FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Visualization of Compliance with Medical Guidelines

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Master of Science

im Rahmen des Studiums

medizinische Informatik

eingereicht von

Peter Bodesinsky
Matrikelnummer 0304343

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung: ao. Univ. Prof. Silvia Miksch
Mitwirkung: Paolo Federico MSc

Wien, TT.MM.JJJJ

(Unterschrift Peter Bodesinsky)

(Unterschrift Betreuung)

FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Visualization of Compliance with Medical Guidelines

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Medical Informatics

by

Peter Bodesinsky
Registration Number 0304343

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: ao. Univ. Prof. Silvia Miksch
Assistance: Paolo Federico MSc

Vienna, TT.MM.JJJJ

(Signature of Author) (Signature of Advisor)

Erklärung zur Verfassung der Arbeit

Peter Bodesinsky
Lechnerstraße 2-4, 1030 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

(Ort, Datum) (Unterschrift Peter Bodesinsky)

Acknowledgements

I want to thank Paolo Federico and Silvia Miksch for their support and helpful advice. Their review and feedback proved to be indispensable for this work and for the integration of many ideas and concepts. I also like to thank Dr. med. Daniel Firouzabadi, resident physician in neurosurgery, for his participation in the evaluation and his suggestions and ideas. Furthermore I would like to thank Theresia Geschwandtner for providing the source code of CareCruiser.

Abstract

Clinical guidelines recommend applicable actions in a specific clinical context. Computer Interpretable Guidelines (CIG) aim to achieve guideline integration into clinical practice and to increase health care quality. Analyzing compliance with a CIG can facilitate implementation and assist in the design of CIGs, but enabling medical experts to derive insights from this wealth of data is a challenging task.

In this thesis the procedural checking of compliance with an example guideline (artificial ventilation of newborn infants) is combined with interactive visualization. The approach suggested in this thesis covers action-based checking of compliance with a CIG, three types of actions considering compliance are defined: invalid actions, valid actions and missing actions.

Visual mappings and encodings for compliance information found in the course of this work have been integrated into views for patient, treatment and guideline data. Highlighting and visual abstraction inside views for time-oriented patient parameters are supported. The plan-execution view, showing the executed actions during a treatment session, is enriched with encodings for invalid actions and interval-based representations of repeatedly missing actions. Furthermore aggregated compliance information is represented visually inside a view showing the guideline structure.

The approach presented in this thesis is based on the extension of the CareCruiser prototype and implemented in Java. A low scale evaluation was conducted with a physician to gather qualitative data and feedback. Results indicate that the visual encodings for compliance information seem to be intuitive and useful in general, suggestions and found issues give input for future adaptions and refinements.

Kurzfassung

Medizinische Leitlinien enthalten Empfehlungen für die Ausführung von Aktionen in einem spezifischen klinischen Kontext. Computer-ausführbare Leitlinien (computer interpretable guideline, CIG) haben eine bessere Integration von Leitlinien in der klinischen Praxis und die Verbesserung der medizinischen Versorgung zum Ziel. Die Analyse, inwieweit eine Leitlinie befolgt wurde oder nicht, kann deren Implementierung vorantreiben und bei deren Design helfen, jedoch ist es eine große Herausforderung, Experten aus dem medizinischen Umfeld zu ermöglichen, Erkenntnisse aus der großen Menge an Daten herzuleiten.

In dieser Arbeit wird eine prozedurale Methode für die Prüfung der Befolgung einer exemplarischen Leitlinie (Beatmung für Neugeborene) mit interaktiver Visualisierung kombiniert. Der vorgeschlagene Ansatz beinhaltet die Prüfung der Befolgung von Leitlinien auf der Ebene von Aktionen, wobei drei verschiedene Arten von Aktionen, bezogen auf Leitlinienbefolgung, unterschieden werden: nicht korrekte Aktionen, korrekte Aktionen und fehlende Aktionen.

Die visuellen Abbildungen und Kodierungen für Informationen zur Leitlinienbefolgung, die im Laufe dieser Arbeit gefunden worden sind, sind in Ansichten für Patientendaten, Behandlungsdaten und Leitlinien integriert. Innerhalb der Ansichten für die Patientenparameter werden Hervorhebungen und visuelle Abstraktionen unterstützt. Die Ansicht für die Planausführung einer Behandlungssitzung ist mit visuellen Kodierungen für nicht korrekt ausgeführte Aktionen und intervall-basierten Repräsentationen von wiederholt fehlenden Aktionen angereichert. Außerdem ist aggregierte Information zur Leitlinienbefolgung in eine Ansicht, die die Struktur der Leitlinie darstellt, integriert.

Der in dieser Arbeit vorgestellte Ansatz basiert auf der Erweiterung des CareCruiser Prototypen und ist in Java implementiert. Eine Evaluierung wurde in geringem Umfang gemeinsam mit einem Arzt durchgeführt, um qualitative Daten und Feedback zu erhalten. Die Ergebnisse zeigen an, dass die visuelle Darstellung zur Leitlinienbefolgung prinzipiell intuitiv und nützlich zu sein scheint, Vorschläge und gefundene Probleme zeigen Möglichkeiten für zukünftige Anpassungen und Verfeinerungen auf.

Contents

CHAPTER 1

Introduction

1.1 General Introduction

In order to manage the huge amount of complex data and information in health care, Information Visualization (InfoVis) methods may help to perceive relevant patterns visually. On the one hand, InfoVis methods can assist in managing and understanding patient data, on the other hand they can help in the design of computer-executable clinical guidelines. Clinical guidelines are protocols, for example, treatment plans based on scientific evidence, with the intention to maintain and monitor the quality of health care. Decision Support Systems (DSS) based on these guidelines are able to support health care providers and can assist to improve guidelines quality. Adherence or compliance to a guideline can be, among other reasons, an important indicator not only for the quality of treatment, but also for the quality of the guideline. Detecting and managing compliance to a clinical guideline requires both, a computer-executable guideline and patient data, and is closely related to the task of critiquing systems. Supporting compliance management with visualizations therefore requires bringing together visualizations for treatment plans and patient data, together with means to define and analyze compliance to a guideline prescribed medical treatment.

The focus of this work was the extension of already existing visualizations methods and tools and the development of new concepts for visualization of compliance with treatment plans. Furthermore a procedural compliance analysis module for a specific guideline (artificial ventilation of newborn infants) was implemented, together with a categorization scheme for valid, invalid, and missing actions, in order to test, refine and illustrate the visualization concepts within the prototype.

1.2 Motivation and Context

A general definition for compliance with a guideline can be found in [34, p.161] as “acting according to the recommendations of a guideline”.

Guidelines aim to improve health care quality (i.e. the outcome of treatments) and to reduce costs [37]. In general guidelines are defined to be recommendations or statements to assist participants, like physicians or patients, in the care process. According to Croonenborg et al. [37] the definition of the Institute of Medicine (IOM) is commonly used: "Guidelines are systematically developed statements to assist practitioner and patient decisions about appropriate healthcare for specific circumstances." Guidelines are worked out by expert groups, which develop them based on evidence gathered from literature (studies). Recommendations inside a guideline are backed with the level i.e. grade of evidence that can be found in literature. This development process, which takes scientific evidence into account, aims to distill medical knowledge in order to improve the quality of health care.

Importance of Guideline Compliance

Guideline implementation is an important factor to achieve application of guidelines in the daily clinical practice, computer interpretable guidelines (CIG) have been developed to support this task [16]. CIGs can be integrated into Decision Support Systems, which can assist health care providers in decision making; in addition they can be verified to fulfill certain properties due to their formal structure.

Decision Support Systems with means to analyze compliance to a guideline and to visualize the results of this process can further support the process of guideline implementation. Furthermore (non-)compliance can have several reasons and is therefore useful as important indicator. According to Quaglini [34] non-compliance with a guideline is related to both the quality and implementation of the guideline, as well as the users and context related issues. Quality related deviations from a guideline may occur for example due to outdated information in the guideline or due to the fact that the guideline does not cover some cases. Among the reasons of non-compliance which can be associated to the user of the guideline can be errors in the application or disagreement. Issues concerning the implementation context of a guideline can be associated with the organization (e.g. resources, personal) or malfunction related to hard or software. Some reasons of non-compliance are inherent to the guideline design, it is possible that the patient does not match the target population or that the treatment has to be adapted for an individual patient. The visualization methods presented in course of this work might assist in the discovery of (non-) compliance and might therefore also help to understand and identify the various reasons for it.

Need for Visualization

Dealing with medical data, which includes data about patient observations and treatment data, together with clinical guidelines in order to visualize compliance with a guideline involves handling with huge amounts of data. Increasing amounts of data are generated due to technological advancement, not only in medicine and patient related health care, but also in fields like engineering, oceanography [8], and finance [42]. The means of receiving and storing such huge amounts of data offer the possibility to extract knowledge, relate different parameters to each other and to identify complex patterns. In order to achieve this it is necessary to process and analyze data in advance, because a human would be overwhelmed with the interpretation of raw

data. Information Visualization is an approach to solve this issues, Card et al. give the following definition: “The use of computer-supported, interactive, visual representations of abstract data to amplify cognition” [11, p.6]. Information Visualization can help to emphasize important aspects of the data and to compress information, by generating interactive images which take advantage of the visual pattern recognition ability of human cognition. Visualization sort of combines the ability of machines to process large amounts of data to generate images, with the human ability to extract patterns and hypothesis from these visualizations.

This trend of generating more and more data, sometimes referred to by the term big data, makes it necessary to extend common visualization techniques only showing raw data by additional analytic methods. Advanced methods are especially useful for healthcare data, as described by Aigner et al. [2], this data can be very complex, involves a lot of semantic information, includes patient parameter observations and treatment data, demands scalability, and requires task-specific visual representations and interactions. Visual analytics [26, 47] is a field, which tries to integrate information visualization and advanced computational analysis methods to further reduce the information overhead. It applies the mantra “Analyze First – Show the important - Zoom, Filter and Analyze Further – Details on Demand” [26, p.82]. Important aspects are extracted as far as possible by automatic means before generating visualizations. Visualization of compliance to a guideline can also be seen under this aspect, a machine preprocesses data in a certain context and preselects important information in order to generate interactive images tailored to gain insight to the data from a specific point of view.

1.3 Research Questions

The main research question of this thesis is:

RQ How to support a medical expert in the analysis of compliance with a guideline by means of visual analytics?

In order to achieve this, additional (sub-)questions arise:

Rq1 How to assist in the detection of patterns?

Rq2 How to integrate information about guideline compliance into views for guidelines, treatment and patient data in a consistent manner?

User feedback and information about the usability of the system has also to be gathered to answer the following questions:

Rq3 Does the visualization match the need of the user?

Rq4 How well can the visualization be understood and helps in the analysis tasks?

From a more technical point of view, the following questions can be considered:

Rq5 How to define compliance in order to generate information about compliance?

Rq6 How to represent this information inside interactive views?

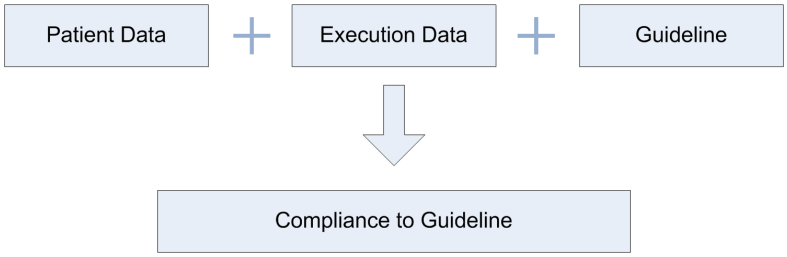


Figure 1.1: The data and information required to determine guideline compliance

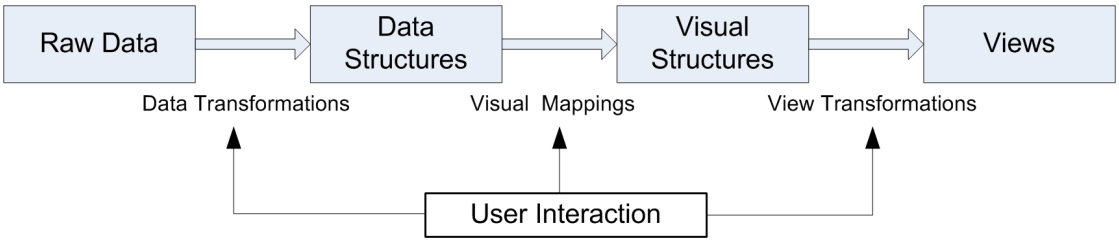


Figure 1.2: the pipeline i.e. transformations used in information visualization (as described in [10, 11, 28, 29])

1.4 Problem Definition

The visualization of compliance with medical treatment plans therefore requires two different subtasks to be solved. The first task is to define and determine compliance to a computer-executable guideline. Followed by the visual representation of the generated information and the integration it into visualizations for patient data and guidelines.

The first step requires an exact definition of compliance to a guideline, because adherence can be defined on several levels, e.g., on the level of executed actions (also see section 3.1 Compliance analysis). This can depend on several factors like the scope of the guideline, the guideline formalism, the affordable computational complexity, and the tasks to be solved. Determining i.e. analyzing compliance based on the definition requires patient data, which includes measurements (observations), as well as execution data (treatment), together with a guideline (CIG) (cf. Figure 1.1)

The latter step of visualization is necessary to make the complex and hard to grasp data accessible, this data includes patient observation and treatment data, as well as the guideline itself, which are connected with the analysis of compliance. Generating images that enable to visually perceive patterns directly requires to find proper mappings and visual encodings for the underlying data [10, 29], together with interaction methods. This mapping process can be illustrated by the information visualization pipeline (cf. Figure 1.2). Raw data undergoes a series of transformations, data transformation are applied to create data structures suitable for the visualization. Visual mappings are applied afterwards in order to create the visual representations. The last step consists of view transformations

Considering this thesis work, especially the mapping of the results generated by the compliance analysis module had to be determined (together with adaptations of already existing visual mappings provided by Care Cruiser) to create suitable representations. The mapping to visual properties allows the data to be processed automatically by the so-called preattentive memory. Raw data can be mapped to different properties, one of the most effective and accurate attribute is the spatial substrate (e.g. mapping to x or y axis), but only up to three (3D,in most cases rather 2D) variables can be represented in this way (in a single view). Another possibility is the mapping to properties of the visual elements themselves, which can be e.g. the shape, size or volume. Furthermore properties like color, transparency, or texture can be altered to encode data.

1.5 Thesis Structure

The thesis is structured in several parts:

- **Chapter 2 Method and Concepts**: This chapter starts with an outline of the research methodology applied for the thesis (section 2.1 Research Process). The basic concepts and technology are explained afterwards. Section (2.2 Guideline) describes the semantics of the example guideline. Section (2.3 Asbru) gives an explanation of the main properties and elements of the Asbru plan representation language, which is the language used for the guideline supplied with CareCruiser and used as basis to analyze compliance in the course of this work. Section (2.4 Definition of Action Compliance) introduces the used definition of action compliance and the different action types identified in this context. Section (2.5 Frameworks) provides a quick overview of the architectural frameworks used by CareCruiser (prefuse,swing), as knowledge about them is required to introduce extensions.

- **Chapter 3 Previous and Related Work**: Section (3.1 Related Work) gives an overview of work related to this thesis considering different fields like visualization of patient data and compliance analysis. Section (3.2 Previous Work) gives an overview of the preceding work and the original prototype (CareCruiser), which has been extended and modified in the course of this work.

- **Chapter 4 Implementation and Visualizations**: The main contributions, found visual mappings and interface modifications are presented in this chapter. Section (4.1 Overview of Interface Modifications) provides a quick overview of the original interface and the introduced adaptions. Section (4.2 Visual Artifacts and Encodings) outlines the visual artifacts and encodings, together with a more detailed explanation of the interface modifications. Section (4.4 Documentation) provides a technical overview of the original CareCruiser and of the extensions and modules developed in course of this work.

- **Chapter 5 Evaluation**: Results of the evaluation and the method used for it are presented in this chapter.

- **Chapter 6 Discussion and Future Work**: This chapter concludes the thesis, provides a summary of the achieved results, discusses them and gives an outlook to possible future work.

This thesis is partly based on the paper "Visual Analysis of Compliance with Clinical Guidelines" [9] published in conjunction with this work.

CHAPTER 2

Method and Concepts

This chapter starts with a description of the process and methods of research for the thesis (section 2.1 Research Process). To give insight into the basic concepts, a short description of the semantics of the used clinical guideline is given afterwards. (section 2.2 Guideline). The guideline, which is used as basis to analyze compliance and supplied with CareCruiser, is specified in the Asbru plan representation language, section 2.3 Asbru gives an overview of the main language properties. The subsequent section presents the definition of action compliance elaborated in course of this work (section 2.4 Definition of Action Compliance). The most important frameworks, used by the prototype and for the implementation of the additional functionality in this work, are Prefuse and Swing, a quick overview of them is therefore provided in section 2.4 Frameworks.

2.1 Research Process

The research and work for this thesis consisted of several steps. At first it was necessary to do extensive literature research to identify the main concepts. The definition and context of terms like “clinical guidelines” and “guideline compliance” had to be found out, furthermore general knowledge about information visualization and visual analytics had to be gathered. The problem definition was worked out as result of this process. An approach based visual analytics was worked out (see chapter Introduction), which means that compliance to a guideline is determined automatically and the results are represented within an interactive visualization. The next step was to search and review existing approaches. Literature dealing with different definitions of compliance, as well as work in the field of guideline and patient data visualization was explored. Possible solutions and problem solving approaches were considered. CareCruiser was chosen as foundation for the implementation, as it provides suitable views for patient, treatment and guideline data. The next step consisted in working out the exact definition of compliance and concepts for the visual representation of compliance information. To start with the actual implementation, the necessary frameworks, libraries and the source code of CareCruiser had to

be explored. The implementation was done iteratively, the underlying concepts were reviewed and refined in course of the implementation (e.g. the visual representation of missing action intervals). The final step was the evaluation of the system, methods used in information visualization and usability engineering have been explored (also see chapter Evaluation). A procedure for the user test and a subsequent interview was worked out, the evaluation was performed and the results were documented.

2.2 Guideline

The guideline used in this work contains treatment prescriptions for the artificial ventilation of newborn infants and is packaged within CareCruiser, for more information about this topic and on the guideline design please refer to Fuchsberger [19], this section only gives a brief overview.

The intention of (mechanical) ventilation is to ensure the oxygen supply for the patient by applying pressure (pressure gradient) to the lungs. During the process of ventilation the parameters SO2 (*oxygen saturation*) and pCO2 (*partial pressure of carbon dioxide*) have to be kept in a normal range. The ventilation of the patient can be controlled by adjusting a set of parameters (PEEP, PIP, FIO2, frequency):

- PIP (*peak inspiratory pressure*) is the highest level of pressure applied in the inspiratory phase.
- PEEP (*positive end expiratory pressure*) is the pressure applied at the end of expiration, normally a positive pressure is applied during the end of expiration, although not as high as the PIP pressure.
- FIO2 (*fraction of inspired oxygen*) this factor represents the amount of oxygen in the air, provided by the ventilation machine.
- The *ventilation frequency* i.e. respiratory rate is measured in respirations per minute.

The guideline starts with a set of initial plans, which are applied in sequential order. The initial values of ventilation parameters (PIP, PEEP, FIO2, frequency) are configured within these plans depending on the patient state. After the initial configuration phase the controlled ventilation starts, causing the execution of repeating subplans, which aim to continuously keep the parameters SO2 (i.e. tcSO2 measured transcutaneously) and PCO2 (calculated using tcpPCO2 and offline PCO2 measurements) within normal range.

- PCO2 handling: The plan responsible for handling the PCO2 parameter adjusts the ventilation frequency accordingly, if the PCO2 value gets out of range.
- SO2 handling: The tcSO2 values are handled in a similar way, by adapting the oxygen level (FIO2).

Controlled ventilation stops, if either the conditions for plan abortion (FiO2 or PIP or pCO2 above a certain threshold) or completion are true (FiO2 and PIP beneath threshold).

