in the development from the very beginning.

We saw that difficulties concerning the introduction of information technology are mostly of non-technical nature, but rather social or organizational. Due to the tough working conditions, rejection of information technology happens quite frequently in the medical domain and can only be minimized by involving users and pointing out the benefits specifically for the individual user.

These issues have to be kept in mind in order to be successful when carrying out projects in this area.

Chapter 10

Future Work

Work that is left to do includes first of all solving the bugs of the software prototype found out during testing and evaluation. Furthermore, the prototype has to be improved by implementing the full set of features as defined in the *Conceptual Design* chapter respectively the missing features listed in Section 6.7. Proposed improvements that emerged during prototype evaluation like changing the view coupling interaction scheme should be considered as well.

Moreover, having a more mature prototype, user testing should be performed with a bigger focus towards *Usability* issues. Besides, such a prototype would allow cognitive walkthroughs utilizing the created *Personas* and *Scenarios* (see Sections 4.4 and 4.5).

Cooperation with physicians could be improved mostly by using example data sets of the specific work area a doctor is working in. Thus, selected guidelines need to be modeled in *Asbru* and specific parameter sets have to be provided.

Appendix A

Basic Terminology and Concepts

Note:
The following definitions are taken literally from the sources specified right next to the explained term.

A.1 Information Visualization, Human-Computer Interaction, and Usability

Cognitive Walkthrough [Usability First, 2003]

An approach to evaluating a user interface based on stepping through common tasks that a user would need to perform and evaluating the user's ability to perform each step. This approach is intended especially to help understand the usability of a system for first-time or infrequent users, that is, for users in an exploratory learning mode.

Based on a user's goals, a group of evaluators steps through tasks, evaluating at each step how difficult it is for the user to identify and operate the interface element most relevant to their current subgoal and how clearly the system provides feedback to that action.

Domain Expert [Usability First, 2003]

Or "subject matter expert" (SME); a person with special knowledge or skills in a particular area; a person extremely familiar with a given group of users and their work habits (because they belong to the group).

A truck driver is an expert in trucking and the needs truckers have for the types of systems that truckers might use, such as delivery or inventory systems. A manager may be a domain expert for project management systems. An accountant is a domain expert for accounting systems.

Programmers and user interface designers need to interview domain experts and work with domain experts in the design of systems for the expert's field.

Focus + Context [Usability First, 2003]

A principle of **Information Visualization** – display the most important data at the focal point at full size and detail, and display the area around the focal point (the context) to help make sense of how the important information relates to the entire data structure. Regions far from the focal point may be displayed smaller (as in **Fisheye Views**) or selectively.

Glyph [Keller and Keller, 1993]

An object or symbol for representing data values. Glyphs are generally a way of representing many data values and are sometimes called **icons**. A common glyph is the arrow, often chosen to represent vector fields. The arrow depicts both speed and direction at a point.

GUI [Usability First, 2003]

Graphical **U**ser **I**nterface, pronounced "GOOEY". A user interface that presents information graphically, typically with draggable windows, buttons, and icons, as opposed to a textual user interface, where information is presented on a text-based screen and commands are all typed.

HCI [Usability First, 2003]

Human-**C**omputer **I**nteraction. The study of how people work with computers and how computers can be designed to help people effectively use them.

Icon [Usability First, 2003]

In computer terminology (as opposed to graphic design terminology), an icon is a small image used most often to represent files or label a button. Much discussion goes into how icons should be designed: the use of line, color, and shading; how to effectively use **symbols** or representational images, and how to design families of icons.

Information Visualization (InfoVis) [Usability First, 2003]

The study of how to effectively present information visually. Much of the work in this field focuses on creating innovative graphical displays for complicated datasets, such as census results, scientific data, and databases. An example problem would be deciding how to display the pages on a website or the files on a hard disk. Visualization techniques include selective hiding of data, layering data, taking advantage of 3-dimensional space, using scaling techniques to provide more space for more important information (e.g. **Fisheye views**), and taking advantage of psychological principles of layout, such as proximity, alignment, and shared visual properties (e.g. color).

Interaction Design [Usability First, 2003]

The design of how a user communicates, or interacts, with a computer. Interaction designers focus on the flow of interaction, the dialog between person and computer, how input relates to output, stimulus-response compatibility, and feedback mechanisms.

This is in contrast to a visual designer, who may be trained in designing visualizations for static media but not necessarily in the dialog which is present in all interactive media. A "visual interaction designer" is a visual designer with interaction design skills. Interaction design is also in contrast to information architecture - an information architect looks at the organization of information to make the structure of a complex system easy to conceptualize and navigate, but is not usually focused, for instance, on low-level interactions. For example, an information architect may design the structure of an entire website, but not have as much interest in the design of individual pages and how users interact with forms and other controls.

Metaphor [Usability First, 2003]

The use of one idea or object to represent another; making an implicit comparison between concepts to provide insight into those concepts.

Metaphor is used widely in graphical user interfaces to help set users' expectations and make the behavior of computers clearer. The "desktop metaphor" is used to suggest that a computer screen is like a physical desk, with papers and folders to shuffle around and various desk accessories, such as calculators, printers, and notepads. A general physical world metaphor is what allows a beveled border to suggest a button and allows close parallel lines to suggest that something is draggable.

Metaphors are also useful techniques for designers to explore representations of concepts and the behavior of interface elements. Designers may also apply wild and unrelated metaphors as a useful brainstorming device.

Mockup [Usability First, 2003]

Another term for prototypes, usually referring to low-fidelity prototypes, such as paper illustrations, screenshots, or simple configurations of screens with limited interaction.

Paper-and-Pencil Prototype [Usability First, 2003]

A paper sketch of a user interface with enough detail to make design decisions and **usability** evaluations, whether through a **usability** inspection, a focus group, or a simple user test.

Overview + Detail Two images are used for presentation. One shows a rough overview of the complete information space and neglects details. The other one shows a small portion of the information space and visualizes details. Both images are either shown sequentially or in parallel.

Persona [Usability First, 2003]

A description of a specific person who is a target user of a system being designed, providing demographic information, needs, preferences, biographical information, and a photo or illustration. Typically, multiple personas are developed in the early stages of design that represent the spectrum of the target audience. Personas are one piece of a "**Scenario"**, the other piece being a description of how this person would typically interact with the system being designed.

The point of developing personas is to avoid the trap of designing for the "average" user that doesn't actually exist, and instead to make sure that the system will work for somebody specific rather than no one in particular.

Prototyping [Usability First, 2003]

The development of incomplete representations of a target system for testing purposes and as a way of understanding the difficulties of development and the scale of the problem.

Prototyping is an essential element of an iterative design approach, where designs are created, evaluated, and refined until the desired performance or **usability** is achieved. Prototypes can range from extremely simple sketches (low-fidelity prototypes) to full systems that contain nearly all the functionality of the final system (high-fidelity prototypes).

Scenario [Usability First, 2003]

A design envisionment technique whereby a set of representative target users are identified and an outline is created of their lives, their goals, their interests, their schedules, and their interaction with the system being designed. While these descriptions of different target populations may be driven by survey and interview data, for instance, it is primarily a conceptual, introspective technique. Scenarios are often created quite early in a design cycle to help identify requirements and, like task analysis, to identify necessary features that might otherwise have been overlooked. Scenarios make sure that you have a specific idea of who the product is targeted at and that you have considered the different types of users and how their needs and goals may be different.

Scientific Visualization [Usability First, 2003]

The graphical representation of complex physical phenomena in order to assist scientific investigation and to make inferences that aren't apparent in numerical form. Typical examples include processing of satellite photographs and 3D representations of molecules and fluids to examine their dynamics.

Symbol [Duersteler, 2003]

Image, figure or object that represents an abstract, moral or intellectual concept.

The symbol has to be distinguished from the sign. A symbol implies more than its immediate meaning. Sometimes even the concept represented can be different depending who is considering it. A flag is a clear example of a symbol. As a sign a flag can point to its particular nation or state. As a symbol it represents an ensemble of people and institutions, emotions and non rational feeling in some ambiguous way.