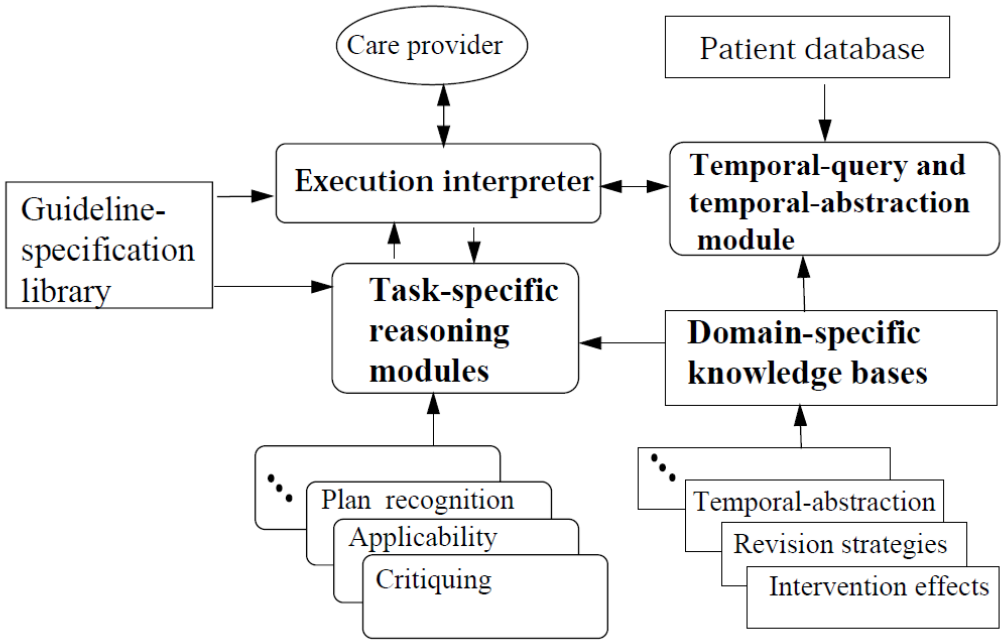


Figure 2.1: The architecture of asbru/asgaard, image from [41].

2.3 Asbru

The guideline supplied with CareCruiser and used in the course of this work is formulated in Asbru, a plan representation language, which is part of the Asgaard project [39, 41]. Asgaard aims to support guideline development and applications and the tasks related to it. This includes guideline verification and validation, as well as tasks like execution and critiquing [41]. Asbru formalization language is intended to facilitate the solution of these tasks by providing enough expressivity. Guideline critiquing (i.e. compliance checking) on a high level, for example, requires language elements representing intentions and effects of a guideline (to check if the execution matches the actual intentions).

The architecture of Asbru (cf. Figure 2.1) takes patient data and the guideline as input, in order to instantiate the plans for the current treatment. By using knowledge bases and abstraction, the patient data is prepared to be handed to the interpreter and reasoning modules. The output can be plan critique, as well as recommendations, explanations, or visualizations [25, 41].

Language Elements

Asbru is used to model time-oriented skeletal plans, which are meant to allow flexible execution. It supports different language elements (cf. Figure 2.2).

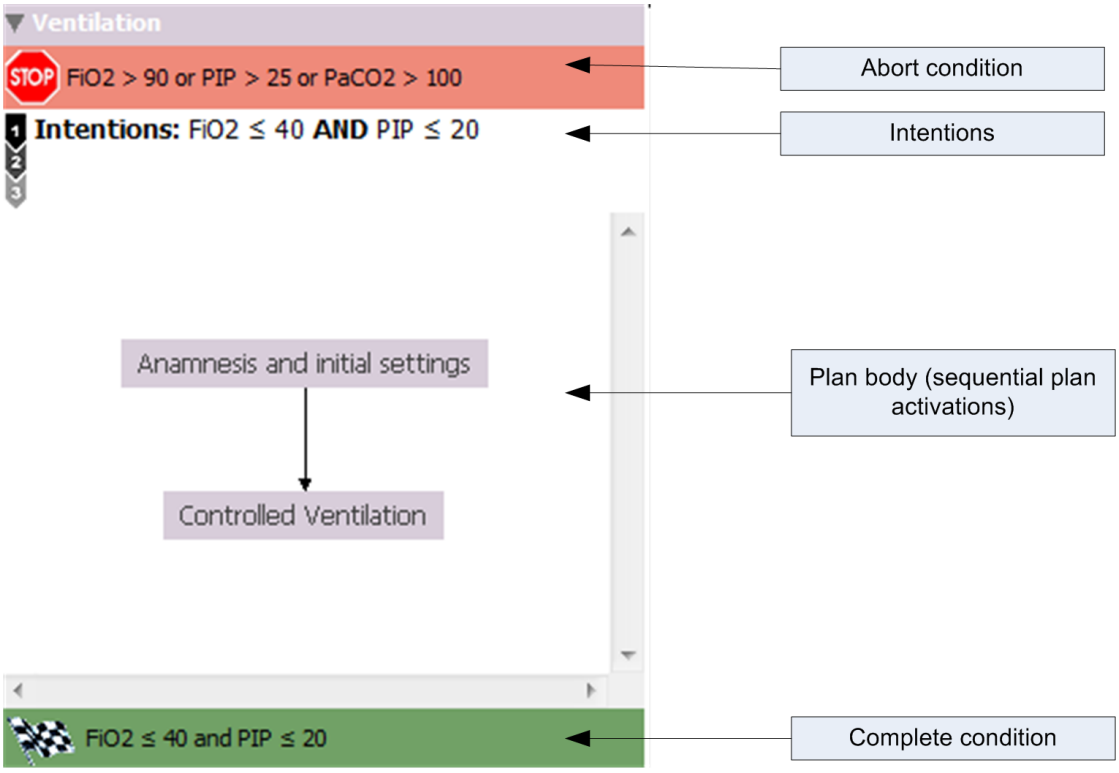


Figure 2.2: Example for an Asbru plan (visualized by CareCruiser)

- *Preferences* influence plan selection, e.g. costs and resources or the selection strategy can be specified for a plan.
- *Intentions* model the goals of a treatment, they are modeled by temporal patterns, which should be "maintained, avoided or achieved" [41]. An example would be an intention stating that a physiological parameter of a patient should stay below a certain threshold, or that the temperature of the patient should not exceed a certain value.
- *Conditions* are used to alter the execution state of a plan (e.g. from active to suspended), i.e. state transition are caused by conditions (also see Plan States).
- *Effects* model the relationship between the execution of plan (i.e. action) and a parameter measurement of a patient, e.g. an effect can state that a plan decreases the level of blood glucose with specific likelihood.

Time Annotations

Complex time annotations are supported (cf. Figure 2.3) e.g. for defining the duration of a plan/action, or for conditions (i.e. time during parameter propositions have to be fulfilled), and

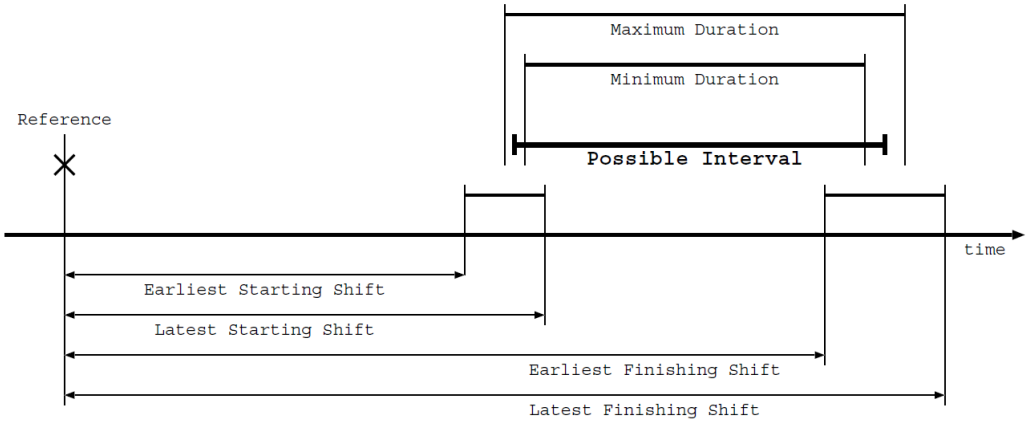


Figure 2.3: Asbru time annotations, image from [39]

are meant to allow modeling temporal uncertainty. A time annotation consists of a reference point, which, for example, can be a certain date, a plan state (time where another plan e.g. gets activated) or the time point when the plan itself is executed (keyword "self"). Time ranges are defined in relation to a reference point and are specified by a set of parameters. The *earliest starting shift* (ESS) and the *latest starting shift* (LSS) define the interval allowed for starting, the *earliest finishing shift* (EFS) and the *latest finishing shift* (LFS) define the allowed interval for finishing, in addition the minimum and the maximum duration of the time range can be defined.

Plan Body

An Asbru plan can contain other plans, i.e. subplans, the ordering and activation of these plans is specified in the plan-body. Several types of multiple subplan execution are supported. Subplans can be executed *sequentially*, which means that the plans are activated in sequence, a succeeding plan can only be activated if the preceding plan has finished. *Parallel plans* are executed together, i.e. start at the same time. Alternatively subplans can be executed in *any-order* or *unordered*. All these types of subplan execution handle multiple plan-activations.

A *cyclical plan* defines repeating plan activations, i.e. a single plan step activated repetitively. It contains a repeat specification, which allows to set a retry delay (minimum and maximum delay) i.e. the duration and the start and end of the cyclical execution. For example it can be specified that the time delay between a repeating activation of a plan has to be at least 10 min (minimum delay) and not longer than 12 min (maximum delay).

A *single plan step* contains elements like a (single) plan activation, a variable assignment or a if-then-else construct, which allows the choice of multiple alternatives (i.e. plan activations).

Actions are atomic plans, which do not contain any subplans, these can be user performed plans (xml element user-performed), containing e.g., an assignment, or primitive plans, which implement system-supplied methods [39]. The concept of an action, modeling an atomic clin-

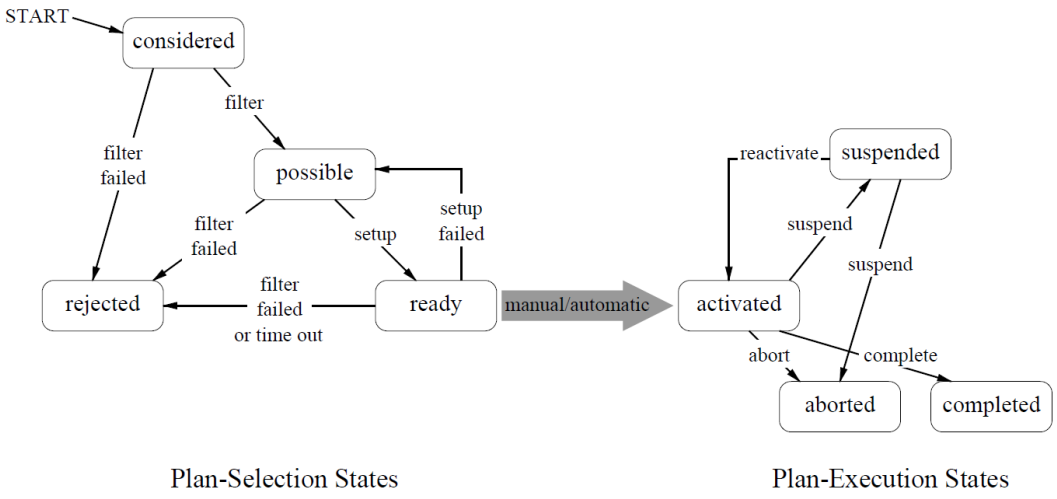


Figure 2.4: States and conditions of Asbru plans, image from [39]

ical activity, is important for this work, because the compliance checking is done on this level, comparing actual executed actions against guideline prescribed actions.

Plan States

Asbru plans can have several states, which can either belong to the plan preselection or the execution phase (cf. Figure 2.4).

Initially an Asbru plan is in the considered state of the *preselection phase*. A change to the state possible is achieved, if the filter precondition of the plan is fulfilled (e.g. gender or age), if the condition is not fulfilled the state is altered to rejected. A plan can get in the state ready if the setup precondition holds. Setup preconditions are conditions, which get fulfilled by the patient or the physician. Being in the ready state, a plan can get activated, either manually or automatically. As soon as the plan is in activated state, the plan is in the *execution phase*. The plan can be in four different execution phase states:

- Activated
- Suspended
- Aborted
- Completed

A plan can get suspended if the suspend condition gets true, but can be reactivated as soon as the reactivate condition gets true. A plan can either fail or be finished successfully; the plan is completed successfully, if the complete condition is fulfilled (e.g. parameters below a certain threshold) or fails, if the abort condition is fulfilled.

2.4 Definition of Action Compliance

This section gives a detailed explanation of the definition of action compliance used and elaborated in the course of this work. Compliance in this work is checked on the level of (atomic) clinical actions, which means an action, which is executed by a care giver (or by a machine). The actions prescribed by the guideline are checked against the user executed actions i.e. the actions applied during treatment. In this context a user executed action can have three types regarding compliance to the guideline:

- *Valid Action*: The action is applied correctly i.e. the executed action is compliant with the guideline.
- *Invalid Action*: An action, which has been applied during the treatment, but should not have been applied according to the guideline. This can, for example, be an action for which the conditions for action application do not hold, or temporal restrictions are not fulfilled.
- *Missing Action/Missing Action Intervals*: These are actions, which should have been applied, but are missing in the execution. In case of repeatedly missing actions, these are time-spans (intervals) during which an action should have been applied, but was not applied.

The computation of valid, invalid and missing actions is handled differently in detail for the initial plans and the two (more time dependent) ventilation plans, as explained in the following sections. Executed actions, which have no equivalent in the guideline, are automatically defined as invalid actions.

Initial Plans

Initial plans and their actions are applied sequentially, which means that they have to be applied in the correct order. It is assumed that they have been executed in the correct order or that the order of the actions is not important, if they are applied at the same point in time, in this case the definition of action compliance types is:

- Valid action: The conditions of the action are true, it has been applied correctly
- Invalid action: The action has been applied, but the conditions are not true.
- Missing action: The conditions for the action are true, but it has not been applied

The more complex case occurs if the action order is regarded additionally i.e. the initial actions have been applied at different time points.

- Valid action: If the action is executed in the right order and the action condition is true, the action is considered to be valid.

- Invalid action: An invalid action is an action, where the conditions are false and/or where the order of execution is wrong. The action is invalid due to execution order, if the action is executed, although later actions (that are applied after the action) already have been applied.

- Missing action: If an action is applied, but other actions should have been applied before, the action(s), which should have been applied before, are generated as missing.

Table 2.1 gives an example for that behavior. A more fine grained scheme might be possible (also see section Related Work), e.g., differentiation between invalid due to condition and invalid due to order, but would also increase the complexity of the visualization and take away the possibility to aggregate invalid/valid/missing counts for all plans (classification scheme would be different for initial versus ventilation plans).

(a) Executed Actions	**(b) Prescribed by Guideline**	**(c) Checked Actions**
Action 1	Action 1	Valid: Action 1
Action 4	Action 2	Missing: Action 2
Action 2	Action 3	Missing Action 3
	Action 4	Valid: Action 4
	Action 5	Invalid: Action 2
		Missing: Action 5

Table 2.1: Example for compliance related to action order for sequential actions (plans). Column (a) shows an example user execution of three actions, Column (b) shows the actions, which should have been executed according to the guideline. Column (c) shows the actions after compliance has been checked. The instance of Action 1 is valid. Missing actions instances are generated for Action 2 and 3, because they have not been applied until the execution of Action 4. Action 4 is applied validly. The execution instance of Action 2 is invalid, because it appears after the execution of Action 4. Action 5 is missing in the execution.

Ventilation Plans

The more time-dependent ventilation plans are repeated according to their minimum and maximum delay. The definition of action compliance for these plans is, see figure 2.5 for an example:

- Valid Action: An action is regarded as valid, if the conditions for the action are fulfilled (parameter too low or too high) and if it is applied after reaching the minimum delay, i.e. if the temporal distance to the previously applied action of the plan is equal or greater than the minimum delay.

- Invalid Action: Invalid actions are actions, which have been applied although the conditions are not fulfilled, or which are applied before the minimum delay is reached i.e. the temporal distance to the previously applied action is smaller than the minimum delay. An action is also marked as invalid, if it occurs after a change to abort or completed state.

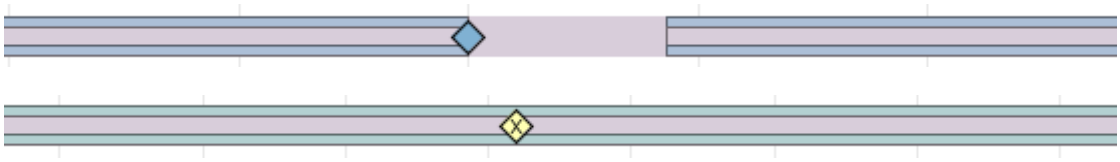


Figure 2.5: Execution example (a) shows an action, which has been applied validly (blue diamond). The missing interval (marked with the upper and lower blue bar) on the left side spans until the valid action is applied. It marks a time span, starting at the point where the action should have been already applied, but was not applied, and ending at the application of the valid action. The missing interval on the left starts at the time point at which the maximum delay is reached, this is the latest possible time point at which the next action should be applied. It continues, because no action is applied so far. Example execution (b) shows a missing interval, which indicates that no valid action has been applied for the whole duration (although it should have been applied). An action has been executed by the user (yellow diamond), but it is the wrong action, i.e. an invalid action not according to the parameter value. It is marked as invalid therefore und the missing interval continues without interruption.

- Missing Intervals: A missing interval has a start and an end point in time. A missing interval starts as soon as two prerequisites are fulfilled: The maximum delay has to be reached, i.e. the temporal distance to the previously applied action is greater than the maximum delay, and the conditions for action application are true (but no valid action is applied). A missing interval is closed, if the conditions are not fulfilled anymore, if the corresponding action is applied, or if the abort or complete condition gets true. The amount of missing actions can be estimated from the missing intervals (e.g. dividing the missing interval time span by the maximum delay time). The amount of actual actions, which might have been applied during a proper execution cannot be exactly computed, because of the specified minimum and maximum delay (exact time point of execution depends on user choice).

2.5 Frameworks

The prototype framework (CareCruiser) and the extensions made to it in the course of this work are implemented in Java. The Prefuse toolkit is used for the visualizations, it enables to define the necessary transformations and mappings, to specify rendering and views alongside with interaction behaviour (section Overview of the Prefuse Toolkit). Swing is the basic framework applied for GUI definition, GUI component layout and interaction can be set with the help of this API (section Overview of Swing API).