The study of symbols is called symbology.

Task Analysis [Usability First, 2003]

A set of methods for decomposing people's tasks in order to understand the procedures better and to help provide computer support for those tasks. The basic approach is to define the task and the goal of the task and then to list the steps

involved. The level of detail in decomposing the steps is determined by how the analysis is going to be used.

Task analyses are useful for making time predictions for how long a task will take and for spotting potential errors (steps in the process which are extremely difficult or confusing). Task analyses are also good for spotting areas in a user interface that may have been overlooked or oversimplified. A task analysis is a fundamental part of a cognitive model of user performance, such as GOMS (goals, operators, methods, and selection rules. See [Preece et al., 2002]).

Usability [Usability First, 2003]

The characteristic of being easy to use, usually applied to software, but relevant to almost any human artifact. What makes an artifact easy to use? Broadly, something is easy to use to the extent that it effectively performs the task for which it is being used. Ease of use can be measured by how quickly a task is performed, how many mistakes are made, how quickly the system is learned and how satisfied people are who perform the task. Usability may also include factors such as safety, usefulness, and cost-effectiveness.

User Interface [Usability First, 2003]

The parts of a computer system that a person uses to communicate with the computer. This includes the way the computer conveys messages to the person (output devices), the way the person talks to the computer (input devices), and the steps the person must perform to do their task.

User Interface Design [Usability First, 2003]

The overall process of designing how a user will be able to interact with a software application.

User Interface Design is involved in many stages of product development, including: requirements analysis, information architecture, interaction design, screen design, user testing, documentation, and help system design. User interface designers may require skills in many areas, including: graphic design, information design, software engineering, cognitive modeling, technical writing, and a wide variety of data collection and testing techniques.

User Studies [Usability First, 2003]

Any of the wide variety of methods for understanding the **usability** of a system based on examining actual users or other people who are representative of the target user population. Such methods include **user testing**, focus groups, surveys, interviews, observational studies and ethnographic methods, and diary studies.

User Testing [Usability First, 2003]

A family of methods for evaluating a user interface by collecting data from people actually using the system.

A simple user test would be to bring in a small number of potential users of the software (perhaps 8 to 10) and have each person sit down and use the software to perform a series of tasks while an observer takes notes about what difficulties

each user encounters. Typically, users are asked to think out loud while they work with the software to help the observers understand how the users think about their problems and how the interface could be improved.

More involved user testing may test more users, get as representative a selection of users as possible, try out a variety of tasks, control the testing environment in various ways (or test a more naturalistic work environment), use more careful or thorough measurement instruments (videotaping, recording keystrokes, etc.), or combine the testing with other methods of data collection, such as interviews of users.

A.2 Medical Guidelines & Protocols

Guideline [Roomans, 2001]

Most commonly, the term *guideline* is used as an abbreviation of the term **Practice Guideline**, or the full term **Clinical Practice Guideline**. *"Clinical practice guidelines are systematically developed statements to assist practitioner and patient decisions about appropriate health care for specific clinical circumstances."* [Field M., 1990]

The term **protocol** is often used equivalent with **guideline**, though guidelines tend to be more declarative whereas protocols tend to be more procedural. The procedural content of protocols tends to be more detailed, which is probably the reason why some see protocols as a more detailed version of a guideline [Miksch, 1999]. There is no consensus on the meaning differences of both terms.

Protocol [Miksch, 1999]

Or **clinical protocol** is a more detailed version of a **clinical guideline** and refers to a certain class of therapeutic interventions. Protocols are used for utilization review, for improving quality assurance, for reducing variation in clinical practice, for guiding data collection, for better interpretation and management of the patient's status, for activating alerts and reminders, for improving decision support [Pattison-Gordon et al., 1996].

Appendix B

Interview Guideline

B.1 Original German Version

Vorstellung des Asgaard Projekts

Kurzpräsentation

Kenntnisse & Erfahrung (Computer, Domäne, System)

Computer

Allgemein

- Beurteilung der eigenen Computerkenntnisse (Anfänger, Fortgeschrittener, Experte - evt. Skala von 1-5).

Anfänger	1	2	3	4	5	*Experte*

- Beruflich?
  - welche Aufgaben?
  - Welche SW wird hauptsächlich verwendet?
  - wie oft?
- Privat?
  - besitzt selbst einen Computer (Notebook), Handheld, Handy, Spielkonsole, Gameboy?
  - welche Aufgaben?
  - Welche SW wird hauptsächlich verwendet?
  - wie oft?
- Welche Betriebssysteme? (Windows, Mac, Linux, Unix)
- Erfahrung mit Projektmanagement SW / Techniken?

Bekannte Diagrammformen

- Bekannt?, Beurteilung?
  - Datenvisualisierung
    - Liniendiagramm
    - Balkendiagramm
    - Tortendiagramm
  - Ablaufvisualisierung
    - Flow-Chart
    - Struktogramm
    - PERT Diagramm
    - GANTT Diagramm
    - AsbruView
    - LifeLines
    - Overviewtool
  - Verwendung von Piktogrammen?
- Sonstige Diagramme die oft verwendet werden bzw. gängig sind?

Medizin

Allgemein

- Position/Aufgabengebiet
- Laufbahn/Erfahrungen

Therapiepläne

- Kenntnisse/Umgang mit Therapieplänen
- Welche Darstellungsformen werden dort verwendet?
  - Text
  - Tabellen
  - Diagramme
  - sonstiges
- Ablauf von der Diagnose über Therapiewahl zu Therapiedurchführung?
- Welche Daten werden ständig beobachtet? / Gibt es Daten, die unabhängig vom vorliegenden Krankheitsbild allgemein immer wichtig sind und dargestellt werden sollten?
- evt. Kopien von verwendeten Dokumentationsbögen, Formularen, die verwendet werden?

Therapieplantool

- Schon mal davon gehört?
- Vorstellung einer Computerunterstützung?
- Was halten sie davon
  - grundsätzlich sinnvoll, gut, schlecht, überflüssig
- schon ein ähnliches System verwendet?
- Zufriedenheit mit diesen Systemen?, Erfahrungen (positiv, negativ)

Typische Systemnutzung, Ziele, Aufgaben, Wünsche (Goal + Task Analyse)

Allgemein

- Würde man ein solches System nutzen?
- Welche Aufgaben würde man gerne erledigen?
- Wie stellt man sich den Umgang mit dem System in der täglichen Praxis vor?

Konkretere Vorstellung

- WAS ist wichtig? (allg. Charakteristik)
- Was muss unbedingt dargestellt werden?
- Was ist wünschenswert / eher zweitrangig?

Bedienung, Ablauf

- Gibt es eine konkretere Vorstellung zum Aussehen / Bedienung eines solchen Tools
  - Vergleich zu bekannter SW?
- Vorstellung, WIE man damit arbeiten würde?
  - Stationär, Hauptrechner auf der Station
  - Laptop, Tablet PC?
- Bedienung mit Tastatur, Maus, Stift,...?

Planvisualisierung

- Welche der bekannten / vorgestellten Visualisierungsformen könnten hier zur Anwendung kommen?

Charakteristik

- Farbe?
- Automarke, -typ?
- System als Haus beschreiben?

Weiteres

- Bereit zur Evaluation des Prototypen?
- Kennt andere Kollegen, die möglicherweise für ein Interview bereit wären?

B.2 English Translation

Presentation of the Asgaard project

Short presentation

Knowledge & Experience (Computer, Domain, System)

Computer

General

- Judgement of own computer knowledge (Beginner, Advanced, Expert - maybe scale from 1-5).

Beginner	1	2	3	4	5	*Expert*

- Job-related?
  - Which tasks?
  - What software is being used mainly?
  - How often?
- Private?
  - Owns a computer (Notebook, Handheld, Handy, Game-console, Gameboy)?
  - Which Tasks?
  - What software is being used mainly?
  - How often?
- Which operating systems? (Windows, Mac, Linux, Unix)
- Experience with project management sofware / techniques?