Overview of the Prefuse Toolkit

Prefuse [28] is a free framework intended to be used in information visualization applications. It represents a practical implementation of the information visualization reference model [13] and

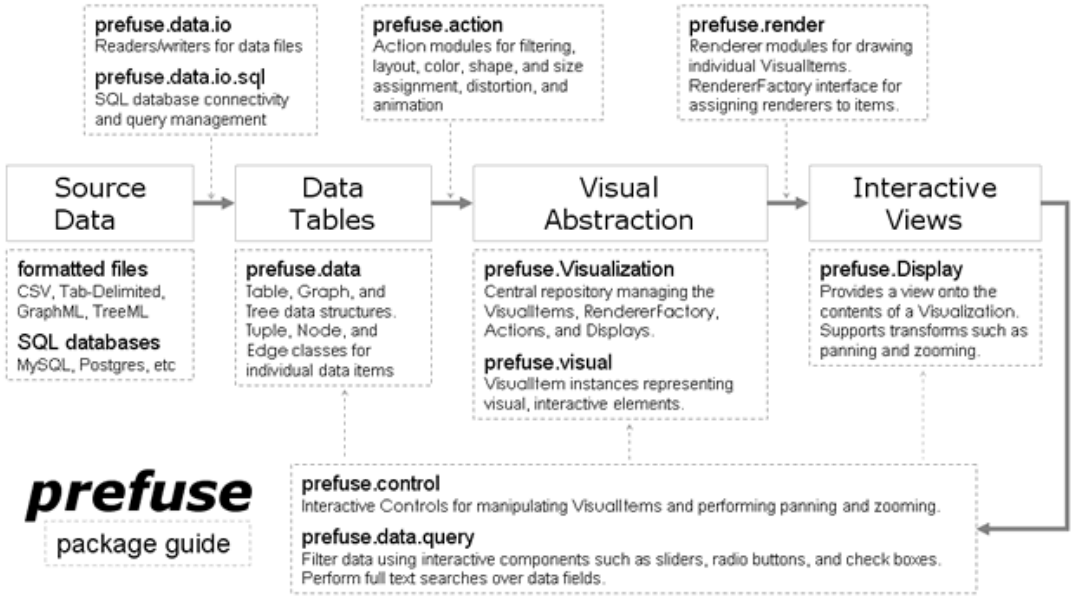


Figure 2.6: Structure of the prefuse toolkit, image from [28]

facilitates application design following this model. The framework offers predefined classes, as well as the possibility to extend existing functionality through the use of inheritance.

The source data used by the visualizations is accessed through reader (and writer) classes (`prefuse.data.io`), which for example can execute SQL queries or phrase CSV files. The resulting data is represented in prefuse data structures (`Table`, `Graph` and `Tree`) and can be transformed, i.e. filtered by expressions (`prefuse.data.expression`).

After reading and data structure generation it is possible to generate visual abstractions, which are represented by `VisualItem` instances. These items contain attributes like shape, size and color and represent visualization elements, which have been derived from the data. The way the item attributes are derived from the data is determined by action classes. These actions perform the visual mappings from the data structures to visual items (`prefuse.action`). An example for this mechanism is a graph data structure containing the attribute gender in his graph node (see prefuse manual example application[1]). This attribute can be mapped to the color of the visualized node by a `DataColorAction`, which can connect the data attribute gender to the visual color attribute of the node, e.g. the fill color. Other actions are responsible for performing operations on visual items related to the layout or animation.

Drawing of the visual items can be performed by different renderer classes, which are responsible for the appearance on the screen (`prefuse.render`). The viewable area and items, and the view transformations of the visualization are determined by a `Display` instance

[1] http://prefuse.org/doc/manual/introduction/example/

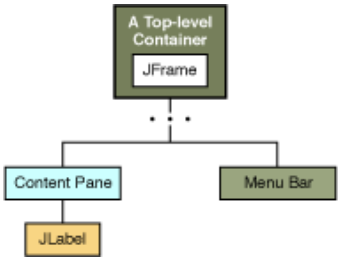


Figure 2.7: Example containment hierarchy, image from [48]

(`prefuse.display`). Different controls can be added to the display to support interactions like zooming or panning(`prefuse.controls`).

Overview of Swing API

The GUI of the used framework is based on the Java Swing API [48], which offers components for the design of interactive interfaces. A Swing program uses a hierarchical model i.e. containment hierarchy to manage the separation and layout of different GUI Elements inside a window. Top level containers form the root of this hierarchy, e.g. `JFrame` (decorated window) or `JDialog` (used in conjunction with a main window). The subnodes of this hierarchy can also be containers (components extending `javax.swing.JComponent`), which can contain other containers or basic UI components (cf. Figure 2.7). `JPanel` is a lightweight container and is commonly used to subdivide a window into different areas. Child nodes can be added to their parent containers by a method call. The layout of GUI elements inside a container is determined by a layout manager like `FlowLayout` or `GridBagLayout`, or by supplying the GUI components with absolute coordinates.

Interaction is supported by the `EventListener` interface, which can be registered on a GUI component (e.g. `button.addActionListener(listener)`). Listeners have to implement callback methods, which are executed inside a special thread (event dispatch thread) if the corresponding event occurs (e.g. button click). Another important concept is implemented by models connecting data to the GUI (adapted MVC model called separable model architecture). Operations on the data of the model are propagated to the GUI and changes in the GUI are propagated to the model data.

CHAPTER 3

Previous and Related Work

This chapter presents previous and related work. An overview of work related to the thesis on different conceptual levels is provided in the next section (Section 3.1 Related Work), this includes work dealing with information visualization in medicine, event visualization and compliance with clinical guidelines. Preceding work (Section 3.2 Previous Work) includes the original CareCruiser prototype, which forms the basic framework for the extensions made in course of this work, and its predecessor CareVis.

3.1 Related Work

This work deals with compliance analysis and particularly with the visualization of the result of this analysis process, which is based on the extension of views for patient treatment data, patient observation data and guideline (CIG) data. Related work therefore involves concepts and works dealing with compliance analysis, with the visualization of patient observation data, treatment data and guideline structure, and work dealing with basic concepts, like data abstraction and event visualization.

Visualizations for Patient Data and Guidelines

Visualization of patient data is a huge field, which requires visualization of numerical as well as categorical data together with appropriate interaction techniques [36]. The time dimension is an important factor, because patient data is very commonly linked to the time dimension. There are various aspects to be considered [5], for example, the time scale can be ordinal (e.g. before, after), discrete or continuous. Time-depended values i.e. instances can stretch over an interval or represent a single point in time, or the time scale can be represented in different granularities (hours, days, weeks etc.). Patient data includes data from medical observations and treatment, this data is also important for compliance analysis, visualization of results of the analysis process and for identifying the effects of compliance/non-compliance. The application of visualizations for patient data is therefore required. Common techniques for numerical data

visualization are scatter plots, line plots, or bar charts, moreover numerical data can be abstracted and thus represent also categorical data. Categorical data is commonly displayed using bars i.e. lifelines (indicating the duration of an event), matrix charts, or icons [36]].

Systems applying Dot Plots for Patient Data Visualization

2D Scatter i.e. point plots (or line plots, if the data points are connected by lines), where the time of an observation is mapped to the spatial time axis (horizontal) and the value of the observation is mapped to the vertical axis, is a technique also applied by CareCruiser, from which the corresponding views are extended in this work. Examples for systems using this technique in a similar way for the display of multiple patient parameters are Graphical Summary of Patient Status and CPDV (cf. Figure 3.1).

Graphical Summary of Patient Status [43], one of the earliest attempts to visualize patient data, displays patient observations and treatment with the help of small repeated graphs, the information for each variable is contained in one graph. Data values are represented by dots, whereas the time scale is compressed to show the history of development of the values in a compact way. Values from years ago are visualized on a very compact time scale, together with the development of the most recent values.

CPDV (critical care data visualization) [17] i.e. MIVA uses scatter plots to visualize multivariate (i.e. more than one variable describing the patient's state) patient data. The authors point out that it was designed to integrate data from various sources and to allow selecting, arranging and visualizing data related to a patient's state in a way that supports clinical decisions. Multiple datasets for one patient can be selected and grouped together by drag and drop, depending on the needs of the physician. It is possible to select the data sources and the temporal resolutions for the different visualizations, for each day of treatment, one can specify which hour of the day should be displayed and the granularity of time (ranging from 30 minutes to 30 seconds).

Latter iterations of CPDV resulted in the MIVA Prototype [18], which supports the dragging of parameters listed on the left side of the interface to open a visualization for them. Furthermore the data can be displayed together with intervention notes entered by a physician.

Systems applying Multiple Coordinated Views for Patient Data Visualization

VisuExplore [35] is an example for a system unifying visualizations for quantitative as well as qualitative data types, similar to the patient data dealt with in this work (and CareCruiser), the data is time oriented and time is mapped to the horizontal axis. According to the authors the goal was to design a simple and easy to use system for the long term care of patients with chronic diseases (diabetes mellitus). The interface supports multiple views, which are arranged on a common time axis and different visualizations depending on the data type are provided. Multiple variables of the same data type can be viewed in one diagram (e.g. in the line plot) and it is possible to rearrange, add, close and resize the views.

The tool provides basic visualizations (cf. Figure 3.2), it offers line plots and bar charts for numerical data, the diagram can scale either to a set value range, or to the minimum and maximum value range of the data itself. Qualitative data can either be displayed by event charts

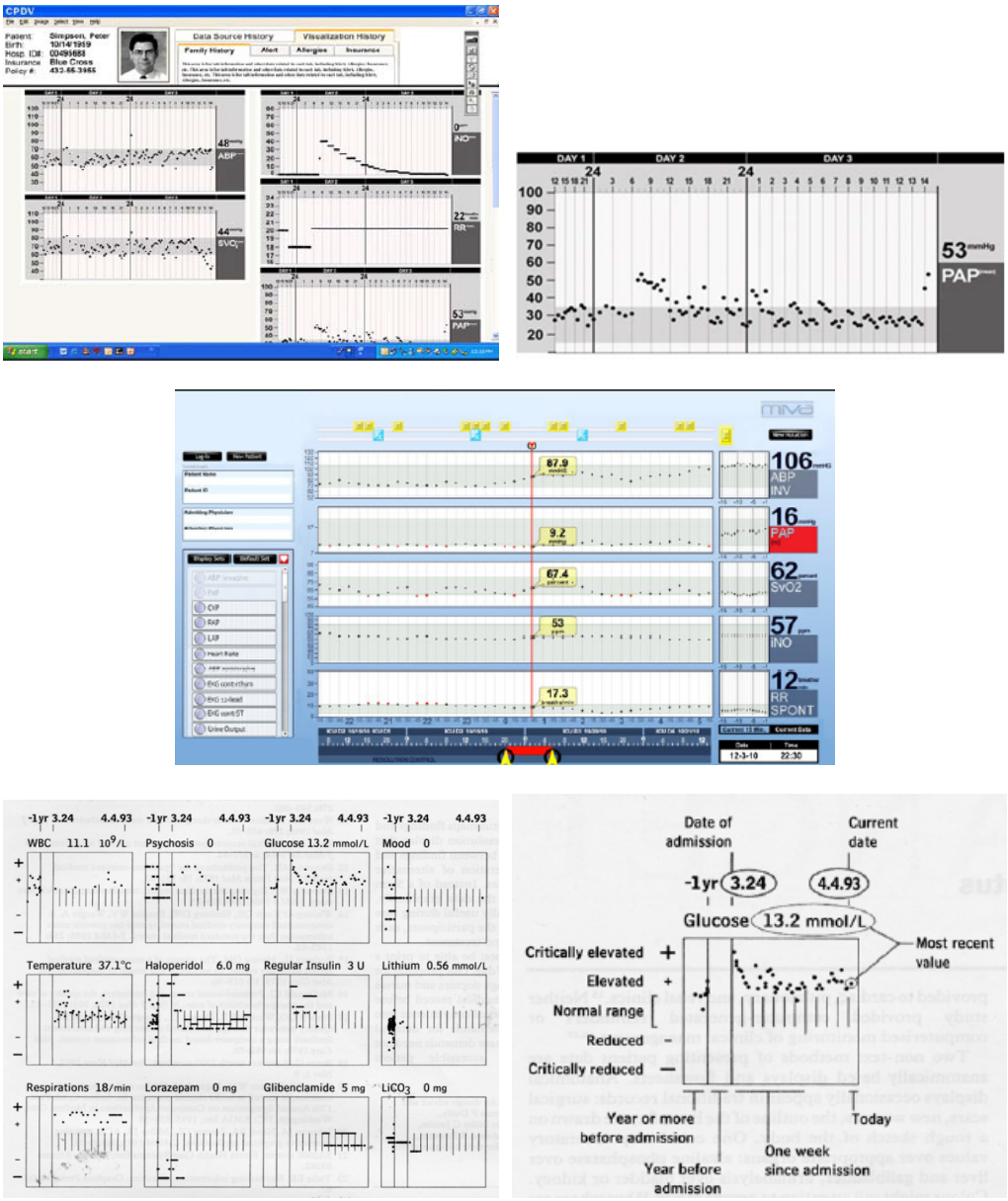


Figure 3.1: Examples for the display of patient data by applying dot plots in CPDV (on top), image from [17], further development resulted in the MIVA prototype (in the middle), image from [18]. The graphs of graphical patient summary are shown on the bottom, image from [43].

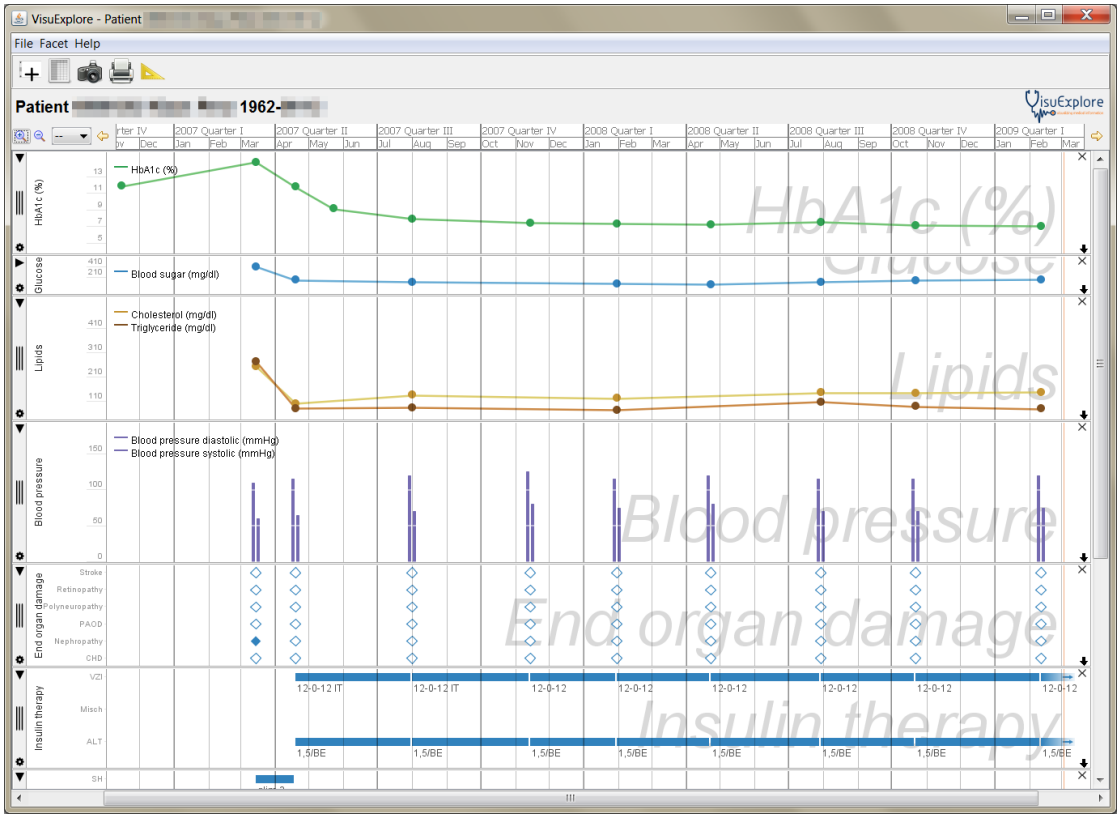


Figure 3.2: Interface of VisuExplore showing the basic visualization techniques, i.e. line charts, bar charts, event charts and time line charts (source: http://ieg.ifs.tuwien.ac.at/projects-/VisuExplore/visuexplore-web-simple.png).

or by timeline charts, which are designed to show time interval data. In addition the tool also implements more advanced visualizations, like horizon graphs and semantic zoom charts.

Interaction methods include panning, zooming and tooltips to see the value of a variable, all values of a view can also be inspected inside a textual table. A mouse tracker spanning over the vertical extend of all views allows to relate values occurring at the same time point. Furthermore a functionality called measure tool enables to connect two points of two different diagrams to measure the temporal distance between two time points.

Visualizations for patient data have also been adopted for mobile devices, Chittaro proposes a system [14], which displays patient data on small screens. The interface allows switching between different time granularities. Different visualization techniques are supported, bar charts (temperature) are applied for numerical values and treatment (e.g. medications) events are marked by circles or intervals in case of durative events. Multiple visualizations can be viewed together, either in a stacked above manner, or combined into one view (using line charts for

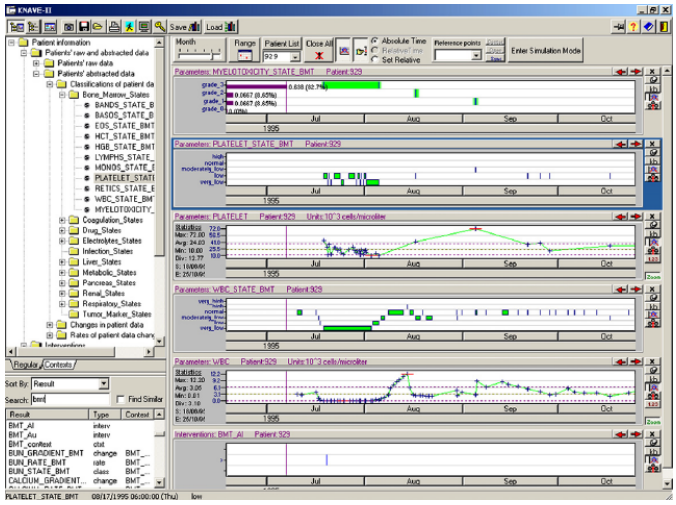


Figure 3.3: Knave II interface with the concept browser on the left and the visualizations on the right, the abstractions are encoded by green rectangles, the class is mapped to the vertical displacement and the duration mapped to the length (image from [40].

numerical values).

Knave II [40] supports visualizations for abstracted data (i.e. qualitative data visualizations derived from source data) based on a spatial mapping similar to the one used for the abstraction overlay in this work (cf. Figure 3.3). The abstraction class is mapped to the vertical displacement of a rectangle, whereas the duration of time, during which the values fall into the same qualitative class, is encoded by the horizontal length of the rectangle.

Knave II is a system which integrates different abstraction concepts for time-oriented data from multiple sources. It makes use of domain specific knowledge to calculate the abstractions and supports multiple abstractions for the same data type. The interface includes a browser on the left, where the source data and the abstractions derived from it are hierarchically listed. Double clicking selects an item for visualization and opens a view for it. Furthermore it is possible to view the function used for data abstraction and to inspect related concepts (semantic explorer).

The left side of the interface contains the view panels with the visualizations, which can be rearranged freely. Interactions include zooming and panning together with an option to preserve the temporal alignment of all views. Different zooming methods are supported, it is possible to switch between different temporal granularities (days, month, years), the user can click on an item (e.g. a month) on the timeline to zoom in, and a value range can be selected and displayed in an enlarged version inside a separate view. In addition a reference point can be defined on the time axis. Statistics for numerical variables can be displayed inside their panel, containing information like the minimum and the maximum value, or the standard deviation. Different visualizations are supported depending on the data type and abstractions, .e.g. line plots for numerical data types and the initially mentioned visualization for derived abstractions.

Midgaard [6, 7] supports the visualization of quantitative and qualitative data together with plans, which are displayed as background. The most prominent feature is semantic zooming of numerical data by switching through different visualization techniques. It uses colored bars on the lowest level of detail ("color-coded timelines"), this represents a different kind of abstraction visualization (compared to the one used in this work), where the calculated abstraction class is mapped to the color instead of a spatial dimension. The next level uses the color and the height to visualize the abstraction class, similar to bar charts. On the highest levels area charts and line charts are applied to show the data in full detail, showing the crossing into another qualitative class by a marker. It furthermore allows the visualization of single data points and high frequency data by applying an extended information mural technique (minimum, maximum, percentiles and median marked by curves) and connected turkey plots.

Techniques for Guideline Visualization

Depending on the tasks, which, for example, can be plan generation or plan execution, and on the target user group different visualization techniques are applied for guideline visualization [3].

Plan Strips [38] focus on the hierarchical properties of plans, i.e. Asbru plans, by applying nested strips. The plans are positioned on a horizontal axis representing the sequence of their execution. Parallel plans are placed on the same vertical position. Different colors are used to encode the execution order i.e. synchronization of plans, color saturation and brightness is increased for selected plans.

Glare [46] is a tool intended for guideline design and execution, it applies a graph (i.e. flow chart) like visualization to illustrate the structure of a guideline (cf. Figure 3.4). The nodes of the graph represent actions, whereas the edges represent the control flow between them. Different shapes and colors are used to encode the type of action, which can be a work action i.e. atomic action (similar to the concept designated by the term action in this thesis), which has to be executed, a query action, e.g. for a patient parameter, a conclusion, or a decision node i.e. conditions used to select between different paths of the guideline. The user can browse in a hierarchical list of actions (children compose parent actions) and inspect and draw the graph representing the guideline.

AsbruView [27] supports the visualization of Asbru plans, emphasizing temporal dimensions of Asbru plans together with their hierarchical structure (cf. Figure 3.4). It represents them as 3D diagrams using a "running track metaphor", showing different information on the width, height and depth dimensions. Subplans are stacked above a parent plan and parallel plans are placed beneath each other, time is mapped to the width i.e. the horizontal axis of the diagram.

Different metaphors are used to represent plan components. Conditions are represented by traffic signs/elements, filter-preconditions and setup preconditions symbols are placed near the starting point of a plan, the former is represented by a "no entrance with exception" sign and the latter by a turnpike (which can metaphorically open if a setup condition is reached). Traffic lights and a finish flag indicate the plan execution state, green means that the reactivate conditions got true, yellow means the suspend condition got true and red means that that the abort condition was hit, the finish flag marks the successful completion of a plan. The two points marking the start or the finishing line can be shifted depending on the latest or earliest start/finishing times.

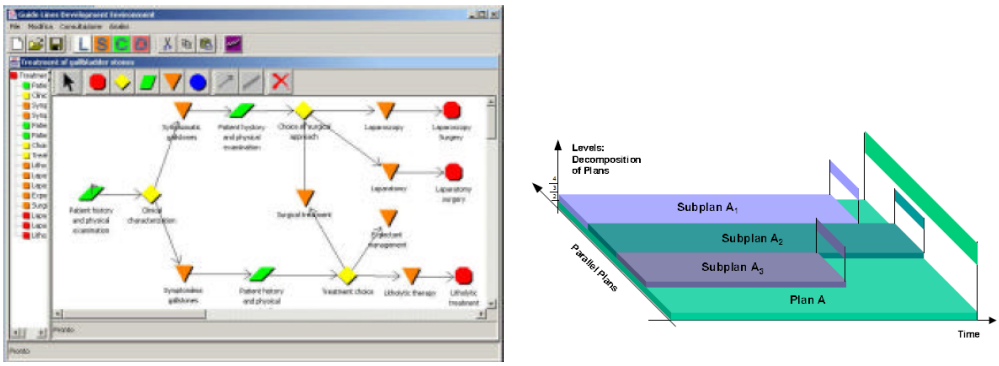


Figure 3.4: Interface of GLARE (on the left, image from [46]) and visualization of parallel plans in AsbruView (image from [27]), parallel subplans are stacked above a parent plan.

Event Visualization

Visualization of events plays an important role, when large amounts of data need to be processed. It is intended as a mechanism for data compression and aggregation, and helps to focus the user on important aspects of the data. Events in this context are often related to time-oriented data and highlighting is one possible visual mapping of events. The execution of valid and invalid actions and the appearance of missing action intervals, i.e. the occurrence of these three compliance types as they are defined in this thesis, can be considered as specific types of events related to the time-oriented parameters of a patient. The highlighting of invalid and valid action time points and highlighting of missing action intervals inside the parameter view can be seen as a specific kind of event visualization inside the parameter visualization of a patient, intended to draw the attention of the user to important episodes.

Tominski [15] defines a general conceptual model for the visualization of events and gives examples for possible mappings of events to visual attributes. Three main steps for event visualization are identified, event specification, detection of events and event representation (mapping to visualization). The suggested model for event visualization includes an event domain, which means the underlying data, the event type, i.e. selection criteria for interesting data/events and concrete event instances, which are matches for a certain event type. In addition to the formal model and a suggested approach for event visualization using predicate logic, different examples for the visualization of events are given by the author, distinguishing implicit and explicit visualizations. Implicit methods include highlighting or focusing to emphasize events in the data. An example for using highlighting and focus+context is given by the time wheel visualization, which was adapted by the authors. This technique shows the time axis in the middle and multiple parameter axes surrounding it, variables are connection lines between the time line and the parameter axes. In order to focus on a specific event, the view is rotated to the specific axis, the axis itself is stretched and the corresponding connection lines are highlighted (cf. Figure 3.5). Other examples include multiple scatterplots, matching instances are highlighted within them, and a map with a perpendicular time axis rising from each spatial point, which explicitly shows selected events.

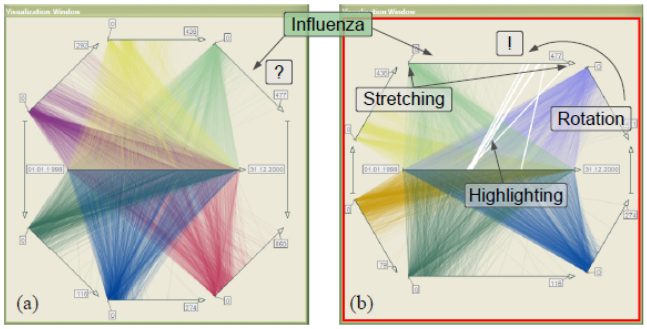


Figure 3.5: Event visualization example by highlighting and focusing the relevant instances (image with labels from [15]).

EventViewer [8] is an example for the explicit handling of events associated to time-oriented data, colored bars are used to mark events. It was developed to support the analysis of sensor data (from ocean observing systems). Events are generally defined to be binary (mapping of data to either true, in case of an event, or to false), the event model presented by the author includes four components, the observation data, transformations on this data (e.g. filtering), extraction conditions, which define if a data value belongs to an event or not (e.g. threshold) and archiving (start and end point of event and additional information stored).

EventViewer is meant to visualize these events. The authors organized the interface into panels, stacks and bands. A band is the basic element of the interface and represents a time line and colored bars show different events within them. Bands can be organized into stacks, and stacks can be organized into panels. Different categories can be assigned to them (e.g. time or location), which allows rearrangement for different analytic tasks (cf. Figure 3.6). The horizontal extend of the bars displayed in the bands represent the duration i.e. time of event occurrence, the whole vertical extend is used for coloring, whereas the color encodes the type of event. This kind of mapping has some similarities with the action and interval highlighting presented in this work, where action time points span across the whole vertical space of the parameter view, to emphasize their occurrence, time spans (missing intervals) are marked by the horizontal extend (time axis) of the highlighting and color encodes the compliance type (valid/invalid/missing).

EventFlow [31] is another framework dealing with the visualization of events, whereas the term event in this context rather denotes real events, which occur in the source data (medication duration, time point of diagnosis) , and not calculated events like in the previously mentioned EventViewer. EventFlow enables to query for certain patterns of point (e.g. heart stroke) and interval events (e.g. medication), it visualizes the individual events occurring for a single patient, as well as the aggregation of similar event patterns (cf. Figure 3.7). The visual appearance of point and interval patterns/events is in some way similar to the action and missing interval highlighting presented in this work (although the mapping is different). EventFlow displays aggregated point events as colored vertical bars, with the height representing the amount of

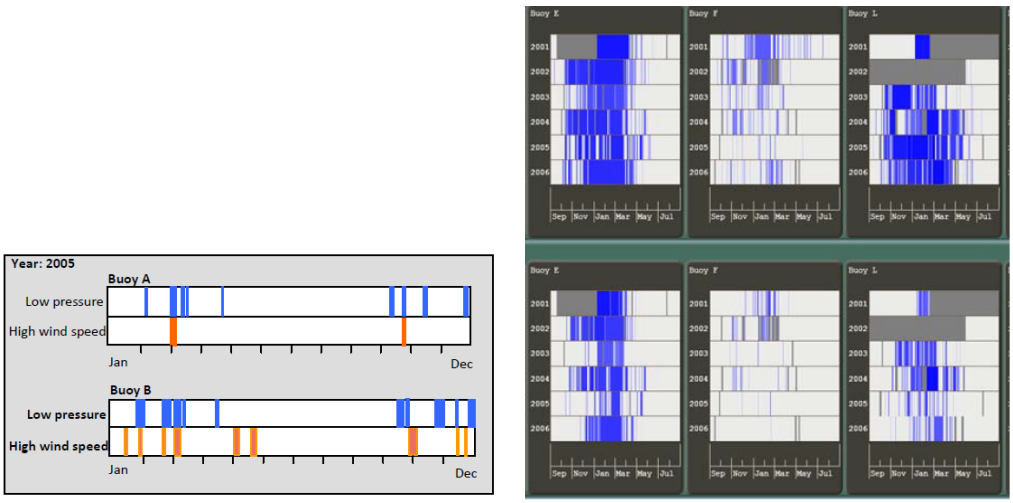


Figure 3.6: EventViewer's visualization of events inside bands representing the time axis (on the left), bands organized into stacks and panels (on the right), images from [8]

records with the same pattern. Intervals are mapped to colored rectangular bars, whereas the height conveys the amount of records with the same pattern and the length of the interval is mapped to the width.

Overlapping intervals are rendered by combining the colors of the overlapping areas, the authors also integrated means for the user to control the opacity of the interval fill color, to enable highlighting of overlapping intervals (e.g. to see if medication intervals overlap) and to merge multiple occurrences of the same event into one event. The authors furthermore identify different temporal relationships between interval and point events, which can be used for pattern based queries. Simple querying is supported by an interface based on drop-down menus, whereas complex queries can be set by arranging events of interest on a timeline.

Data Abstraction

The advantage in the use of data abstraction, which is also used for visualization in this work in order to emphasize the guideline related aspects of the patient parameters, is that knowledge specific to a certain task (e.g. determine if a value lies in normal range or is above or below a certain threshold, which implies the application of a certain action) can be integrated, this makes it easier to interpret the data in a certain context. Huge amounts of data from multiple sources can be compressed in this way and allow focusing on the important aspects of the data. Furthermore visualizations for qualitative data might be reused for abstractions of numerical data.

Data abstraction i.e. temporal data abstraction can be seen as a mapping from low level data to higher level concepts, which means a mapping of numerical to qualitative data values in most of the cases. On a higher level it is also possible to derive [44] more complex abstractions

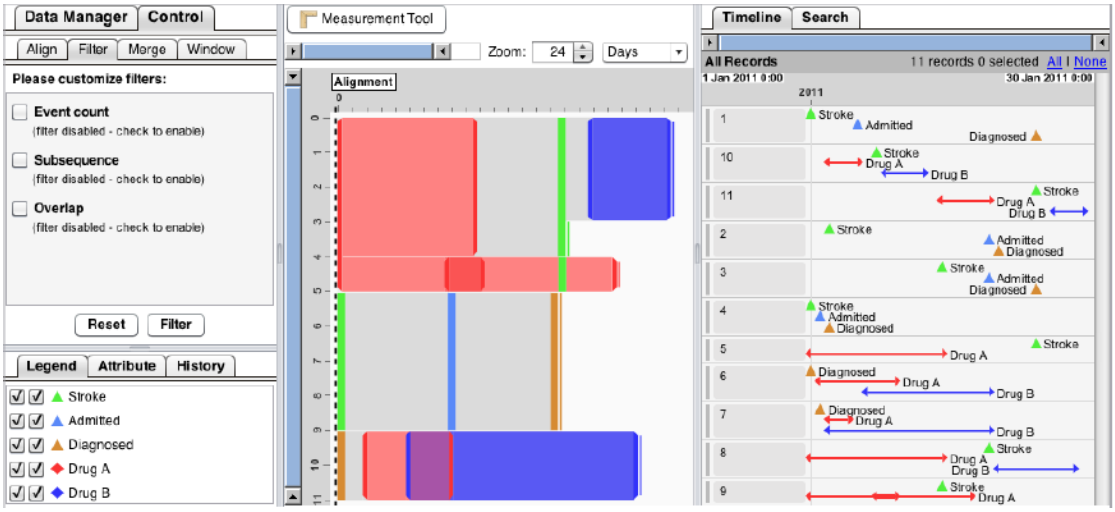


Figure 3.7: Interface of EventFlow, aggregated event patterns are displayed in the middle, the event view for single patients is placed on the right, image from [31].

based on a set of simple abstractions. Abstraction can be performed in several ways, Aigner et al. [5] categorize actions into vertical and horizontal abstractions. Vertical abstraction rather neglects the time axis and only considers values at a certain time point. Qualitative abstraction, which is a kind of simple vertical abstraction, for example maps a numerical value to a qualitative value (i.e. category), this kind of abstraction is also performed in this work by mapping patient parameter values in the categories low, normal and high for visualization (and implicitly by the guideline itself). Horizontal abstractions are more complex abstractions and involve the time axis; examples include trend abstraction (e.g. decreasing value) or the detection and categorization of reoccurring time events (periodic abstraction).

Abstraction is applied commonly in the medical field, not only for information visualization, but also for DSS systems (for example to perform reasoning on concepts derived from abstractions). Stacey et al. [44] review several systems, which apply advanced data abstraction. Some system use statistics and regression to preprocess and filter the data, in addition to complex data abstraction techniques. Complex pattern can be specified, allowing e.g. the detection of trends and periodic patterns in time dependent data. Applied methods, of the systems reviewed by the authors, include rule based systems (e.g. production rules) together with ontologies, temporal template matching (templates extracted from regression) or temporal constraint networks (events represented by vertices and time constraints represented by edges).

VIE-VENT uses temporal abstractions adapted to the context and derives high level trend patterns [30]. Smoothing is applied to avoid fast oscillations between qualitative categories, trend patterns are derived depending on the current qualitative category and slope thresholds. Examples of systems using data abstraction (Knave II and Midgaard) especially tailored for visualization, can be found in section Visualizations for Patient Data and Guidelines 3.1 of this chapter.

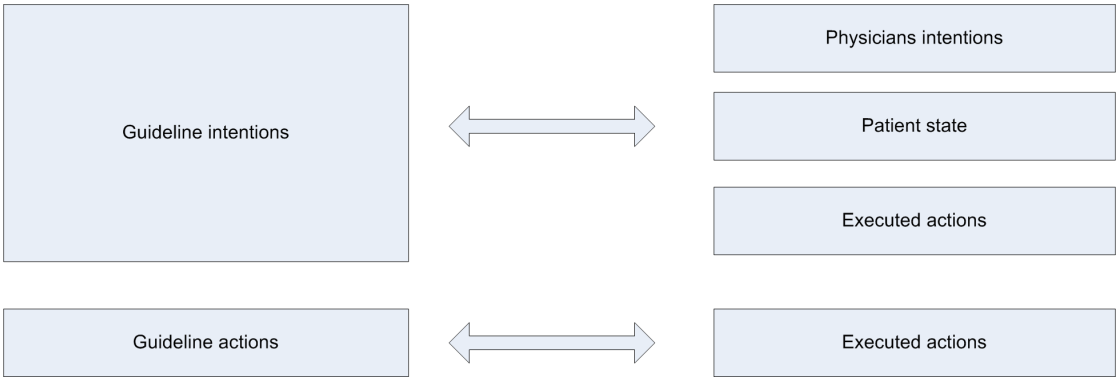


Figure 3.8: Illustration of different compliance analysis levels (based on the graphic in [1]).

Compliance analysis

Given a guideline, patient and treatment (i.e. execution data) data, compliance to the guideline specifications can be determined. Different levels of compliance analysis, which is also referred to as plan critiquing in this context, can be considered and various approaches exist in literature. Advani et al. [1] identify different levels of guideline based plan critiquing (cf. Figure 3.8). A possible measure for compliance can be, if the actions that have been taken in an actual treatment comply with the actions defined in the guideline. This can be done by comparing executed actions and prescribed guideline actions with each other, which is also the approach chosen in this work. On a higher level of interpretation the intentions of the guideline can also be taken into account (compared to the patient state, the actions, or the intention of the physician). An example for this is the comparison of the guideline intentions and prescriptions with the executed actions by taking the effects of actions into account, which means that some actions not explicitly stated in the guideline might still be allowed if they match the intentions. In general the level of compliance analysis to be chosen depends on various factors like the application scenario and the available knowledge and reasoning facilities.

Different approaches exist in literature for action based compliance analysis (i.e. compliance analysis based on the comparison of prescribed guideline actions and executed actions), applying different classification schemes for non-compliance actions. Although compliance analysis in this work is based on a less formal procedural approach, in order to focus on the visualization and ease the integration of the analysis results into the prototype, concepts and definitions of action compliance are partially inspired by the following work.

Chesani et al. [12] distinguish between events (i.e. actions) which should have happened and events that should not have happened in the execution according to a formal guideline specification. The guideline (cancer-screening care-flows) against which the execution is checked is modeled in a graphical language, which supports flow-chart like guideline specification. A formal language representation (SCIFF) is generated based on this specification, containing expectations about events, which can be checked against an execution log (cf. Figure 3.9). In

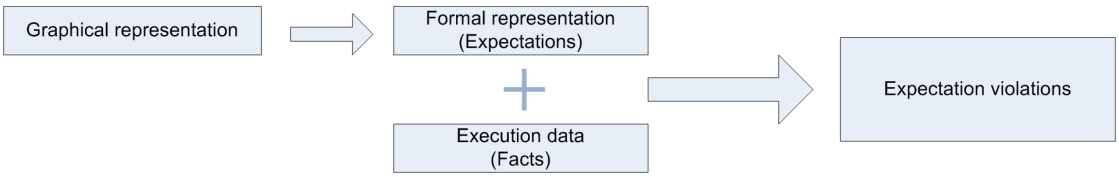


Figure 3.9: The process of compliance checking in [12] (based on information in [12])

particular, given a set of happened events in the log, expectations about events are generated based on the formal representation of the guideline, which are divided into expected events that should happen and expected events that should not happen (due to exclusive choices). Violations of these expectations can be reported for a particular execution.

In [21] a fine grained classification scheme for non-compliant actions is introduced. It distinguishes between:

- Actions unsupported by patient findings: This action type can be seen as an action which should not have been applied, because the necessary patient data was not available.
- Conflicting actions: Actions which should not have been applied, because the patient data is not compatible with the action application.
- Non-compliant order of actions: The actions were applied in the wrong order.
- Missing mandatory actions: Patient data suggests the application of an action, but it has not been executed

In addition to this classification scheme of non-compliant actions, the authors also introduce a scheme for non-compliant findings (e.g. wrong findings for an action), representing the opposite view, where the executed actions are valid, but the findings might be erroneous. Two different approaches for compliance checking are proposed, the first uses model checking, the second uses so called satisfaction sets, which can be generated for an action in advance.

3.2 Previous Work

This work is based on the architecture and visualizations of CareCruiser, which itself is build upon CareVis. A direct comparison of the original interface with the modifications introduced to the interface in course of this work is provided by in section 4.1 Overview of Interface Modifications. The following sections give an overview of the visualization and interaction techniques provided by CareVis and CareCruiser and explains the reasons for choosing this work as foundation.