Known diagram types

- Known?, Judgement?
  - Data visualization
    - Line diagram
    - Bar chart
    - Pie chart
  - Execution sequence visualization
    - Flow chart
    - Structogramm
    - PERT chart
    - GANTT chart
    - AsbruView
    - LifeLines
    - Overviewtool
  - Use of pictograms?
- Other diagrams that are used frequently respectively well known?

Medicine

General

- Position/Responsibilities
- Career/Experiences

Therapy plans

- Knowledge/use of therapy plans
- What forms of visualizations are used?
  - Text
  - Tables
  - Diagrams
  - Others
- Workflow from diagnosis to therapy selection and therapy execution?
- Which data is always used? / Are there data items that are generally important and should be displayed always independently of the particular case?
- Copies of documentary sheets or forms that are getting used?

Therapy planning tool

- Already heard about it?
- Imagination of computer support?
- Opinion about it?
  - basically useful, good, bad, dispensable
- Already used a similar system?
- Satisfaction with these systems?, Experiences (positive, negative)

Typical System Use, Goals, Tasks, Desires (Goal + Task Analysis)

General

- Would such a system be used?
- What tasks would be accomplished?
- Imagination of the interaction with the system in daily work?

Concrete Image

- WHAT is important? (general characteristic)
- What has to be displayed necessarily?
- What is desirable / rather secondary?

Interaction, Workflow

- Is there a concrete image of the appearance / handling of such a tool?
  - Comparison to familiar software?
- Image, HOW it would be used?
  - Stationary, workstation at the department
  - Laptop, Tablet PC?
- Control via keyboard, mouse, stylus,...?

Plan Visualization

- Which of the known / introduced visualization techniques could be used?

Charakteristic

- Color?
- Car brand / type?
- System as a house?

Others

- Wants to participate at evaluation of the prototype?
- Knows other collegues who would maybe participate in the user study?

B.3 Diagram Examples

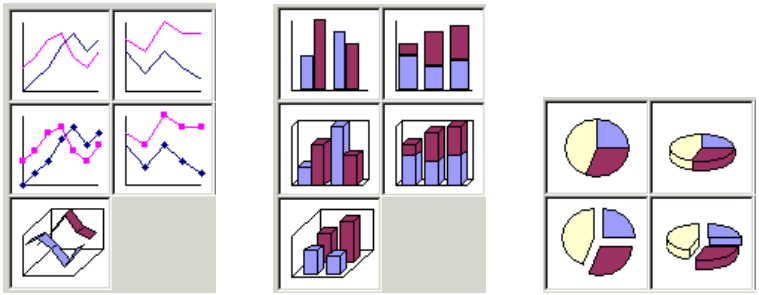


Figure B.1: Standard diagrams.

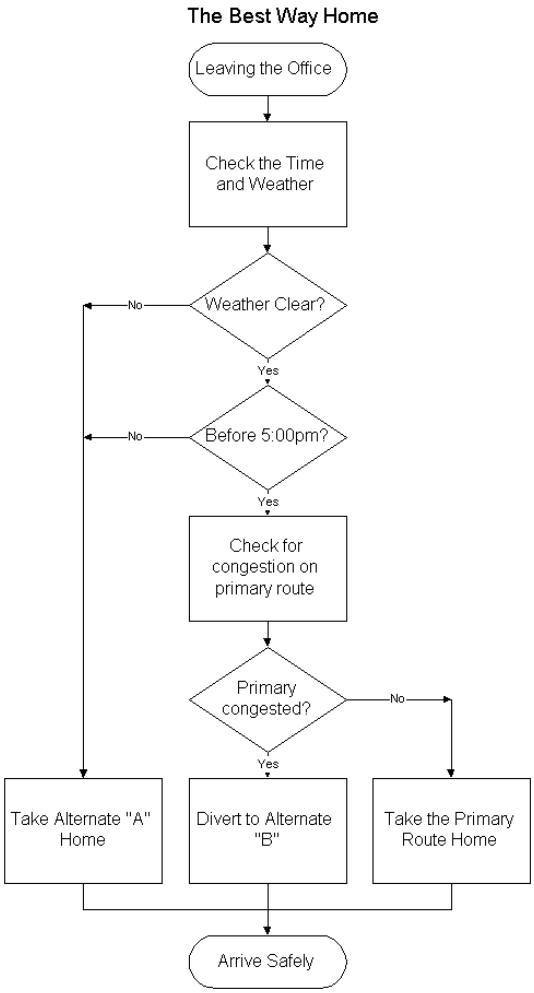


Figure B.2: Flow chart.

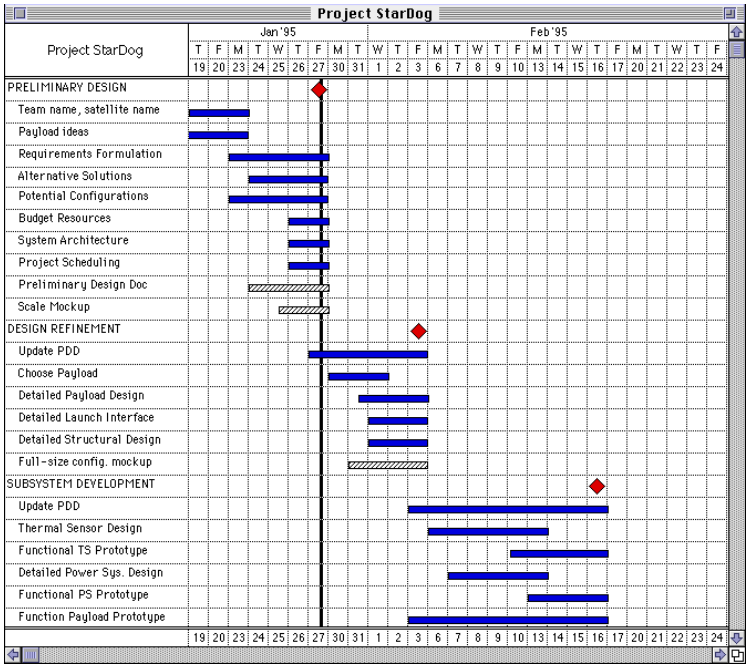


Figure B.3: GANTT chart.

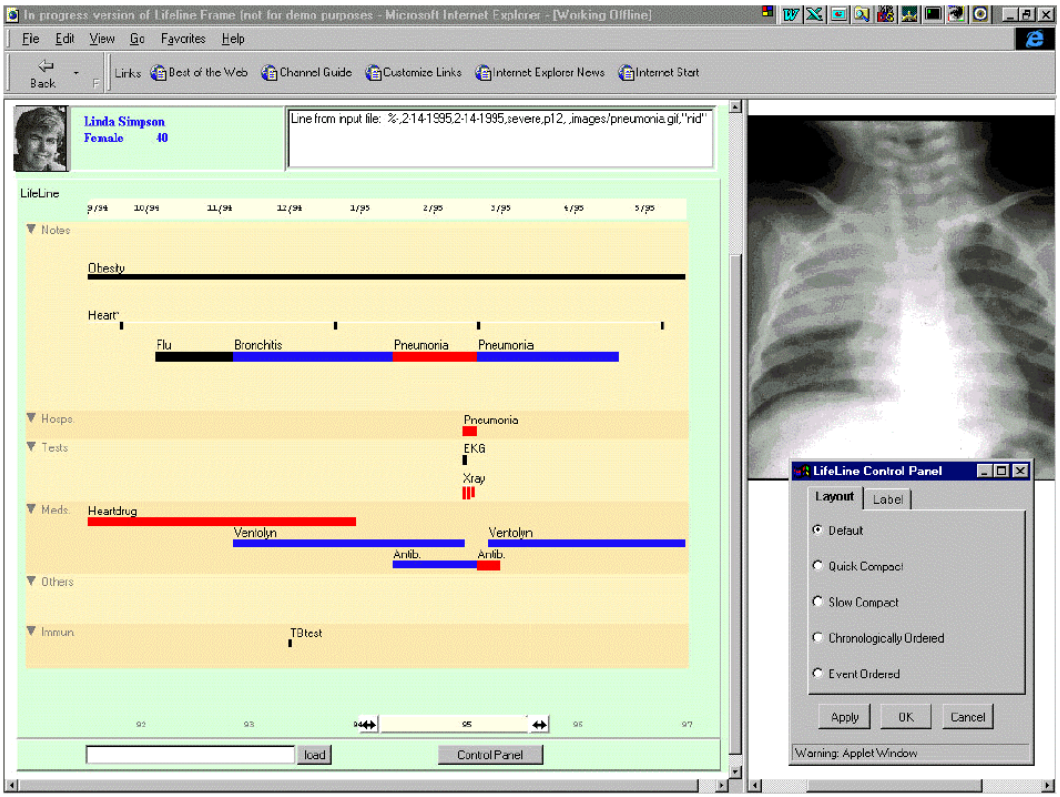


Figure B.4: LifeLines [Plaisant et al., 1998].

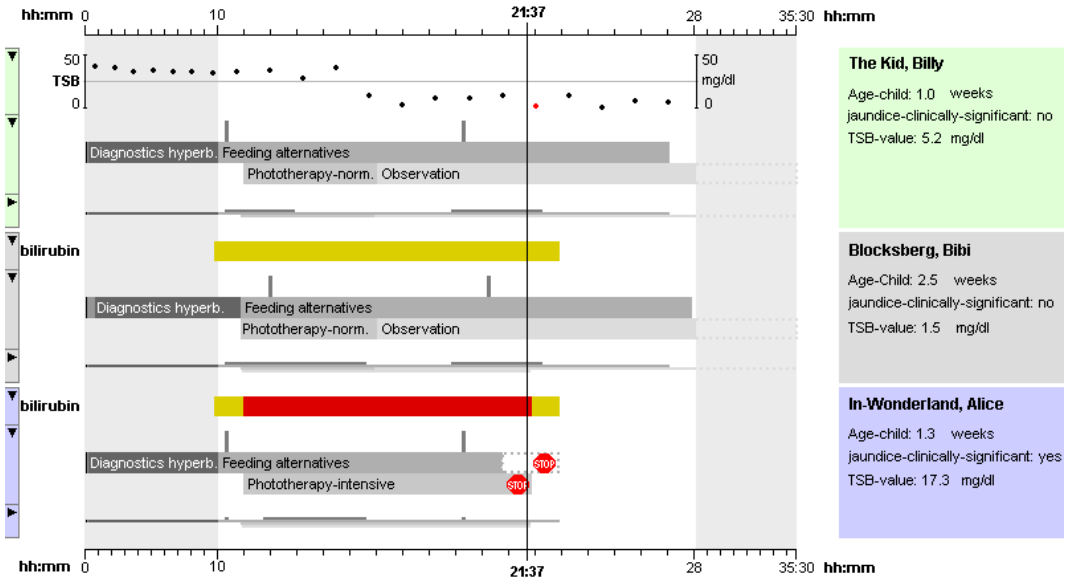


Figure B.5: Guideline Overview Tool (GOT) [Aigner, 2001].

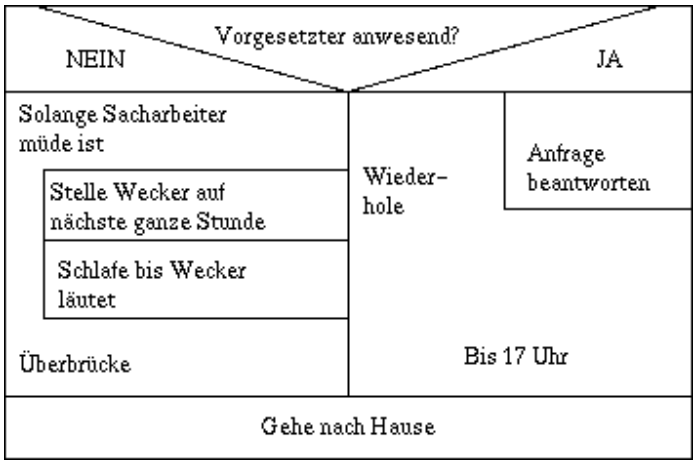


Figure B.6: Structogram.

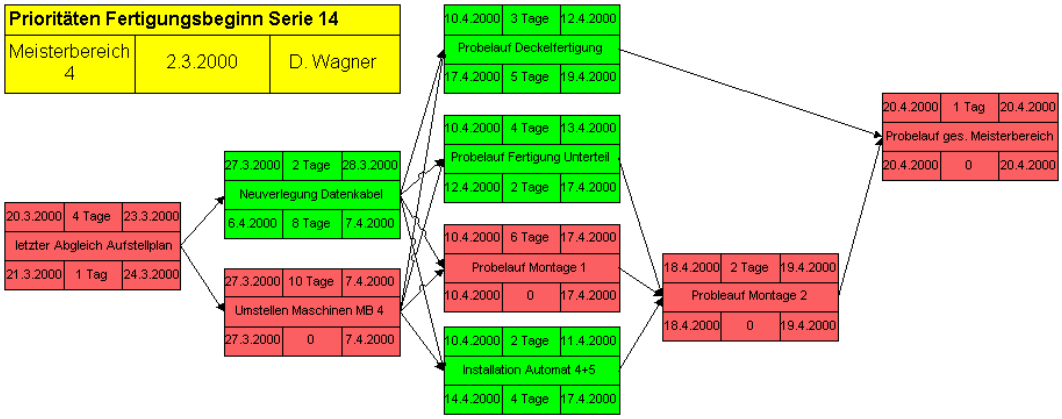


Figure B.7: PERT chart.

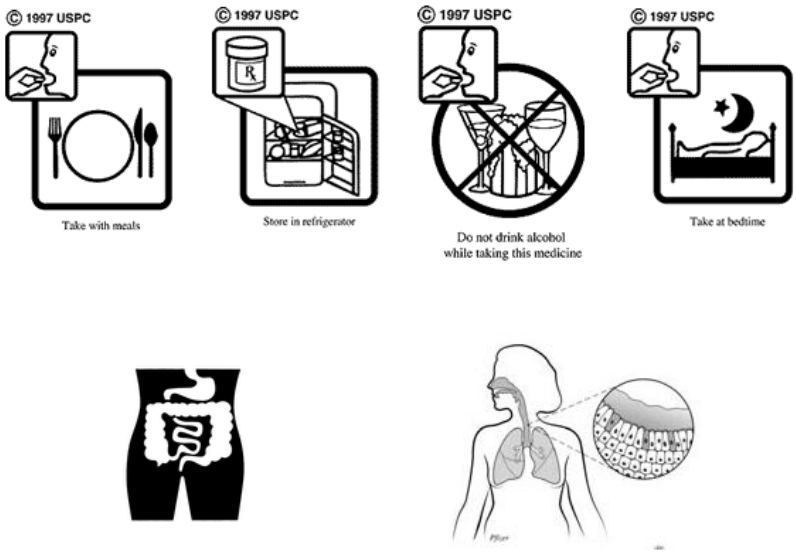
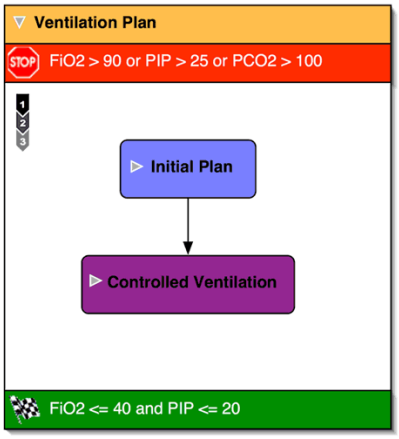