CareVis

CareVis [4] was developed with the goal to combine views for patient observations, treatment data and computer-executable guidelines. The design was especially adapted for the needs of

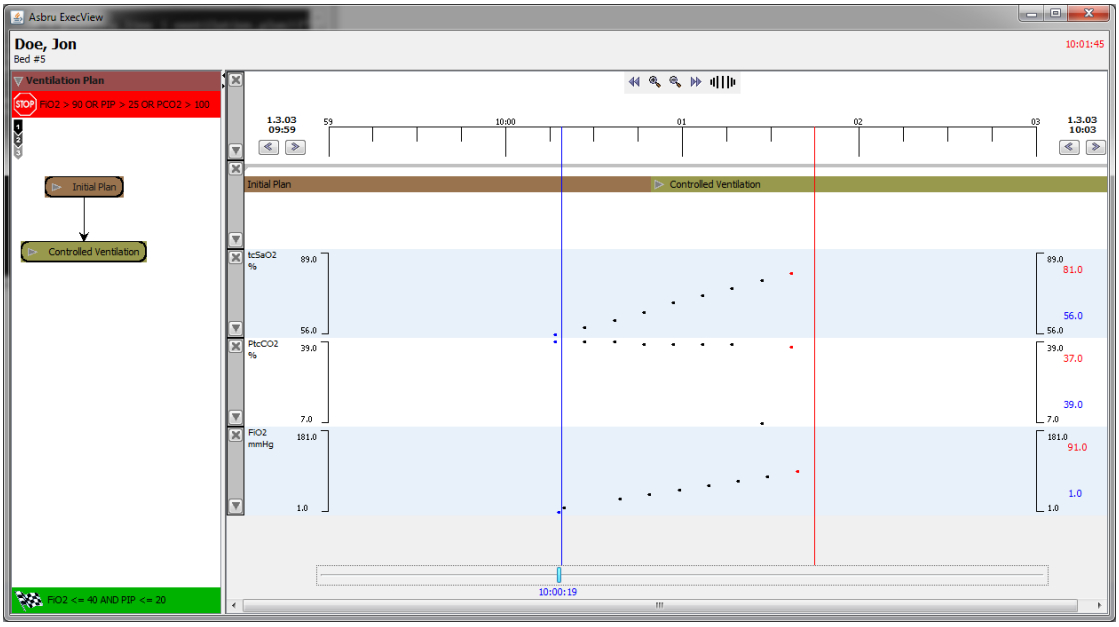


Figure 3.10: Interface of CareVis consisting of the logical view on the left, showing the guideline, and the temporal view on the right, showing plan execution data and parameters (image from screenshot of the demo prototype of CareVis).

physicians, who for example prefer flow-chart like guideline representations. The main interface consists of three separate views; a quick view panel, a logical view and a temporal view (cf. Figure 3.10).

On the top of the interface the quick view panel displays the current parameter values. The logical view on the left side displays the structure and elements of a guideline similar to a flow-chart. The body of an asbru plan is illustrated by different graphical elements, abort conditions are shown at the top and complete conditions are shown on the bottom of the plan display area. Subplans inside the plan body are rendered as rounded rectangles and if-then-else constructs are visualized as hexagons with arrows pointing to the alternatives. A special icon called execution sequence indicator encodes the execution order of the contained subplans. Two different interaction techniques are supported. The fisheye view causes elements not in focus to be displayed smaller. Overview+detail mode displays a hierarchical tree, showing the structure of the guideline, together with a detail view, showing the selected element.

The temporal view on the right side of the interface visualizes time-oriented data i.e. patient parameters and plan execution data. The display area is divided into facets, which can be minimized in order to obtain a shrunken view of the original displayed data. Furthermore a time cursor, spanning vertically over all temporal views, can be used to compare values occurring at the same point in time. Interaction methods like zooming and panning are supported by the temporal views. Plan execution data is depicted by bars (i.e. *Lifelines+*) having a length equal to the execution time of a plan, they can also be expanded to show the execution of children

i.e. subplans. Similar elements are used for feature, i.e. not already instantiated, plan data (*PlanningLines*), which show the temporal constraints of the anticipated plan.

The authors describe three evaluation steps that were performed, a user study to gather the requirements, expert reviews during the design phase and a qualitative prototype evaluation, where interviews were performed. It turned out that the prototype is appreciated as "clear, simple" and "helpful in working with and exploring treatment plans" [4, p.22].

CareCruiser

CareCruiser [22] is built upon CareVis and visualizes parameter and plan execution data of a patient together with the guideline. The authors state that it was designed with the goal to help in the exploration of effects resulting from clinical actions and to increase the quality of guidelines.

The executed actions during a treatment are mapped to diamond shapes with the color encoding the type of action (Action-ID), whereas the duration of an action is shown by whiskers. Executed actions and patient parameters are aligned together, to ease the detection of treatment effects. It is also possible to view multiple patient records together and to align action instances of a particular action for different patients. Actions instances of the same action type are highlighted, if an instance is selected.

A major feature of CareCruiser is the color-coded highlighting inside the parameter views. The user can choose to highlight medical observation data based on three types of information, the distance of the test values to the intended value, the progress from the initial value and the slope of the value sequence itself. According to the authors each type of highlighting serves different purposes. The slope is mapped to turquoise in the case of a negative drop and in brown for a positive rise of the values and intended to show the consequences of applied actions. The distance to the intended values is mapped to dark magenta color highlighting for extreme and light magenta for values in normal range (cf. Figure 3.11), it is designed to help to identify critical values at first sight. The third type of highlighting encodes the progress from the initial value, ranging from white for the initial value to dark blue for the intended value and dark red for diverging values, this type of highlighting is intended to show the success of the treatment.

Furthermore features like filtering and focus+context are supported. Filtering can be achieved via a range slider, which allows restricting the coloring to a certain value range. Focus+context is supported by a focus window, highlighting is only active inside the focus window. Additional interaction methods like zooming and panning along the time axis and tooltips for visual elements, e.g. actions and plans, are available.

The evaluation [23] has been performed in three steps, at first the requirements have been gathered with the help of a medical expert, in a later development state the authors conducted a heuristic evaluation with experts, who evaluated the prototype based on a list of usability heuristics. User testing with physicians was performed for the summative evaluation of CareCruiser, which was appreciated and well-received by the physicians.

In addition to CareCruisers many other features, the highlighting of the distance of the parameter values to the intended values and progress from initial value can be already seen as a specific kind of compliance visualization, which is also supported by the alignment of the actions together with the parameter values. In contrast to compliance in terms of the patients in-

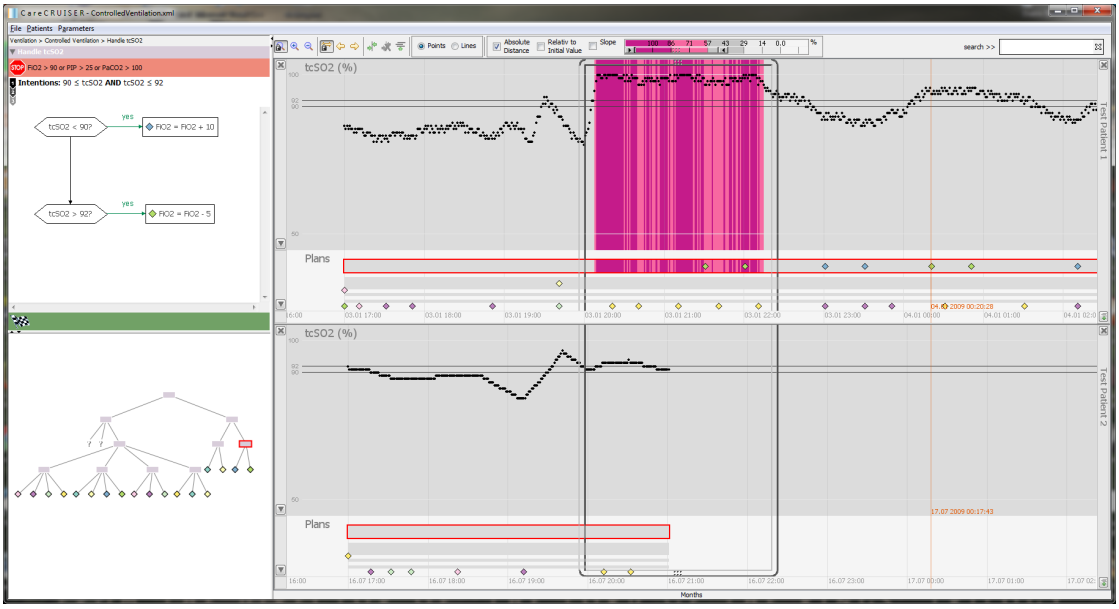


Figure 3.11: The interface of CareCruiser, the logical structure i.e. guideline semantics are shown on the left side, actions are mapped to colored diamond shapes. The right side shows the temporal view with actions aligned to the patient's parameters. Values diverging critically from the intended values are highlighted inside the focus window (screenshot taken from CareCruiser prototype).

tended state to the patient's actual state, the work presented in this thesis is focused on a different level of compliance i.e. action compliance (compliance of executed actions with the guidelines prescribed actions).

Reasons for Selecting CareCruiser

CareCruiser was selected as basis for the extensions worked out in course of this thesis, because it provides suitable views for patient, treatment and guideline data. Actions and patient data (observations) are aligned beneath each other, this joint view is appropriate, because action compliance and patient data are closely entangled to each other. Inheriting this kind of views provided a good foundation to implement additional information about action compliance with a clinical guideline. Understanding the guideline structure and inspecting it is another important feature for an expert, performing the analysis, it requires additional views as they are available in CareCruiser.

CHAPTER 4

Implementation and Visualizations

This chapter outlines the extension made to the prototype, including the different visual mappings and encodings for compliance information. An overview of the modified interface is given (section 4.1 Overview of Interface Modifications), followed by a more detailed description (section 4.2 Visual Artifacts and Encodings). A technical overview of CareCruiser and the modules developed in course of this work are presented in the documentation (section 4.4 Overview of Interface Modifications).

4.1 Overview of Interface Modifications

This section gives an overview of interface modifications and visual encodings for guideline compliance implemented in the prototype. Figure 4.1 shows the original prototype and Figure 4.2 shows the extended interface. Changes have been made to the patient parameter view, the plan execution view and the plan overview, in order to visualize the results of compliance analysis. Furthermore a view showing patient information, statistics and settings has been integrated. Please refer to the following sections for a more detailed explanation of introduced visual encodings and modifications.

4.2 Visual Artifacts and Encodings

This section introduces the different visual artifacts and encodings considering compliance that can be found in the visualizations. Three different types of encodings related to action compliance are present in the interface, depending on the patterns to reveal and analytical task to support. The first type of encoding is directly related to action instances and can be found in the plan execution view; instances of valid/invalid actions and missing action intervals are represented here. The second type can be found in the parameter views i.e. views for parameters that are connected to the application of an action inside a ventilation plan, time points for valid/invalid actions and the time span of missing action intervals are represented here. Statistics and the

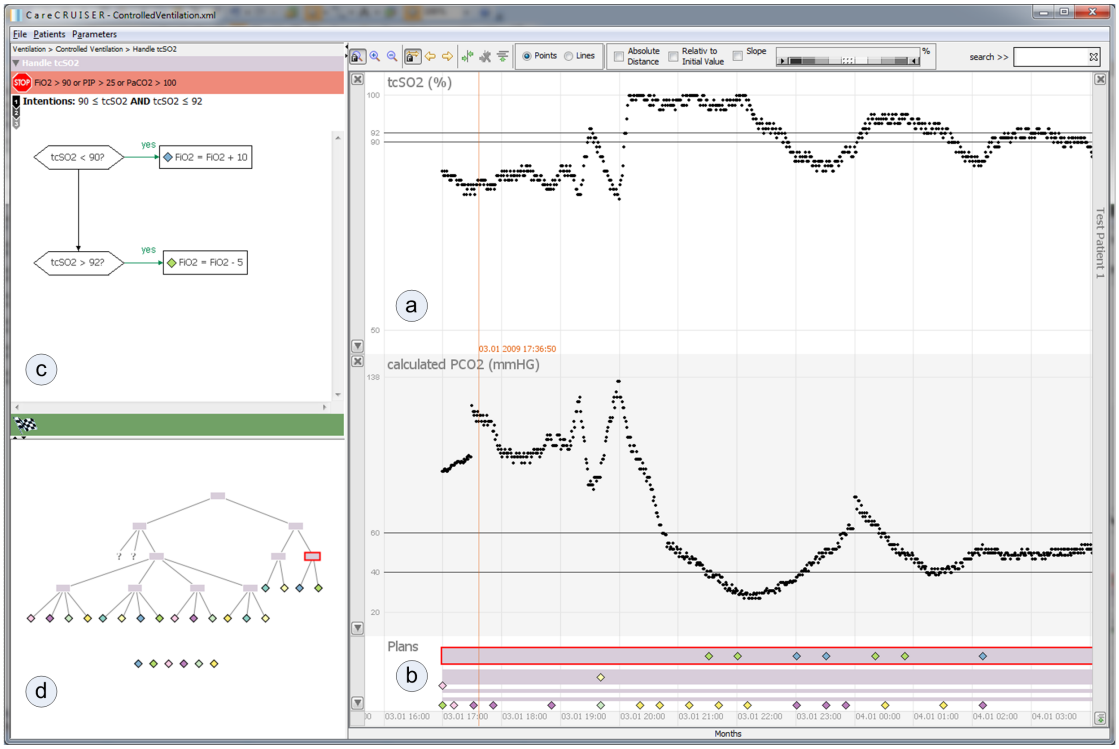


Figure 4.1: Original CareCruiser interface: The main interface is separated into views showing the structure of the guideline (c,d) and views showing time-dependent execution data for a specific patient (a,b). (a) Time-oriented parameter view: Parameter data is plotted against the time axis, the boundaries for normal value range and the minimum and maximum value are also shown. A major feature is the color coded highlighting to show e.g. the distance from the intended values (it is not shown in the screenshot to ease the comparison to the modified interface, see Previous Work 3.2 section). (b) Plan execution view: The plans and actions executed during the patients treatment are shown here, actions are visualized as diamonds, with the color encoding the action-ID. The execution data is aligned together with the patient parameters to allow the analysis of effects resulting from action execution (c) Logical detail view: This view offers a detailed representation of the currently selected plan, showing, for example, action and abort conditions (d) Logical overview: An overview of the structure and relationships between the plans and actions of the guideline is represented here. As soon as an element is selected here, it is also selected in the other views and vice-versa.

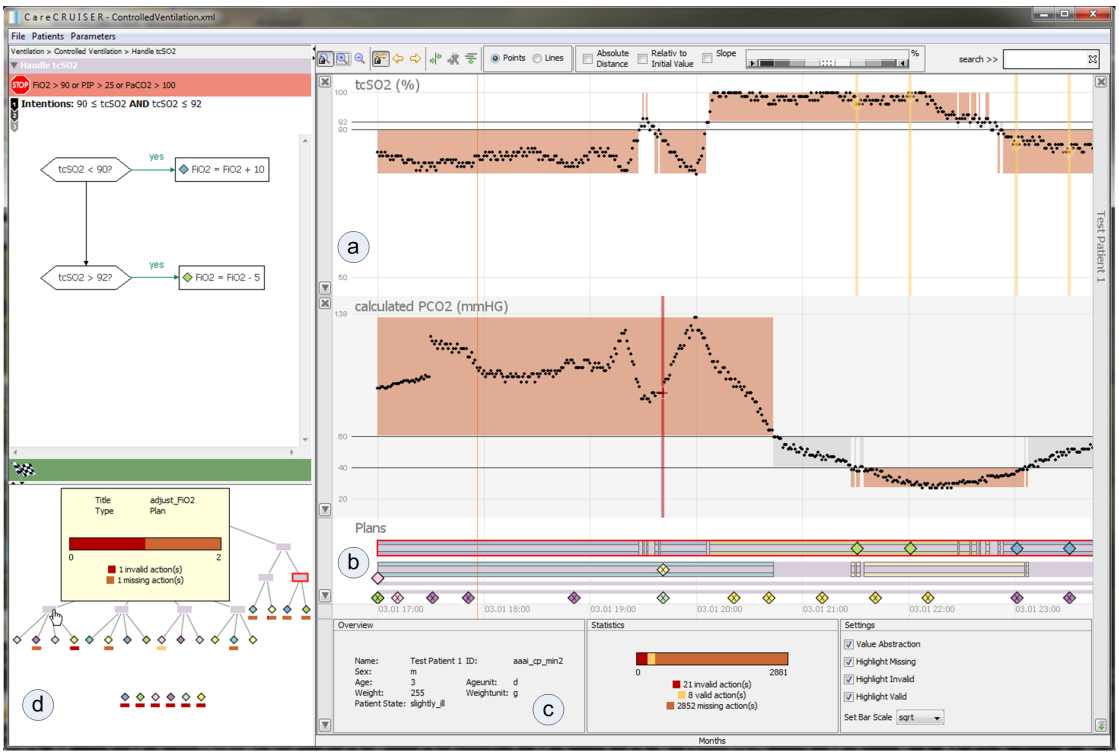


Figure 4.2: Screenshot showing the interface modifications. (a) Data abstraction into three categories (high, normal, low) is integrated into the parameter views. Highlighting of valid and invalid action time points and missing action interval time spans for relevant parameters is supported, making it easier to analyze the relationship between compliant/non-compliant execution and corresponding values. (b) The modified plan execution view, invalid actions are marked accordingly (labeled with "X") and intervals i.e. time spans where the execution of an action is missing, are encoded by a bar consisting of an upper and lower part. (c) Pane containing patient information, options and statistics. Features like highlighting or data abstraction can be turned on and off in the settings. The statistics contain a stacked bar, showing the proportions between valid, invalid, and missing action counts for the whole execution, together with a legend, which also displays the absolute counts (d) Aggregation of valid, invalid, and missing action counts is implemented in the overview. A stacked bar showing the aggregated counts for the current element (action or plan) is displayed in a tooltip. Miniature versions of a stacked bar are rendered directly beneath the action diamonds to allow the immediate identification of actions, which have valid, invalid or missing instances.

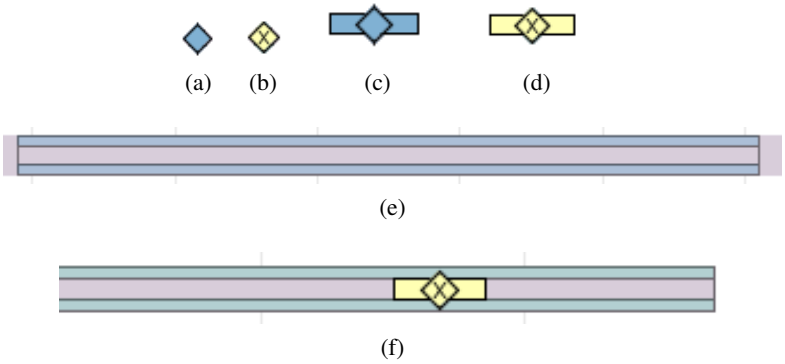


Figure 4.3: The different encodings for action instances in the plan execution view. (a) The action instance itself i.e. the point in time of their application is marked by a diamond, the color encodes the type of action/ the action-ID. (this is inherited from CareCruiser, only the size is slightly increased) (b) Invalid actions are marked by an "X".(c) A valid action and an (d) invalid action and a with a time span are shown here, this is represented by the colored horizontal bars spanning from the diamond. Invalid actions with a time span are marked with an "X", similar to invalid actions without a time span. (e) A missing interval i.e. the time span, where an action should have been executed, but was not applied.(f) An invalid action with duration is occurring at the same time another valid action is missing.

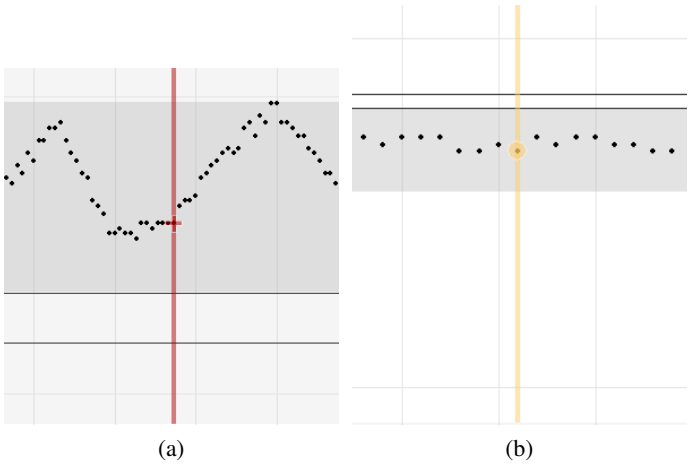
structural plan overview contain the last type of encoding, the counts of valid/invalid/missing actions are represent as a stacked bar.