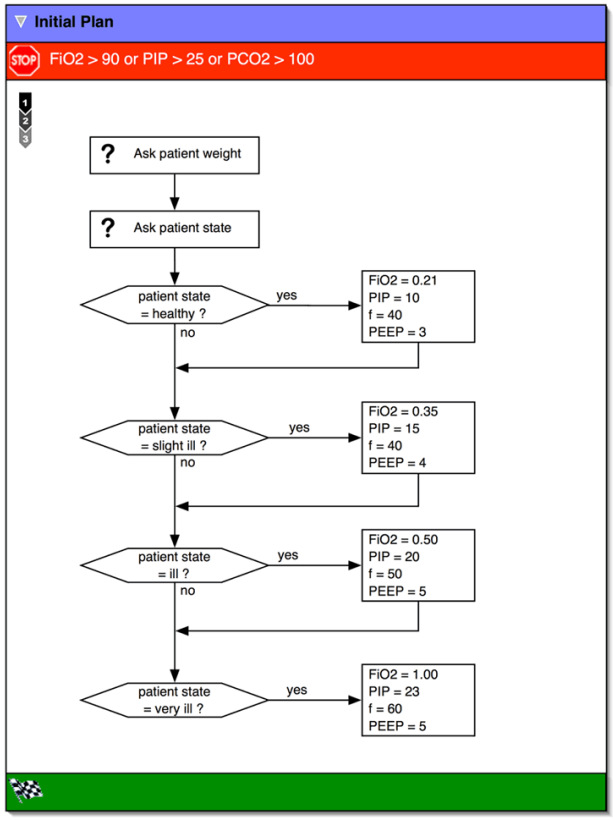
Figure B.8: Pictograms.

Appendix C

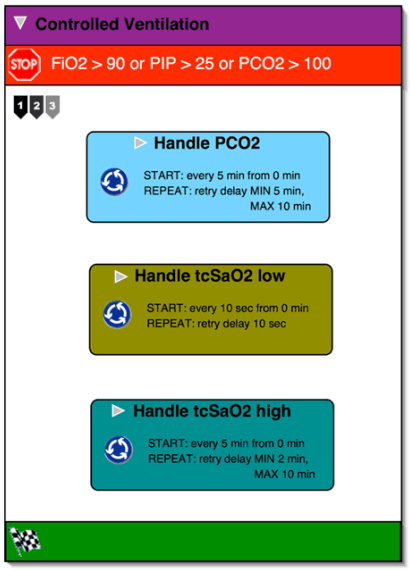
Logical View Example



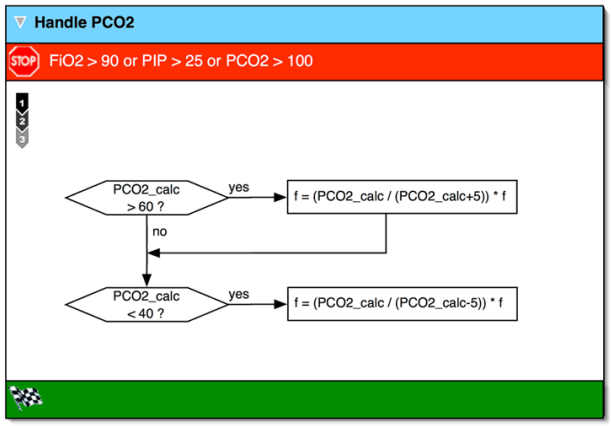
(a) Ventilation Plan.



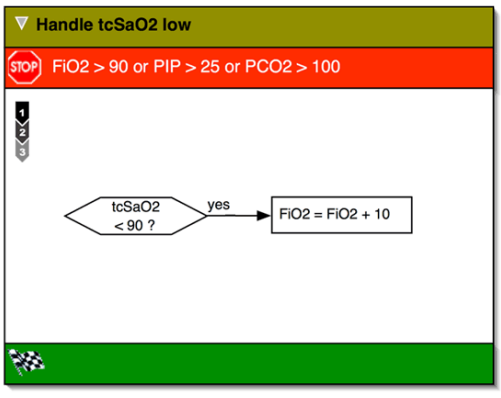
(b) Initial Plan.



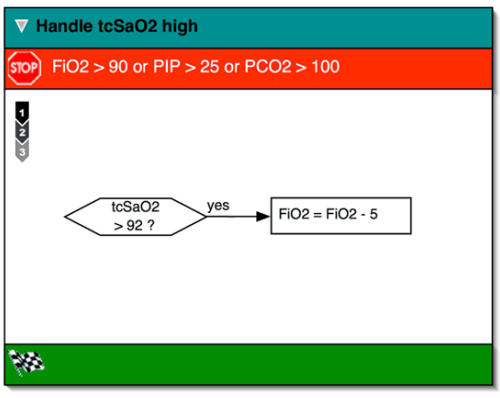
(c) Controlled Ventilation Plan.



(d) Handle PCO2 Plan.



(e) Handle tcSaO2 low Plan.



(f) Handle tcSaO2 high Plan.

Figure C.1: Artificial Ventilation of Neonates

Appendix D

Prototype Evaluation Form

D.1 Original German Version

Einleitung und allgemeine Bemerkungen

Aufgabenstellung

Entwicklung eines Tools, dass die protokoll- bzw. guidelinebasierte Behandlung sowohl während der Behandlung, als auch zur Analyse unterstützt.

Vorgehensweise bisher

Um die oben genannten Anforderungen zu erfüllen, wurden auf Grundlage der Ergebnisse der durchgeführten Interviews und Untersuchung bestehender Lösungen folgende, visuelle Darstellungsformen herausgearbeitet:

- logische Ansicht basierend auf "klinischen Algorithmen"
- zeitbezogene Ansicht basierend auf "LifeLines"

Daraufhin wurde ein Prototyp entwickelt, der die wichtigsten Merkmale und Eigenschaften des Entwurfes enthält. Dies ist ein erster Prototyp, der die Umsetzbarkeit des Entwurfes prüfen soll und einen ersten Eindruck darüber vermitteln soll, wie ein solches Tool aussehen könnte. Das Stadium des Prototypen ist daher auch noch sehr weit von einem verwendbaren Endprodukt entfernt und sollte deshalb auch aus diesem Blickwinkel betrachtet werden.

Ich würde ihnen gern diesen Prototypen vorführen, auch selbst ausprobieren lassen und ihnen im Anschluss daran einige Fragen stellen.

Bemerkungen zur Evaluation

- kein Test für den Befragten
- kann jederzeit aufhören
- Ziel ist es, den Entwurf bzw. den Prototypen zu evaluieren, d.h.
  - Schwachstellen und Unzulänglichkeiten herauszufinden

- Schwer bzw. nicht Verständliches aufzuzeigen
- Positives als auch Negatives aus Benutzersicht zu sammeln

Fragebogen

Prototyp allgemein

- Was finden sie gut am Prototypen?
- Was finden sie schlecht am Prototypen?
- Was finden sie verwirrend oder schwierig?
- Was halten sie von der Verwendung zweier verschiedener Ansichten (logisch + zeitbezogen)?
- Was würden sie vorschlagen, um den Prototypen zu verbessern?

Logische Ansicht

- Was finden sie gut an der logischen Ansicht?
- Was finden sie schlecht an der logischen Ansicht?
- Sind die verwendeten Symbole für die Planbedingungen für sie verständlich?

ohne Erklärung sofort verständlich	nach Erklärung klar	Bedeutung nicht klar erkennbar / verwirrend / mehrdeutig	gänzlich unverständlich

- Sind die verwendeten Symbole für die Ausführungsreihenfolge für sie verständlich?

ohne Erklärung sofort verständlich	nach Erklärung klar	Bedeutung nicht klar erkennbar / verwirrend / mehrdeutig	gänzlich unverständlich

- Sind die innerhalb des Planes verwendeten Symbole verständlich für sie?

ohne Erklärung sofort verständlich	nach Erklärung klar	Bedeutung nicht klar erkennbar / verwirrend / mehrdeutig	gänzlich unverständlich

- Was halten sie von der Platzierung der Planbedingungen?

Platzierung ok	beide Bedinungen sollten oben stehen	beide Bedinungen sollten unten stehen	gänzlich andere Darstellung wäre besser

- Wie finden sie die verwendeten Farben und Schriften in der logischen Ansicht?