Action Instances and Missing Action Intervals

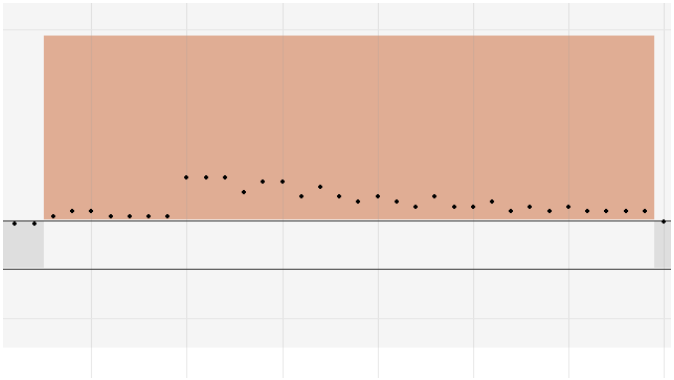
The instances of executed actions are shown in the plan execution view, valid executed actions are visualized in the same way as they are in the original CareCruiser, with the color encoding the action type, invalid executed actions are additionally marked with "X" (cf. figure 4.3). All user executed actions have a similar visual appearance therefore, no matter if they are valid or invalid, this allows separating them visually from missing intervals, which are generated by compliance checking. Contrary to executed action instances, time intervals, during which the execution of an action is missing, are displayed as two connected upper and lower parallel bars; this enables to differentiate them from action durations (which are also time intervals) and to display action durations and missing intervals at the same time.

Time point and time span highlighting

Time points of executed valid and invalid actions and time spans of missing action intervals can be highlighted in their corresponding parameter view (see figure 4.4). A time point is not only marked by a shape which depends on the compliance type of the executed action (valid/invalid), but is also highlighted by a vertical bar spanning from top to bottom. The shape marks the



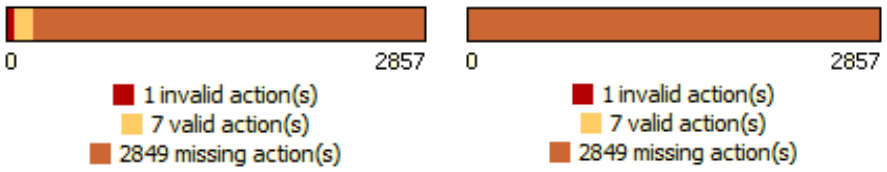
(a) (b)



(c)

Figure 4.4: (a) Time point highlighting for an invalid executed action: The time point is marked by the corresponding shape and highlighted in dark red color. (b) Time point highlighting for a valid executed action: The circular shape and the yellow color represent the compliance type valid. (c) Time span of missing action execution is highlighted inside an abstraction rectangle. In this example, the execution of an action related to high parameter values is missing during the highlighted time span.

parameter value occurring at the highlighted action time point. The purpose of the bar is to enable quick identification of executed action time points together with the parameter values related to them. Intervals of missing action application i.e. time spans are highlighted inside the rectangles of the data abstraction; this allows to quickly see if guideline complaint actions were performed during critical patient states, or if they have been missing. The color scheme of highlighted time points and intervals encodes the compliance type. Highlighting is not fully opaque; this allows seeing the underlying parameter samples.



(a) (b)

Figure 4.5: (a) Stacked bar showing the proportion of the counts of all three compliance types, square root transformation is active in this example, emphasizing low counts, which would be hidden otherwise. (b) The same bar using a linear transformation, resulting in sub pixel width for invalid and valid counts.

Stacked Bar

The counts for different compliance types i.e. valid, invalid and missing action counts, are shown by a stacked bar (cf. figure 4.5) inside the statistics, as tooltips and miniaturized beneath the action nodes in the structural overview. Color coding is used for the compliance type using the same colors as for highlighting inside the parameter views (yellow - valid, light brown – missing, red – invalid). The amount of space taken by the elements of a stacked bar is equal to the proportion of counts for this compliance type in relation to the total count. Without the need of processing numbers, one can quickly estimate the proportion of counts in relation to each other, in addition the exact counts are provided by the legend belonging to the bar. In order to avoid that high counts for a certain compliance type take up too much space to keep the other counts visible, it is possible to use a square root transformation for the counts.

4.3 Temporal View Extensions

This section describes the visualizations i.e. modifications introduced to the temporal views of CareCruiser. This includes parameter views, showing the patient parameters related to ventilation over time and the plan execution view, which displays the executed actions instances on the time axis.

Plan execution view

In order to visualize compliance of the executed actions with the guidelines prescribed actions in the plan execution view (see figure 4.6), the actions are checked and rendered with an “x” label, if they are invalid i.e. applied not accordingly. Time spans, during which the execution of an action is missing, are represented as interval. The use of intervals instead of the generation of “individual” missing actions has several reasons. Repeating plans with a short delay time (e.g. a few seconds) can lead to the generation of a lot of missing actions, even for short time spans of missing action execution, and to visual clutter. Aggregation into intervals avoids this issue. The

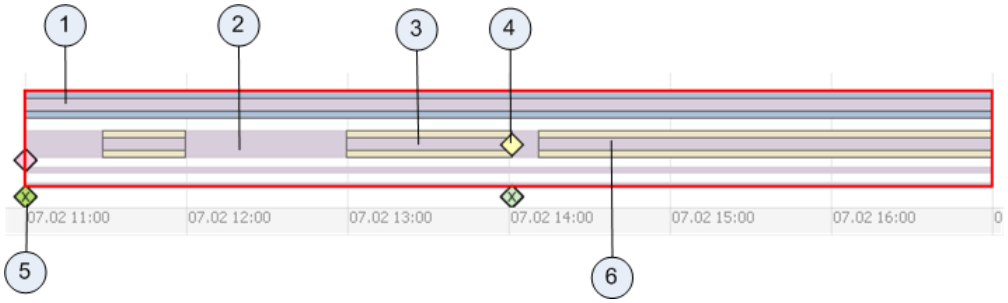
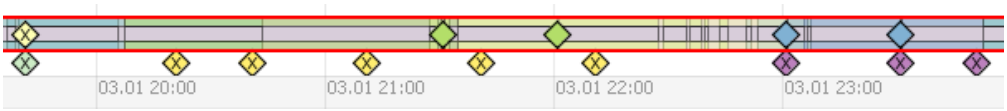
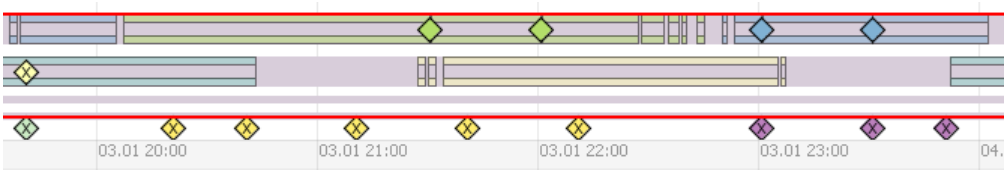


Figure 4.6: Plan execution view for an example patient, the execution/compliance behavior of the ventilation plans is displayed separately for the plan handling the oxygen saturation (1) and the plan handling the partial pressure of carbon dioxide (2). The action for adjusting low oxygen saturation is missing for the whole duration of the execution in the first plan. The other plan (2) has some time spans, during which the action for adjusting low values is missing. As soon as the valid action is applied (4), the missing interval ends (3). A new interval starts afterwards (6), because no valid action has been applied, although the parameter is still beneath normal range. Actions without an equivalent in the guideline (5) are automatically marked as invalid actions.

other reason that the exact time point of valid action execution is depending on user choice (i.e. the action can be applied anytime in the interval between the minimum and maximum delay). Individual missing actions generated for the visualization would only represent one possible solution out of a vast amount of applicable solutions (or infinite for a continuous time scale). The actions and missing intervals are rendered with transparency, in case the ventilation plans are viewed in collapsed form (cf. figure 4.7), this gives a good overall impression of compliance behavior over time (although the more common use case might be the detailed inspection of the individual ventilation plans). Tooltips have been adapted from the original prototype to show information relevant for compliance analysis (see figure 4.8). The start, end and duration, the type (i.e. regular valid action, regular invalid action, irregular invalid action and missing action interval), the title and the count of elements of the same ID and same type is shown.

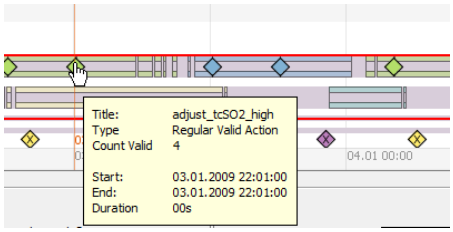


(a)

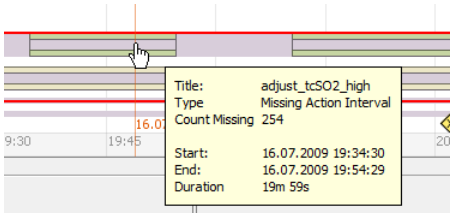


(b)

Figure 4.7: Another execution example: (a) Collapsed view for the two ventilation plans, it is easy to see that actions are missing over the whole duration for at least one ventilation plan. (b) The same view showing the execution of the plans separately (the gap between valid actions and the next missing interval is barely visible in this case, because the delay time span is very small).



(a)



(b)

Figure 4.8: Examples for the altered tooltip in the plan execution view. (a) Action tooltip, in this example the adjust_tcSO2_high action has been applied validly four times, which is displayed by the count valid attribute of the tooltip. (b) Tooltip for a missing action interval, the value of the count missing attribute is estimated from the length of the interval.

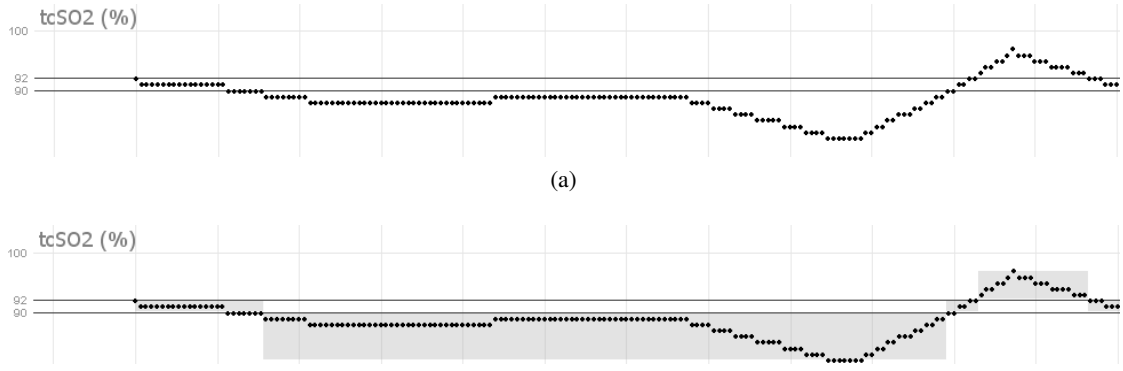


Figure 4.9: (a) The original parameter view inherited from Care Cruiser, with time on the horizontal and parameter values on the vertical axis. (b) Abstraction is set to active.

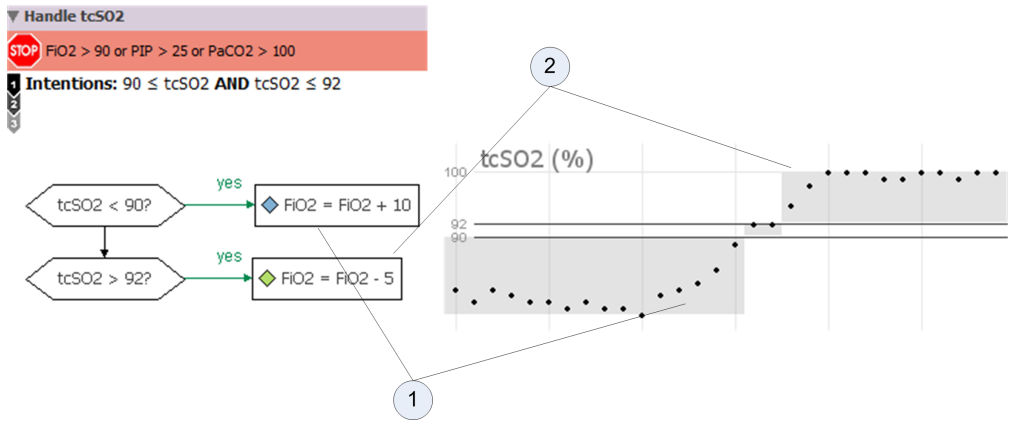


Figure 4.10: Values of the SO2 parameter mapped to the category high are associated with the application of a specific action (1), the same is true for values mapped to low category (2).

Parameter View

The parameter views are extended to enable the identification of the relation between valid and invalid action time points and missing action intervals and their corresponding parameter (i.e. the parameter, which is evaluated in the action condition). Abstraction into three categorical classes (based on the specified risk levels) normal, low and high is introduced (cf. figure 4.9), the visual abstraction allow to see immediately, if a value is above or below normal range. In case a parameter is related to the execution of an action (PCO2 and SO2), this means, that falling into low or high categorical class goes hand in hand with an action condition getting fulfilled (see figure 4.10), this allows to see at one glance if a value gets critical and requires action execution. The abstraction is transparent and also allows to see the parameter dot plot shining through; it is intended to support cognitive pattern recognition without hiding the details of the underlying data.

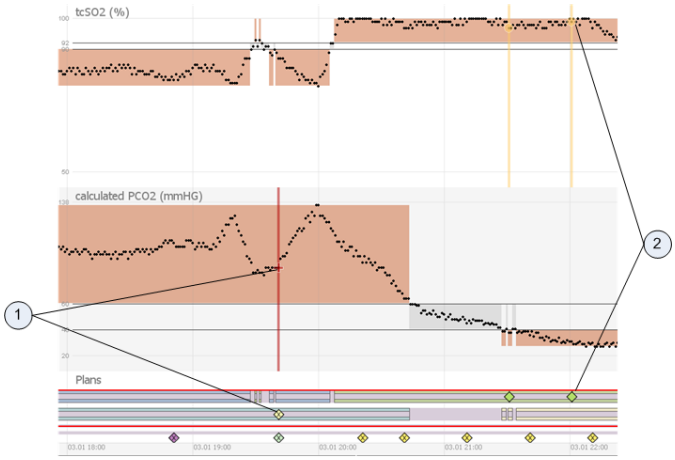


Figure 4.11: Example execution with active parameter view highlighting for valid and invalid actions, and missing action intervals. Invalid and valid action time points, and missing action intervals can be identified quickly without the need to manually refer to the plan execution view. An invalid action time point is highlighted inside the view for the parameter PCO2, the corresponding invalid action instance of the plan handling the PCO2 parameter is also visible in the plan execution view (1). Two valid actions are highlighted inside the SO2 parameter view, with their counterparts in the plan execution view (2).

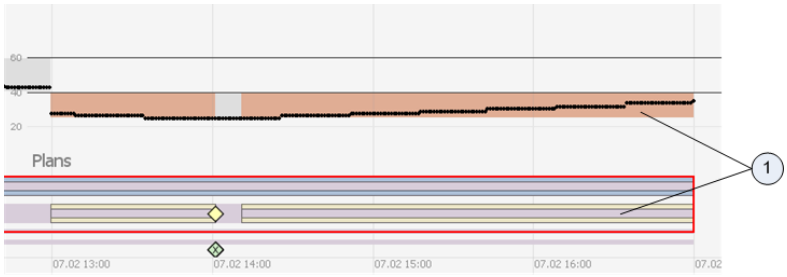


Figure 4.12: Highlighting of two generated missing action intervals in the parameter view and their counterparts in the plan execution view (1) is shown. The short time span, i.e. gap, between the two intervals appears due to the valid action, which has been applied during the execution (i.e. treatment of the patient).

Highlighting of valid and invalid action time points is supported in the parameter view associated to the action condition of a plan (see figure 4.11). Without the need to switch between plan execution view and parameter view, one can directly see the consequences of valid or invalid action execution in the patient parameter view. Highlighting of missing intervals (cf. figure 4.12) helps to identify the associated parameter values and, for example, to see how the parameter values are progressing without the proper action applied and how serious the absence of the action influences the patient state. Parameter values can be associated with compliance or non-compliance in a direct way.

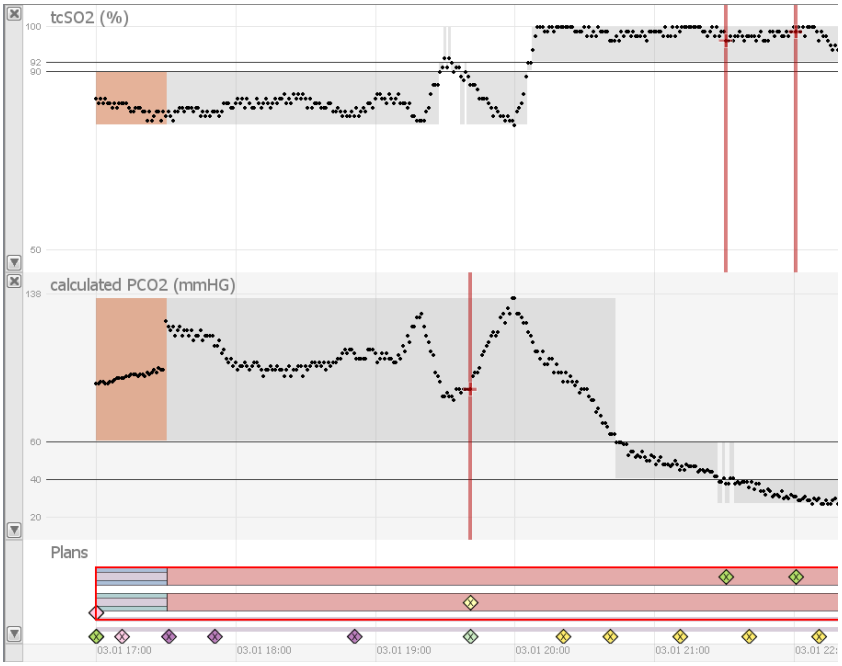


Figure 4.13: The duration of halted execution, starting from time point of change to aborted state until the end of execution, is colored in red. All actions occurring in this time span are marked as invalid; this is also reflected in the highlighting inside the parameter views.

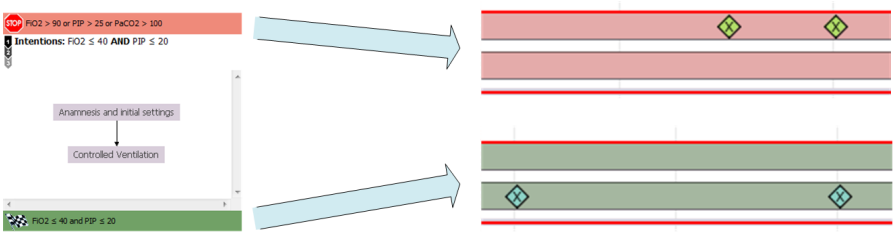


Figure 4.14: The coloring used to mark abort (red) and complete plan (green) state duration corresponds to CareCruiser's coloring of conditions in the logical detail view.

Abort and Complete Conditions

It is possible to optionally regard abort and complete conditions, which are present in the guideline (by passing command line options for program execution). The semantics of abort and complete conditions are different: the former means the plan failed, whereas the latter means the plan finished successfully. On the execution level the results are the same, as the plan(s) are halted as soon as the condition is true, all subsequent actions are set to invalid and open missing action intervals are closed. Colors are used to mark the time span of halted execution, from the beginning of the abort or complete state transition until the end of execution (see figure 4.13 and 4.14).