Farben gut gewählt	zu hell	zu dunkel	nicht aufeinander abgestimmt
Farben legen eine andere Bedeutung nahe	Schrift zu gross	Schrift zu klein	Schrift zu hell
Schrift zu dunkel	Schriftart unpassend		

- Ist die Navigation in der logischen Ansicht intuitiv für sie?

intuitiv und sofort klar	nach Erklärung klar	ist verwirrend und unklar	gänzlich unbrauchbar

- Was würden sie vorschlagen, um die logische Ansicht zu verbessern?

Zeitbezogene Ansicht

- Was finden sie gut an der zeitbezogenen Ansicht?
- Was finden sie schlecht an der zeitbezogenen Ansicht?
- Was halten sie von den verwendeten "PlanningLines" für die Darstellung von, mit zeitlichen Unsicherheiten behafteten, Vorgängen?
- Was halten sie von der Fisheye Ansicht?

sehr sinnvoll	unnötig	unverständlich

- Ist die Bedeutung der einzelnen Teile der Darstellung klar für sie?

ja	größtenteils	teilweise	völlig unklar

- Wie beurteilen sie die verwendete Zeitskala?
- Sind die gebotenen Manipulationsmöglichkeiten für die Zeitskala (Zoom, Verschieben, Einstellen der Randwerte) ausreichend?

ausreichend	zu umfangreich	zu wenig	völlig unklar

- Was halten sie vom verwendeten Zeitcursor und dessen Interaktionsmöglichkeit?

ausreichend	zu umfangreich	zu wenig	völlig unklar

- Wie finden sie die Navigation (in der Hierarchie) der zeitbezogenen Ansicht?

intuitiv und sofort klar	nach Erklärung klar	ist verwirrend und unklar	gänzlich unbrauchbar

- Was halten sie von der Verwendung von Zusammenfassungslinien beim Ausklappen von Plänen?

sehr gut und sofort klar	nach Erklärung klar	ist verwirrend und unklar
gänzlich unbrauchbar	andere Darstellung wäre besser	

- Wie finden sie die verwendeten Farben und Schriften in der zeitbezogenen Ansicht?

Farben gut gewählt	zu hell	zu dunkel	nicht aufeinander abgestimmt
Farben legen eine andere Bedeutung nahe	Schrift zu gross	Schrift zu klein	Schrift zu hell
Schrift zu dunkel	Schriftart unpassend		

- Wie beurteilen sie die Diagramme zum Anzeigen von Parametern und Variablen?
- Ist auf den ersten Blick klar ersichtlich, wo man sich auf der Zeitachse befindet?

ja	nein

- Ist klar erkennbar, wo genau (auf welchem Zeitpunkt) sich der Zeitcursor befindet?

ja	nein

- Ist die Markierung (Einfärben) der zum Cursorzeitpunkt gültigen Werte sinnvoll?

ja	nein	nicht in der jetzigen Form

- Was würden sie vorschlagen, um die zeitbezogene Ansicht zu verbessern?

Kopplung und Interaktion

- Ist die gebotene Kopplung zwischen logischer und zeitbezogener Ansicht ausreichend?

ja	nein

- Finden sie es gut, dass die beiden Ansichten eine getrennte Navigation bieten und nur auf Doppelklick hin die aktuelle Selektion zur Nachbaransicht propagiert wird? (Oder wäre es ihnen lieber, wenn die beiden Ansichten automatisch enger gekoppelt wären?)

getrennte Navigationsmöglichkeit ist besser	direkte Kopplung wäre besser

Verwendbarkeit

- Könnten sie sich vorstellen, ein Programm wie dieses selbst einzusetzen?

ja	nein

- Falls nein, warum nicht?
- Was müsste ihrer Meinung nach unbedingt noch hinzugefügt bzw. verbessert werden, damit sie es einsetzen würden?

Weitere Entwicklung

- Sind sie daran interessiert, in die Weiterentwicklung eines solchen Tools miteinbezogen zu werden?

ja	nein

- In welcher Form wäre das für sie denkbar?

D.2 English Translation

Introduction

Problem Description

Development of a tool to support protocol- respectively guideline-based care during treatment as well as for analysis.

Steps Performed

To fulfill the requirements stated above, visual representations on basis of the results of the conducted interviews and examination of existing solutions have been created:

- Logical View based on "clinical algorithms"
- Temporal View based on "LifeLines"

Next, a prototype has been implemented, demonstrating the main characteristics of the design concept. This is a first prototype, built to proof the feasibility of the design and give a first impression about how such a tool could look like. Thus, the stage of the prototype is far from a usable end product and should be considered from that point of view.

I would like to show you the prototype, let you try out the tool yourself and ask you a few questions afterwards.

Remarks

- no test for the interviewee
- can stop any time
- goal is to evaluate the design respectively prototype, which means to
  - find out weak points and shortcomings
  - point out hard respectively not understandable things
  - positive as well as negative issues from a user's point of view

Questionnaire

Prototype in general

- What do you like about the prototype?
- What do you not like about the prototype?
- What do you consider irritating or hard to understand?
- What do you think about the use of two different views (logical + temporal)?
- What would you suggest in order to improve the prototype?

Logical View

- What do you like about the Logical View?
- What do you not like about the Logical View?
- Are the symbols used for plan conditions comprehensible for you?

comprehensible without explanation	clear after explanation	meaning not clearly graspable / irritating / ambiguous	completely incomprehensible

- Are the symbols used for the plan execution sequence comprehensible for you?

comprehensible without explanation	clear after explanation	meaning not clearly graspable / irritating / ambiguous	completely incomprehensible

- Are the symbols used within plans comprehensible for you?

comprehensible without explanation	clear after explanation	meaning not clearly graspable / irritating / ambiguous	completely incomprehensible

- What do you think about the placement of plan conditions?

placement ok	both conditions should be on top	both conditions should be on bottom	completely different representation would be better

- What do you think about the used colors and fonts in the Logical View?

colors well chosen	too bright	too dark	not harmonic
colors imply a different meaning	fonts too big	fonts too small	fonts too bright
fonts too dark	font not suited		

- Is the navigation within the Logical View intuitive for you?

intuitive and immediately evident	clear after explanation	is irritating and ambiguous	completely unusable

- What would you suggest in order to improve the Logical View?

Temporal View

- What do you like about the Temporal View?
- What do you not like about the Temporal View?
- What do you think about the used "PlanningLines" for depicting time-oriented processes afflicted with temporal uncertainties?
- What do you think about the Fisheye view?

very good	unnecessary	incomprehensible

- Is the meaning of the elements of the representation comprehensible for you?

yes	mostly	partly	completely incomprehensible

- What do you think about the used time scale?
- Are the provided interaction possibilities sufficient for the time scale (zoom, shift, manipulation of border values)?

sufficient	too extensive	too less	completely incomprehensible

- What do you think about the used time cursor and its interaction possibilities?

sufficient	too extensive	too less	completely incomprehensible

- What do you think about the navigation (within the hierarchy) of the Temporal View?

intuitive and immediately evident	clear after explanation	is irritating and ambiguous	completely unusable

- What do you think about the used "summary lines" when plans are expanded?

very good and immediately comprehensible	comprehensible after explanation	irritating and ambiguous
completely unusable	completely different representation would be better	

- What do you think about the used colors and fonts within the Temporal View?

colors well chosen	too bright	too dark	not harmonic
colors imply a different meaning	fonts too big	fonts too small	fonts too bright
fonts too dark	font not suited		

- What do you think about the diagrams for depicting parameters and variables?
- Is the current position on the time scale immediately graspable?

yes	no

- Is the time cursor value immediately graspable?

yes	no

- Is marking the values of a diagram valid at the time cursor's time reasonable?

yes	no	not in the current way

- What would you suggest in order to improve the Temporal View?