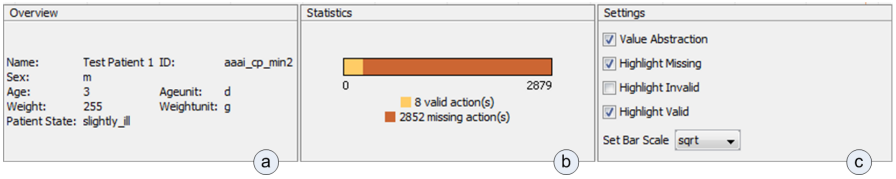


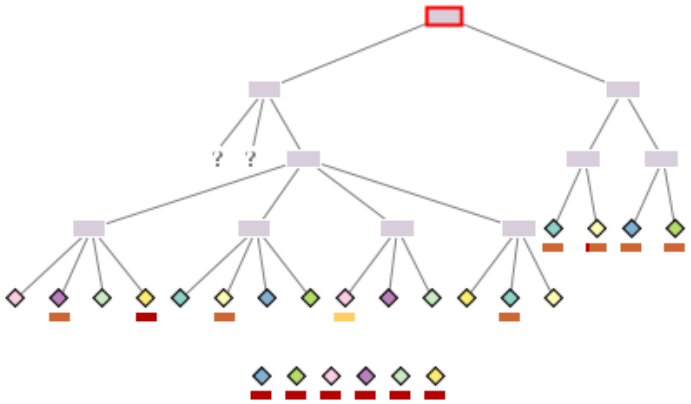
Figure 4.15: View for an example patient, showing patient overview information (a), statistics (b) and settings (c), missing and valid action highlighting is active, whereas invalid action highlighting is deactivated. In this example it is easy to see that the amount of missing actions far exceeds the amount of valid actions.

Options and Statistics View

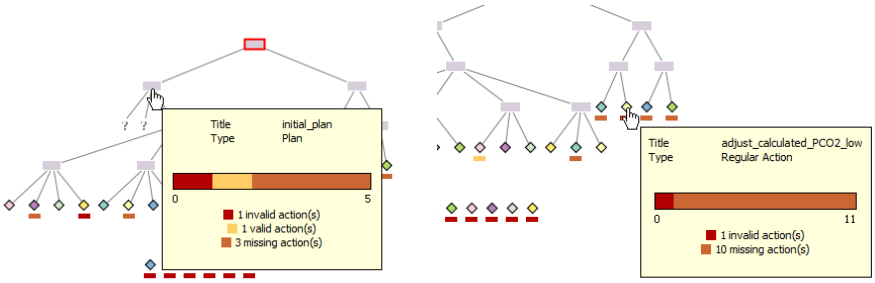
The options, additional patient information and statistics are displayed in a separate view (see figure 4.15). It is possible to adjust different options for the visualizations inside the settings panel. The interface allows to enable and disable action time point and missing action interval highlighting, as well as the activation/deactivation of data abstraction inside the parameter views. In addition the scale of the bars, showing the compliance counts, can be set to linear or square root (default). All changes in the settings immediately trigger an update of the visualizations. Statistics (see figure 4.15) give an overview of the overall counts of the three different compliance types (i.e. valid, invalid and missing actions) for the whole execution. The bar only shows counts for the compliance types, which have highlighting set to active in the settings. A legend beneath the bar shows color mapping and absolute counts. Statistics allow getting a quick overview of the compliance of executed treatment with the guideline. The bar helps to estimate the proportion of counts in relation to each other, for example the amount of missing actions can be compared against the amount of valid actions.

CareCruiser already offers a separate view, giving an overview of the guideline structure. It shows the hierarchical relationships of the plans contained in the guideline and the actions belonging to the guideline, as well as irregular actions (which do not belong to the guideline). The aggregated counts can be inspected for each entity i.e. action or plan. Altered tooltips show an enlarged version of the stacked bar for the selected entity. Miniature versions of these bars are rendered beneath each action (cf. figure 4.16). The tooltips show information similar to the statistics section of the options and statistics view, the difference is that tooltips for plan or action entities in the overview show the aggregated counts of the selected entity and not the overall counts. For an action this means that only valid, invalid or missing action instances with the corresponding action ID are counted, whereas for plans this means that the counts of all actions in the plan itself and for all actions contained in his subplans are aggregated together. Furthermore the name and the type of the element are displayed inside the tooltip.

Providing an overview of the compliance counts of each action is the goal of the miniature bars, which are rendered beneath the action symbols. They are intended to give a quick impression of the overall action compliance and to enable to see if an instance of an action e.g. has been applied invalidly once in the execution.



(a)



(b) (c)

Figure 4.16: (a) Plan overview with miniature bars beneath the action symbols: It is clearly visible e.g. that a lot of guideline action has been applied invalidly at least once (red bar) and that a lot of guideline actions are missing at least once (brown bar). (b) A tooltip for the initial plan shows that most of the actions of the child plans are missing in this example. (c) Tooltip for the action, which is applied, if the PCO2 Parameter is low, in this example the action is missing frequently and has one invalidly executed instance.

4.4 Documentation

This section presents a high level overview of the architecture of CareCruiser and extensions of it implemented in the course of this work, with emphasis on the functionality needed to understand the modifications. The overall architecture of CareCruiser is based on prefuse, which means that the functionality of the visualizations is separated into data, display, visualization, action, control and renderer classes. Data classes are reader classes responsible for reading the data structures and classes holding data structures for the visualizations. Actions perform the visual mappings and layout by accessing the data structures (setting properties of visual items, which are backed by data structures) and controls are responsible for interactions and transformations. Different renderers can be used to determine the visual appearance of visual items. A visualization class is managing visual items, actions and renderers, whereas displays are in charge of providing a view to the visualization and therefore determine view size and control operations, which can be performed (also see section 2.5).

Initialization at Startup

The code of CareCruiser is subdivided into packages. `Asbruflow.main` contains `Main.class`, which is executed on startup and `asbruflow.java`, which is responsible for loading the data and view initialization. The path containing the patient data is predefined in a configuration file, as well as the patient data and parameters that should be displayed or be available for selection. At first the necessary plan, execution and patient data is loaded. The graph structure for the visualization of the logical detail (`AsbruReader.java`, `OverviewGraphCreater.java`) and the logical overview is generated. Patient data and plan (action) execution data is loaded into table structures subsequently (the graph for the visualization of plan execution is created later on, using the table containing the execution data). After these steps the Swing based interface is set up, a main Swing pane together with a menu bar is created at first. The UI structures and visualizations for the logical and temporal (parameter and execution data) views are created afterwards. The logical views, i.e. logical detail and logical overview display (defined in `LogicalDisplay.java` and `LogicalOverviewDisplay.java`), are separated by `JSplitPanes` and embedded in a `JPanel` structure. The temporal (patient) views are embedded in their own separate panel structure. `PatientFacet.java` contains all GUI elements for a patient; it holds a `TimeView.java` instance, which furthermore holds all parameter views i.e. `ParameterDisplay.java` instances and the plan execution view (`PlanDisplay.java` instance).

Logical views

Data model classes (graphs) provide the underlying data model for visualization (`SingleStep`, `SingleStepGraph`, `SingleStepRelation`). The initialization of these data models is done by reader classes in the package `asbruflow.io`, which create the graph structure (e.g. `AsbruReader`, `OverviewGraphCreator`). Package `asbruflow.ui.logicalView` contains the display and visualization classes for the logical views. The plan (detail) view consists of a panel (`LogicalPlanViewPanel`) defining the visual layout of the border of

the view. Functions for controlling the view, like panning, zooming and expanding the plan are contained in the corresponding display class (`LogicalDisplay.java`). `LogicalVisualization.java` determines the rendering and the layout, and offers functions for highlighting and expanding; furthermore it contains actions for zoom in and zoom out animations. The interactions/controls for the logical overview are assigned in `LogicalOverviewDisplay`.

`LogicalOverviewVisualization` assigns renderers, methods for highlighting and actions; including the layout, i.e. tree layout for regular nodes and a horizontal irregular layout for non-guideline nodes. Some controls and layout actions are defined in external classes; `asbruflow.ui.Layout` contains `FullLogicalLayout` used to layout the plan view. `Asbruflow.ui.control` contains customized control classes, e.g. for expanding a plan (`ExpandedStateToggler.java`).The package `Asbruflow.renderer` contains custom renders for the logical views, for example `ifThenElseRender.java`, which customizes the rendering of the condition nodes in the logical detail view, or `OverviewShapeRenderer`, responsible for rendering the nodes in the logical overview.

Temporal Views

Package `asbruflow.ui` contains the main UI classes for the temporal views. The temporal views for a single patient are embedded in `PatientFacet.java`, this facet is responsible for e.g. handling the close button (for the whole patient facet) and drawing the patient name. Its content is defined by `TimeView.java` container, which manages the different time-oriented views (parameter and plan execution) for a single patient. Supported operations are implemented for adding or removing facets (views), minimizing or expanding the views and adding the time scale header. The class `TimeViewDisplay` is the basic class for all temporal views, for parameter views as well as for the plan execution view. Controls/interactions in common for all temporal views are created here, including pan and zoom controls. The pan control itself is defined in `PanAndLensControl.java` (package `asbruflow.ui.control`). `TimeViewVisualization.java` is the basic class for all temporal visualizations. All temporal views are embedded in a container class (`LeftFacet.java`), adding buttons and functionality for closing, expanding and minimizing the views.

Plan Execution View

In order to create the data (graph) structure for the execution view, an instance of `PlanExecutionGraphCreator` (`absruflow.io`) is supplied with the graph containing the logical structure and the data about patient plan execution. The specific controls for the plan execution view are set up in `PlanDisplay.java`, a mouse hover control is specified to enable the display of tooltips for visual items and a control for handling the expansion of a plan is added (defined in `TimeExpandedStateToggler.java`). `PlanVisualization.java` is delegating rendering, layout and actions, and handles highlighting of visual items with the same plan id, furthermore a method for plan expansion is defined. The rendering for the plan nodes (parent plans i.e. nodes which are not user performed actions) is handled by `ActivityNodeRenderer.java`, the rendering for the actions (diamonds) is handled by `UserPerformed-`

PlanRenderer.java, located in the package asbruflow.renderer. Horizontal layout is performed by the external class TimeLayout.java, which places the items on the x-axis, depending on the time scale used. Vertical layout is performed by ActivityLayout.java (package asbruflow.ui.layout).

Parameters Views

A prefuse table is backing the data for the patient parameter views (see package asbruflow.parameter.data). Package asbruflow.parameter.ui contains ParameterDisplay.java, taking responsibilities like creating a hover control for tooltips (showing the parameter values). Visualization is managed by ParameterVisualization.java. Prefuse AxisRenderer instances are used for rendering the horizontal and vertical axis, a custom renderer is used (MyYAxisRenderer.java) to show the risk levels, a prefuse ShapeRenderer is used to render the actual data values. The horizontal layout is determined by the class TimeLayout, whereas the vertical layout is handled by a prefuse AxisLayout action.

Linking and Brushing

EventRouter.java is a mediator class responsible for managing and delegating different events like highlighting or repainting to the various views. It notifies other temporal patient views, if a zooming or panning event happens. Highlighting and plan expansion across different views is supported by corresponding methods. Alignment and repaint events can be propagated between different views.

Technical Overview of Extensions

This section provides insight into the technical realization of the extensions implemented in CareCruiser in the course of this work.

Analysis of Compliance

ComplianceProcessor.java prepares the data structures for compliance analysis. At first a hierarchical plan structure containing all necessary information (action conditions, plan conditions, repeat specifications and plan hierarchy) for compliance checking is built (ComplianceModel.java), also see Figure 4.17 . The data (graph) about the executed actions is also handed to this module, and the executed actions are sorted by date (timestamp). The actual procedural compliance analysis takes place in ExecutionModule.java, which compares the actions prescribed by the guideline with the actions actually executed for the current patient. It further includes methods to retrieve parameter values for a current time point (for condition evaluation). Analysis starts with the less time dependent sequential actions (plans) of the guideline. If a sequential action has been executed in the right order and the action conditions are fulfilled, the action is counted as valid. If an action is applied, but other actions should have been applied before, the actions, which should have been applied before, are generated as missing. If an action is executed, but later actions have already been applied before, the executed

action is marked as invalid. If the executed action belongs to the right plan, but the wrong action is executed, it is marked as invalid (and a missing action is generated).In case the actions are applied at the same time point, it is assumed that they have been executed in the right order.

The two parallel ventilation plans are handled in a different way, for each time step it is checked if an executed action has been applied validly or invalidly (if it exists), or if an action is missing. An executed action is marked as valid, if the action conditions are fulfilled and the minimum delay (distance) for the last previously executed action is kept. An action is marked as invalid, if the action condition is not true at the current time point, the minimum delay is violated, or if it occurs after a plan state change to abort or completed. A missing interval starts, if the maximum delay has been reached (distance to the last executed action beyond maximum delay), action conditions are true and no action has been executed. The interval is continued, until the conditions do not hold anymore, an action is applied, or as soon as the abort or complete condition of the plan is fulfilled (state change).

In order to store compliance information, the corresponding data structures are updated. The graph containing the execution data for the patient (`SingleStepGraph`) is enriched with information, action nodes are marked as valid or invalid and additional nodes for missing action intervals are added to the graph. For each executed (valid or invalid) action and for each interval, the interpolated time point is added to the table structure of the parameter views (`prefuse.data.Table`), to be able to render/highlight the action time points and missing intervals in the parameter views. The hierarchical plan structure model, containing the logical structure for compliance analysis (`ComplianceModel.java`), is enriched with counts (missing, valid and invalid) for each individual action.

Data model for compliance analysis

To represent the plan logic needed for compliance checking and to store the valid/invalid/missing actions counts, appropriate data structures have been created. `Action.java` represents an action holding important information like the name, id and the condition (for action execution). It contains methods for the evaluation of conditions and members holding the valid/invalid/missing counts for an action. `PlanBody.java` represents a plan containing a list of actions and subplans belonging to the plan, and information about plan execution (sequential, repeating). Furthermore methods for the evaluation of plan conditions and the retrieval of compliance counts for a plan (accumulation of compliance counts of all subplans and actions of a plan) are defined. Both `Action.java` as well as `PlanBody.java` implement the interface `ComplianceEntity.java`, declaring getter methods for compliance counts, to enable the uniform retrieval of compliance counts for all plan entities (used by the visualizations). `ComplianceModel.java` contains the root plan and holds the counts and ids for irregular actions (not belonging to the guideline); it contains functionality to retrieve a compliance entity by passing their name.

Plan execution view modifications

The data structure for the visual items of the plan execution view is enriched with information about action compliance, actions are marked as valid or invalid and missing intervals

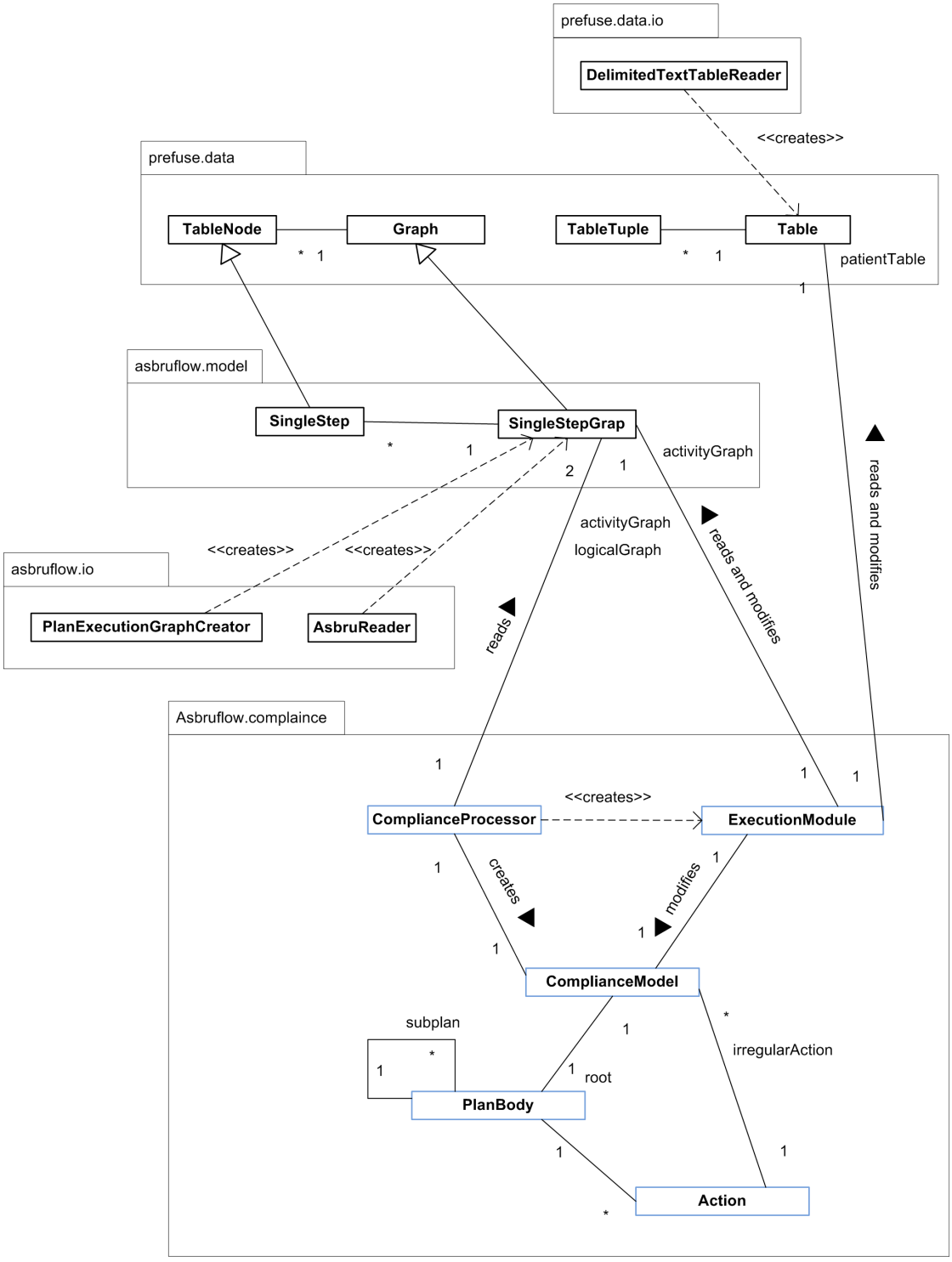


Figure 4.17: Overview of relationships between the different modules, involved in guideline compliance checking (modules added to the prototype are colored in blue).

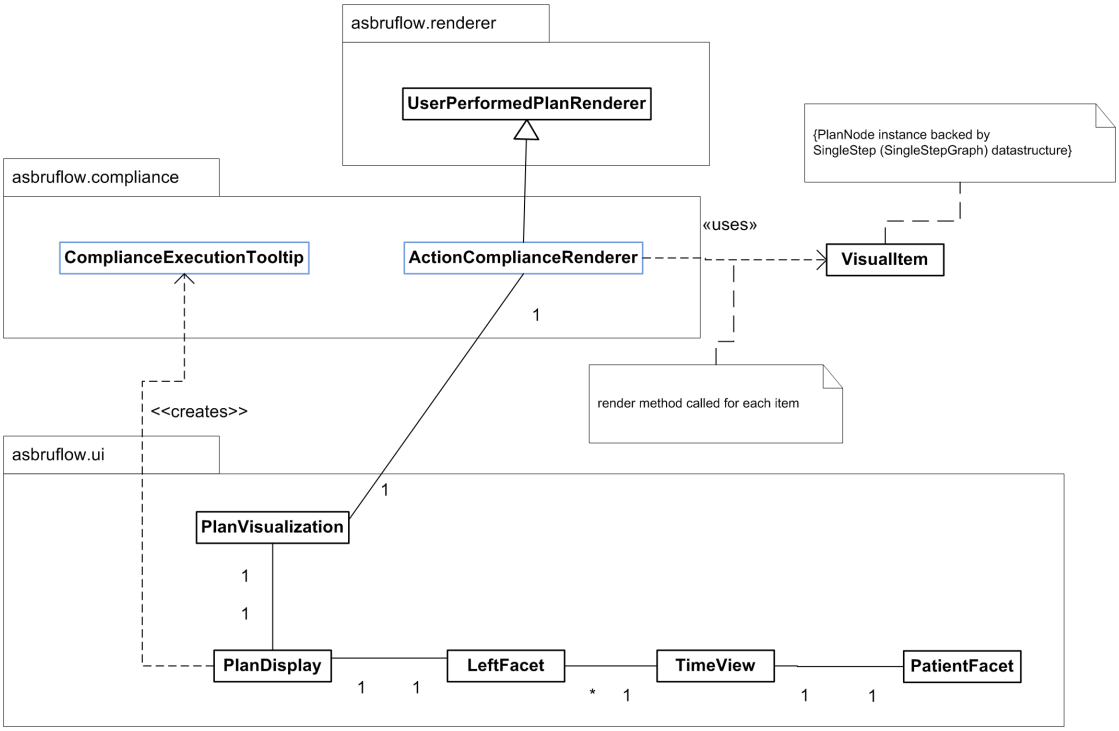


Figure 4.18: The modules involved in the modification of the plan execution view.

are stored as action nodes with a special type and their starting and ending time points. To render plan execution with additional compliance information, the renderer for the executed actions (`UserPerformedPlanRenderer.java`) is extended by `ActionCompliance-Renderer.java` (used by `PlanVisualization.java`), cf. Figure 4.18. This renderer is responsible for drawing the actions depending on their compliance type. Valid actions are drawn as normal diamonds, invalid actions are drawn with an "X" in the center, and missing action intervals are drawn accordingly. It also overwrites the drawing of the action intervals (in case of durative actions). A special (modified) tooltip defined in `ComplianceExecution-Tooltip.java`, shows information like type, name, duration and counts for a visual item. The tooltip is created on mouse hovering in `PlanDisplay.java`, in the same way the old CareCruiser tooltip was created.