Coupling and Interaction

- Is the provided coupling between Logical and Temporal View sufficient?

yes	no

- Do you like separate navigation within the two views and propagating the current selection in case of double-click? (Or would would you prefer closer, automatic coupling of views?)

separate navigation is better	direct coupling would be better

Applicability

- Can you imagine using a tool like this yourself?

  yes no

- If no, why not?
- What would have to be added or improved necessarily to make it applicable for you?

Further Development

- Are you interested in being involved in future development of such a tool?

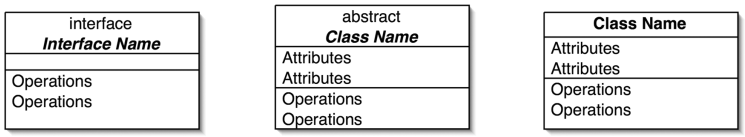
  yes no

- In which form would that be possible for you?

Appendix E

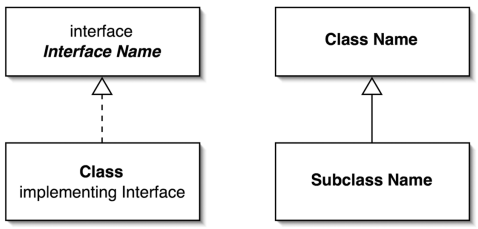
UML

E.1 Used Class Diagram Elements



(a) Interface. (b) Abstract Class. (c) Class.

Figure E.1: Basic elements.



(a) Class implementing an Interface. (b) Subclass extending a Class.

Figure E.2: Class/interface hierarchy.

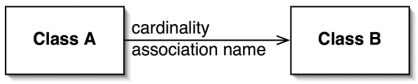


Figure E.3: Association. (Class A holds a reference to Class B and the reference is only navigable from A to B.)

(a) JDK standard library class.

(b) Package.

(c) Note.

Figure E.4: JDK class, Package, Notes.

Bibliography

[Aigner, 2001] Aigner, W. (2001). Guideline Overview Tool (GOT). Technical Report Asgaard-TR-2001-4, Institute of Software Technology and Interactive Systems, Vienna University of Technology, Austria.

[Alder, 2002a] Alder, G. (2002a). Design and Implementation of the JGraph Swing Component. Technical Report 1.0.6.

[Alder, 2002b] Alder, G. (2002b). The Home Page of JGraph. URL, http://jgraph.sourceforge.net.

[Allen, 1983] Allen, J. F. (1983). Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843.

[Astels et al., 2002] Astels, D., Miller, G., and Novak, M. (2002). *A Practical Guide to eXtreme Programming*. Prentice Hall PTR.

[Bade, 2002] Bade, R. (2002). Methoden zur Visualisierung von und Interaktion in zeitbasierten Patientendaten und Behandlungsplaenen. Master's thesis, Otto-von-Guericke Universitaet Magdeburg, Fakultaet fuer Informatik, Institut fuer Simulation und Graphik, Magdeburg, Germany.

[Beck, 2000] Beck, K. (2000). *Extreme Programming Explained*. Addison-Wesley.

[Card et al., 1998] Card, S., MacKinlay, J., and Shneiderman, B. (1998). *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann Publishers, San Francisco.

[Chittaro, 2001] Chittaro, L. (2001). Information visualization and its application to medicine. *Artificial Intelligence in Medicine*, 22(2):81–88.

[Chittaro and Combi, 2001] Chittaro, L. and Combi, C. (2001). Visual Definition of Temporal Clinical Abstractions: A User Interface Based on Novel Metaphors. In *Proceedings of AIME 01: 8th Conference on Artificial Intelligence in Medicine Europe, Lecture Notes in Computer Science*, volume 2101, pages 227–230.

[Combi et al., 1999] Combi, C., Portoni, L., and Pinciroli, F. (1999). Visualizing Temporal Clinical Data on the WWW. In Horn, W., Shahar, Y., and et al., editors, *Proceedings of the Joint European Conference on Artificial Intelligence in Medicine and Medical Decision Making (AIMDM'99)*, pages 301–311, Aalborg, Denmark. Springer, Berlin.

[Cooper, 1999] Cooper, A. (1999). *The Inmates Are Running The Asylum: Why High Tech Products Drive Us Crazy and How To Restore The Sanity.* SAMS Publishing.

[Critical Care Company, 2002] Critical Care Company (2002). QCare. URL, http://www.c3.be.

[Docs Inc., 2002] Docs Inc. (2002). SOAPware. URL, http://www.docs.com/Products/Modules/onlinedemo.htm.

[Duersteler, 2003] Duersteler, J. C. (2003). Information Visualization. URL, http://www.infovis.net.

[Duftschmid, 1999] Duftschmid, G. (1999). *Knowledge-based Verification of Clinical Guidelines by Detection of Anomalies.* PhD thesis, Vienna University of Technology.

[ExtremeProgramming.org, 2002] ExtremeProgramming.org (2002). ExtremeProgramming.org. URL, http://www.extremeprogramming.org.

[Field M., 1990] Field M., L. K. (1990). *Clinical Practice guidelines: Directions for a New Program.* National Academy Press.

[Fox and Thomson, 1998] Fox, J. and Thomson, R. (1998). Decision Support and Disease Management: A Logic Engineering Approach. *IEEE Transactions on Information Technology in Biomedicine*, 2(4):217–228.

[Friedland and Iwasaki, 1985] Friedland, P. E. and Iwasaki, Y. (1985). The concept and implementaion of skeletal plans. *Journal of Automated Reasoning*, 1(2):161–208.

[Furnas, 1981] Furnas, G. W. (1981). The FISHEYE view: A new look at structured files. Technical Report #81-11221-9, Murray Hill, New Jersey 07974, U.S.A.

[Gamma et al., 1994] Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1994). *Design Patterns - Elements of Reusable Object-Oriented Software.* Addison-Wesley, Reading, MA.

[Gershon et al., 1994] Gershon, N. D., Friedhoff, R. M., Gass, J., Langridge, R., Meinzer, H.-P., and Pearlman, J. D. (1994). Is visualization REALLY necessary?: the role of visualization in science, engineering, and medicine. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 499–500. ACM Press.

[Goldstine and von Neumann, 1947] Goldstine, H. and von Neumann, J. (1947). Planning and coding problems for an electronic computing instrument. Part ii, vol.1, U.S. Army Ordinance Department. reprinted in von Neumann, J. 1963. Collected Works Vol. V New York: McMillian, pp. 80-151.

[Gorman and Helfand, 1995] Gorman, P. and Helfand, M. (1995). Information seeking in primary care: How physicians choose which clinical questions to persue and which to leave unanswered. *Med Decis Making*, 15:113–9.

[Gosbee and Ritchie, 1997] Gosbee, J. and Ritchie, E. (1997). Human-computer interaction and medical software development. *interactions*, 4(4):13–18.

[Guarnero et al., 1998] Guarnero, A., Marzuoli, M., Molino, G., Terenziani, P., Torchio, M., and Vanni, K. (1998). Contextual and temporal clinical guidelines. In *Proceedings AMIA Symposium*, pages 683–7.

[Hadorn, 1995] Hadorn, D. C. (1995). Use of algorithms in clinical guideline development in Clinical Practice Guideline Development: Methodology Perspectives. *AHCPR Pub.*, (No. 95-0009):93–104.

[Holzman, 1999] Holzman, T. G. (1999). Computer-human interface solutions for emergency medical care. *interactions*, 6(3):13–24.

[Johnson and Shneiderman, 1991] Johnson, B. and Shneiderman, B. (1991). Treemaps: A Space-Filling Approach to the Visualization of Hierarchical Information Structures. In *Proceedings of the IEEE Information Visualization '91*, pages 275–282. IEEE.

[Keller and Keller, 1993] Keller, P. R. and Keller, M. M. (1993). Visual Clues - Practical Data Visualization. *IEEE Computer Society Press*.

[Kosara, 1999] Kosara, R. (1999). Metaphors of Movement — A User Interface for Manipualting Time-Oriented, Skeletal Plans. Master's thesis, Vienna University of Technology, Institute of Software Technology and Interactive Systems, Vienna, Austria.

[Kosara et al., 2001] Kosara, R., Messner, P., and Miksch, S. (2001). Time and Tide Wait for No Diagram. Technical Report Asgaard-TR-2001-2, Institute of Software Technology and Interactive Systems, Vienna University of Technology, Austria.

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

[Kosara and Miksch, 2001a] Kosara, R. and Miksch, S. (2001a). Metaphors of Movement — A User Interface for Manipualting Time-Oriented, Skeletal Plans. *Artificial Intelligence in Medicine*, 22(2):111–132.

[Kosara and Miksch, 2001b] Kosara, R. and Miksch, S. (2001b). Visualizing Complex Notions of Time. In Roberts, J., editor, *Proceedings of the Conference on Medical Informatics (MedInfo 2001)*, pages 211–215.

[Kosara and Miksch, 2002] Kosara, R. and Miksch, S. (2002). Visualization Methods for Data Analysis and Planning. *International Journal of Medical Informatics*, 68(1-3):141–153.

[Leung and Apperley, 1994] Leung, Y. K. and Apperley, M. D. (1994). A review and taxonomy of distortion-oriented presentation techniques. *ACM Transactions on Computer-Human Interaction*, 1(2):126–160.

[Marcus et al., 2000] Marcus, A., Wieser, K., Armitage, J., and Frank, V. (2000). User-Interface Design for Medical Informatics: A Case Study of Kaiser Permanente. In *Proceedings of the 33rd Hawaii International Conference on System Sciences*. IEEE.

[Messner, 2000] Messner, P. (2000). Time Shapes - A Visualization for Temporal Uncertainty in Planning. Master's thesis, Vienna University of Technology, Institute of Software Technology and Interactive Systems, Vienna, Austria.

[Miksch, 1999] Miksch, S. (1999). Plan Management in the Medical Domain. *AI Communications*, 12(4):209–235.

[Miksch and Kosara, 1999] Miksch, S. and Kosara, R. (1999). Communicating Time-Oriented, Skeletal Plans to Domain Experts Lucidly. In Bench-Capon, T., Soda, G., and A.M., T., editors, *Database and Expert Systems Applications, Proceedings of the10th International Conference of Database and Expert Systems Applications (DEXA'99)*, pages 1041–1051, Berlin. Springer.

[Miksch et al., 1997] Miksch, S., Shahar, Y., Horn, W., Popow, C., Paky, F., and Johnson, P. (1997). Time-Oriented Skeletal Plans: Support to Design and Execution. In *Fourth European Conference on Planning (ECP'97)*. Springer.

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

[Nielsen, 1993] Nielsen, J. (1993). *Usability Engineering*. Academic Press.

[Pattison-Gordon et al., 1996] Pattison-Gordon, E., Cimino, J. J., Hripcsak, G., Tu, S. W., Gennari, J. H., Jain, N. L., and Greenes, R. A. (1996). Requirements of a Sharable Guideline Representation for Computer Applications. Technical Report SMI-96-0628.

[Peleg et al., 2000] Peleg, M., Boxwala, A. A., Ogunyemi, O., and et al. (2000). GLIF3: The Evolution of a Guideline Representation Format. In *Proc. AMIA Annual Symposium*.

[Philips Medical Systems, 2002] Philips Medical Systems (2002). IntelliVue. URL, http://www.medical.philips.com/main/products/patient_monitoring/products/intellivue/index.html#p4.

[Picis, 2002a] Picis (2002a). Chart+ for Critical Care. URL, http://www.picis.com/html/products/module_chart%2Bcritcare.html.

[Picis, 2002b] Picis (2002b). Visual Care. URL, http://www.picis.com/html/products/module_visualcare.html.

[Plaisant et al., 1996] Plaisant, C., Milash, B., Rose, A., Widoff, S., and Shneiderman, B. (1996). LifeLines: Visualizing Personal Histories. In *Proceedings CHI'96 ACM Conference on Human Factors in Computing Systems*, pages 221–227, New York. ACM Press.

[Plaisant et al., 1998] Plaisant, C., Mushlin, R., Snyder, A., Li, J., Heller, D., and Shneiderman, B. (1998). 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.

[Preece et al., 2002] Preece, J., Rogers, Y., and Sharp, H. (2002). *Interaction Design: beyond human-computer interaction*. John Wiley and Sons, New York, NY.

[Quaglini et al., 2001] Quaglini, S., Stefanelli, M., Lanzola, G., Caporusso, V., and Panzarasa, S. (2001). Flexible guideline-based patient careflow systems. *Artificial Intelligence in Medicine*, 22(1):65–80.

[Rao and Card, 1994] Rao, R. and Card, S. K. (1994). The Table Lens: Merging Graphical and Symbolic Representations in an Interactive Focus+Context Visualization for Tabular Information. In *Proc. ACM Conf. Human Factors in Computing Systems, CHI*. ACM.

[Rieger, 1999] Rieger, W. (1999). Process Visualization for Real-Time Applications.

[Rit, 1986] Rit, J.-F. (1986). Propagating temporal constraints for scheduling. In *Proceedings of the Fifth National Conference on Artificial Intelligence (AAAI-86)*, pages 383–388. Morgan Kaufmann, Los Altos, CA.

[Roomans, 2001] Roomans, H. (2001). Formalisation of a medical guideline - Usability investigation of the ASBRU modeling language. Master's thesis, Vrije Universiteit, Amsterdam, Amsterdam, Netherlands.

[Roshtov Software Ind. Ltd., 2002] Roshtov Software Ind. Ltd. (2002). Clicks Medical Information System. URL, http://www.roshtov.com.

[Schaffer et al., 1996] Schaffer, D., Zuo, Z., Greenberg, S., Bartram, L., Dill, J., Dubs, S., and Roseman, M. (1996). Navigating hierarchically clustered networks through fisheye and full-zoom methods. *ACM Transactions on Computer-Human Interaction*, 3(2):162–188.

[Seyfang et al., 2002] Seyfang, A., Kosara, R., and Miksch, S. (2002). Asbru 7.3 Reference Manual. Technical Report Asgaard-TR-2002-1, Vienna University of Technology, Institut of Software Technology & Interactive Systems, Vienna, Austria, Europe.

[Shankar et al., 2002] Shankar, R. D., Tu, S. W., and Musen, M. A. (2002). Use of Protégé-2000 to Encode Clinical Guidelines. In *Proc. AMIA Annual Symposium*.

[Smith, 1996] Smith, R. (1996). Information in Practice: What clinical information do doctors need? *BMJ*, 313:1062–1068.

[Society for Medical Decision Making, 1992] Society for Medical Decision Making (1992). Proposal for clinical algorithm standards. *Medical Decision Making*, 12(02):149–154.

[Spence and Apperly, 1982] Spence, R. and Apperly, M. (1982). Data base navigation: An office environment for the professional. *Behavior and Information Technology*, 1(1):43–54.

[Tufte and Powsner, 1994] Tufte, E. and Powsner, S. M. (1994). Graphical summary of patient status. *The Lancet*, 344(8919):386–389.

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

[Usability First, 2003] Usability First (2003). Usability First. URL, http://www.usabilityfirst.com.

[Wang et al., 2001] Wang, D., Peleg, M., Tu, S., Shortliffe, E., and Greenes, R. (2001). Representation of Clinical Practice Guidelines for Computer-Based Implementations. *MedInfo*, 10(Pt 1):285–9.

[Wellsource, 2002] Wellsource (2002). Coronary Risk Profile (CRP). URL, http://www.wellsource.com/products/crp/crpopen.htm.

[www.openclinical.org, 2003] www.openclinical.org (2003). open clinical - knowledge management for medical care. URL, http://www.openclinical.org.

[XProgramming.com, 2002] XProgramming.com (2002). XProgramming.com. URL, http://www.xprogramming.com.

[Zeng and Cimino, 2000] Zeng, Q. and Cimino, J. J. (2000). Providing Multiple Views to Meet Physician Information Needs. In *Proceedings of the 33rd Hawaii International Conference on System Sciences*. IEEE.