Parameter View Extensions

For each action time point and each time point marking the start or end of a missing action interval, a row containing the (interpolated) value and the time point is added to the data table holding the parameter values. `SyntheticProperties.java` is a data model class holding additional information for these rows, for example, if a invalid or valid action occurs at this time point. Time point highlighting in the parameter views for invalid and valid actions

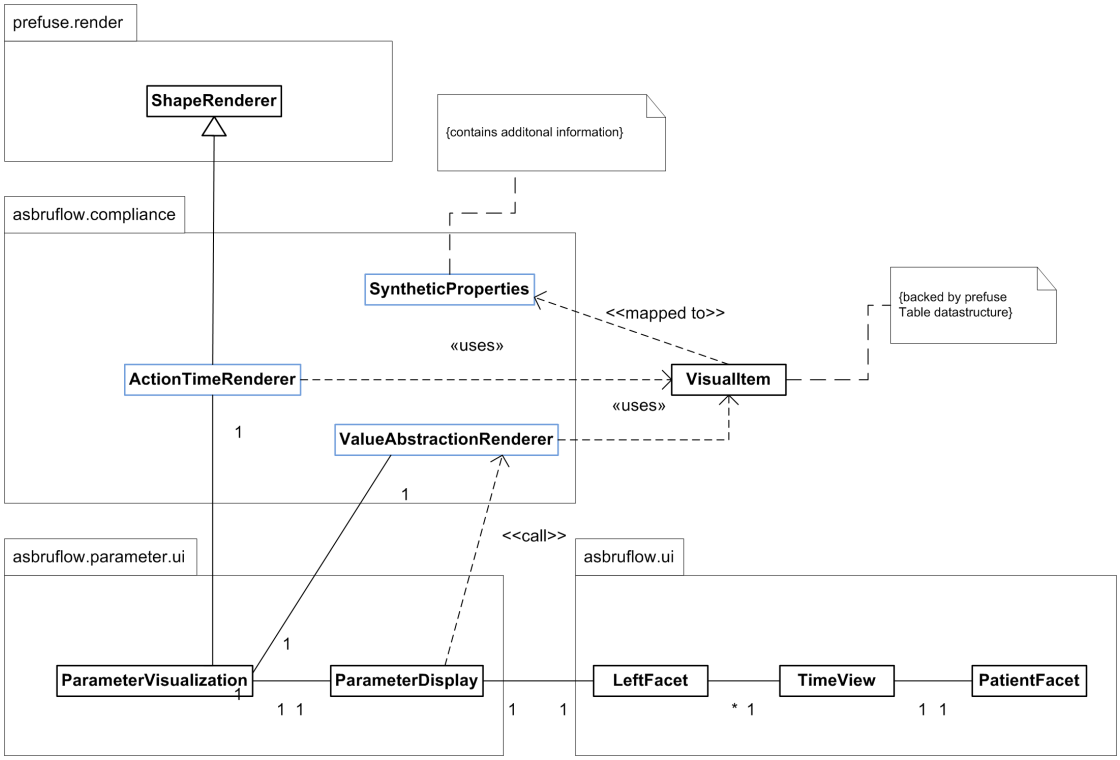


Figure 4.19: Modules involved in visual abstraction and highlighting inside the plan parameter views.

is achieved by `ActionTimeRenderer.java` (instantiated by `ParameterVisualization.java`), which draws the vertical bars and shapes used for highlighting (cf. Figure 4.19). Rendering for data abstraction and highlighting of missing intervals is delegated to `ValueAbstractionRenderer.java` (used by `ParameterVisualization.java`). The abstraction rectangles are drawn based on the specified risk levels of a parameter. This results in three different abstraction classes. If the value is greater than the boundary for high values, it is classified as high value, if it is smaller than the threshold for low values, it is classified as low value, if neither is the case, it is classified as normal value. Missing action intervals are rendered within the abstraction rectangles, based on begin and end time points.

Modification of the logical overview

The tooltip, which is displayed when hovering the mouse over a visual item in the logical overview, has been replaced with a special tooltip (`ComplianceTooltip.java`) showing the compliance counts with a stacked bar (also see next section). The layout is based on swing `GridBagLayout` to achieve proper layout of text and bar inside the tooltip window. Each tooltip shows the compliance counts depending on the hovered plan or action entity (visual

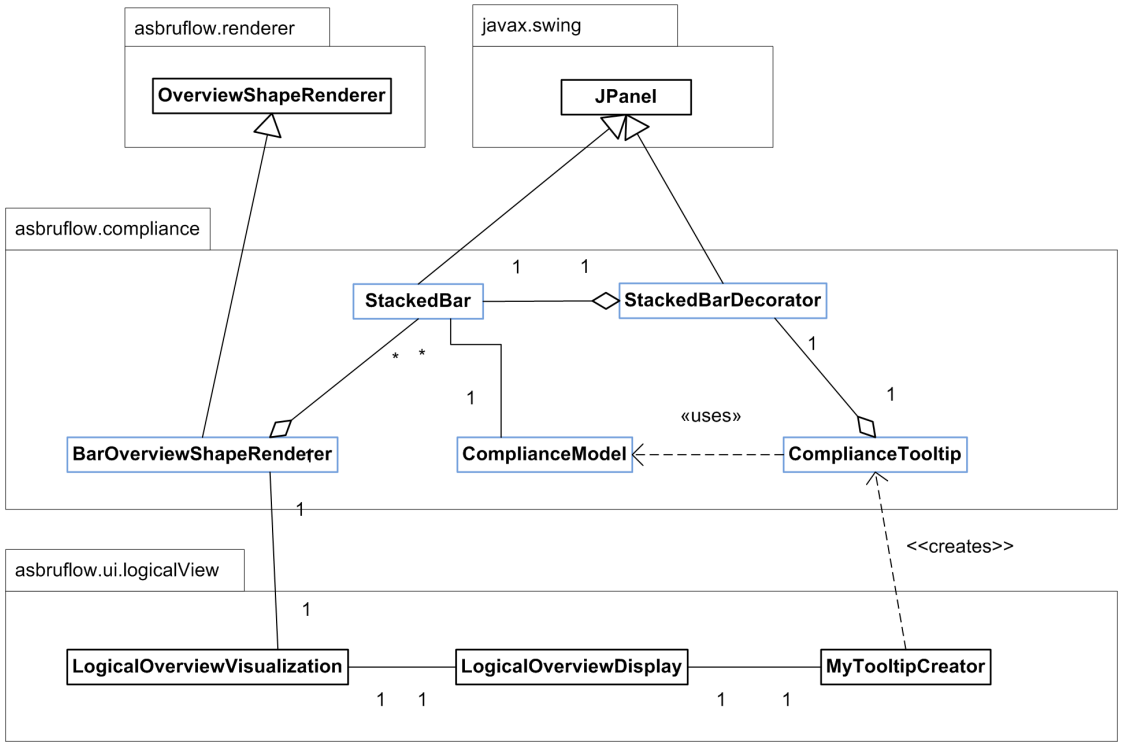


Figure 4.20: Modules involved in the modifications of the overview.

item); the counts are accumulated from all subplans and actions. In order to render miniaturized bars showing the compliance counts for an action, the default renderer is extended by `BarOverviewShapeRenderer` (cf. Figure 4.20), which renders the bars directly beneath the action diamonds in the logical overview.

Additional View

An additional facet containing options, a bar for overall compliance counts and information about patient demographics, is defined in `OptionFacet.java`. (cf. Figure 4.21). The content, GUI elements and layout of the facet (based on a swing `GridBagLayout`) are defined in `OptionContent.java`. This class also implements listeners to react to option menu state changes by notifying the data object defined by `OptionData.java`. The colored stacked bar, visualizing the overall compliance counts for the patients, is implemented in `StackedBar.java`. It supports the drawing of axis labels and allows to choose between linear and sqrt(nth-root) transformation. A decorator class [20], `StackedBarDecorator.java`, enables to plug-in a legend, showing the color coding and counts for the bar (this architecture allows drawing a miniaturized version of the bar, without the legend, in the logical overview and with a legend for a tooltip).

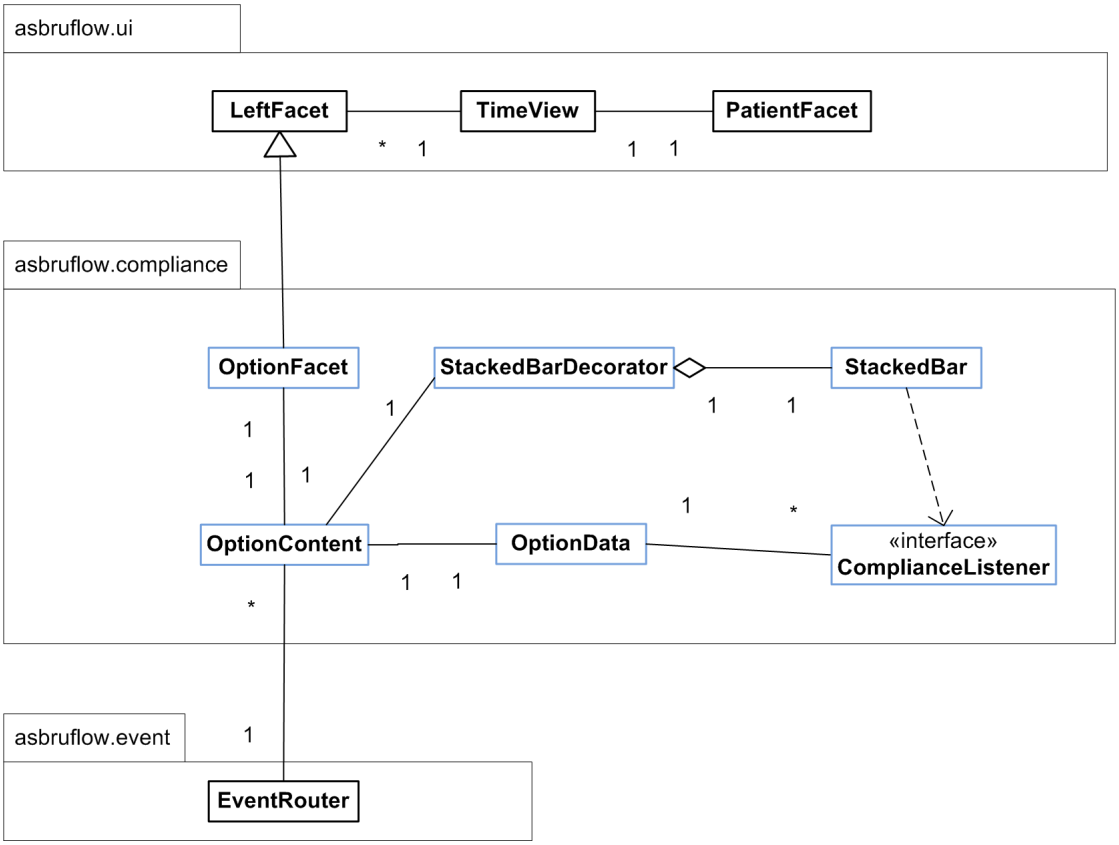


Figure 4.21: Dependencies between modules involved in the definition of the additional view.

CHAPTER 5

Evaluation

The evaluation of the modified prototype consisted of two phases. The first phase was the iterative development of the prototype, where the concepts and iterations were constantly reviewed together with the supervisors, who gave suggestions and ideas for refinement. Problems and options how to proceed were discussed and their feedback was considered during development.

The second phase was the summative evaluation, which had the purpose of giving an overall impression of the quality of the visualization. The background, method and results of the summative evaluation are discussed in the following sections. Section 5 Evaluation Techniques gives an overview about available evaluation techniques. The used evaluation method is outlined in section 5 Method and the results are discussed in section 5 Results.

Evaluation Techniques

The evaluation of information visualization systems is not an easy task and the method to select depends on feasibility, the goals and the type of visualization. Evaluation results are usually hard to generalize, according to [33], results depend on the setting the visualization is used in and it is difficult to find objective measurement for tasks like pattern detection, discovery and hypothesis generation (which are intended to be supported by visualizations). Furthermore the data and tasks used for evaluation have to be the same to compare them and different levels of knowledge (a priori knowledge) of the test users can bias the results.

Evaluation can target different goals, like Effectiveness, Usability and Usefulness, in addition to these factors, issues specific for information visualization systems have to be regarded as well: “Different from a common user interface, a system that uses visual representations must be evaluated not only in terms of the usability and effectiveness of the interface, but also for the information that it manages to communicate to the users through perceptional and cognitive processes.” This involves, for example,“if users manage to decode the graphical codified information” and “if they can recognize interesting patterns” [29, p.128]

A lot of different methods and techniques can be applied for evaluation [29, 33], which have been adapted from fields like HCI (Human Computer Interaction) and Usability Engineering. This involves analytical as well as empirical methods. Analytical methods, like heuristic evaluation and cognitive walkthrough can be performed with reasonable effort and they can be applied in the design phase and for the final evaluation of a system. Empirical methods involve users of a system and can be focused on the collection of qualitative or quantitative data. Methods like controlled experiments are rather focused on quantitative data, which can be used for hypothesis verification, but require a high amount of resources. An example would be two user groups solving specified tasks, one group uses the system to be evaluated, and the other group uses another similar system. Time for task completion is measured and the hypothesis that the efficiency of the new system is higher compared to the old one can be verified by the time measurements. Empirical methods gathering rather qualitative data can be less formal, cheaper and performed with a smaller number of users. These can be user tests, where observation and thinking aloud is in the foreground, or data gathered by interviews, questionnaires and focus groups.

In [32] general aspects of (usability) user tests are described. A test should be carefully planned with goals, procedures and resources etc. considered beforehand. Four stages of a test are identified, preparation, introduction, testing and debriefing. Whereas the introduction has the aim to prepare the test user and to inform him about procedures and test propose, the debriefing phase takes place after testing and is used to gather subjective impressions and comments of the user by questionnaires and interviews. Depending on the test goals, different measurements can be made during a user test, e.g. time or number of errors. Thinking aloud is stated to be a good method for collecting qualitative data from a small number of users, with this method the user is basically asked to express his thoughts during the user test. Interviews can be used for gathering subjective impressions, in comparison to questionnaires they are more flexible and suitable for gathering qualitative data. The type of user testing performed for the summative evaluation in this work is focused on the gathering of qualitative data (also see next section), it therefore relies on observation and applies the thinking aloud method during the test, with an interview scheduled afterwards, the user testing (second evaluation phase) of the original CareCruiser was conducted in a similar way [24]. The authors performed user tests with medical experts and interviews have been conducted afterwards.

Method

The user test conducted for the summative evaluation is rather low scaled and performed with one medical expert. This limitation has several reasons, on the one hand it was difficult to gather expert test users and on the other hand, this work is rather focused on the elaboration of basic concepts that can be found regarding visualization of guideline compliance. The evaluation therefore is concentrated on the gathering of qualitative data, quantitative measurements like time or mouse clicks might be useful, if the amount of users is big enough (significance) and for hypothesis verification, but in this case the user test is rather of explorative nature (and without the possibility to be compared with another visualization type).

The goal of the user test is to gather qualitative data in order to gain insight if the user can work with the visualization (Usability) and understands the visual metaphors represented in the system. The latter is a goal specifically related to information visualization systems, this is

inspired by [45] where user testing is not only focused on usability issues, but also considers comprehension and understanding of visual elements/metaphors. Additional goals are to get user feedback and impressions, to see how an expert uses such a system and if he is able to identify relevant patterns. It also has the goal to give suggestions for small refinements for the future. The scheduled stages are the following:

- Introduction
  - Test purpose
  - Test procedure
  - Explain/show system functionality (of original prototype and modifications)
- User test: Tasks to solve are presented
  The data recorded during the user test includes the following:
  - audio and screen
  - important thoughts and observations during the test
  - problems that occurred
- Debriefing and Interview

The first stage is the introduction, the user is informed about the purpose and procedure of the test, e.g. that several tasks will be given to him, which he is asked to solve. The semantics of the guideline and the interface are also explained, and he is acquainted with the interface and the interactions of the prototype. In the next stage the actual user test is conducted, during this phase the necessary data is recorded. The user is given a set of simple tasks to get his opinion on the system/visualization. Table 5.1 contains two example tasks, a list of all tasks can be found in the appendix (A User Test Task List). In accordance with the thinking aloud method, the user is asked to express his thoughts during the test, important observations are also recorded. The interview is performed afterwards, the user is asked, for example, about his impressions, his opinion about advantages and drawbacks of the visualization, if he sees all necessary information represented, if he would add something or leave something out and if he wants to make any comments or suggestions.

Results

The summative evaluation was conducted with a resident physician in neurosurgery as test user. The overall evaluation together with the expert text user took about 90 minutes. Approximately 30 minutes were spent on the introduction, where the purpose and the test procedure were explained. Furthermore the guideline semantic was quickly covered and the physician was given the opportunity to explore the main interface and interactions of CareCruiser and the modified prototype for a short amount of time (to get an overview of the interaction mechanisms and the interface).

Description	How often has the adjust_tcSO2_high action been applied validly? Achieve this by using the plan execution view and locate the corresponding values in the parameter view.
Preconditions	Execution of ventilation subplans visible and test patient data loaded.
Result	User names the action count and points to the according timepoints .
Description	Find out, if the adjust_tcSO2_low action has been missing at least once, and/or has been applied validly at least once, and/or has been applied invalidly at least once.
Preconditions	Execution of ventilation subplans visible and test patient data loaded.
Result	User explains which type of action instance occurred.

Table 5.1: Examples of the tasks handed to the user.

User Test

After the introductory phase, the physician was asked to accomplish 12 tasks to see how well the visualizations und interactions are working, when operated by a real user and to ensure that the test user is sufficiently acquainted with the prototype for the subsequent interview. It turned out that the problems found in the prototype are related to usability issues and small refinements, that would need to be considered for an application intended for end-users. The overall understanding of the visual mappings and encodings was very good and the use of the visualizations turned out to be intuitive most of the time. Important observations and issues found are discussed in the following.

When confronted with the first task, which required the use of the overview visualization (which shows action compliance counts) for identifying actions with a missing instance, he quickly identified the actions having a miniature bar beneath them, which had the according color. He got stuck when trying to scroll the overview in the correct position (this is achieved by mouse dragging). This might have happened, because he was still in the learning phase, maybe explicit scrollbars or interface elements for panning would help inexperienced users in this case, but they would also cover up valuable screen space. Another task required to check the highlighting of valid actions in the settings, in order to see them in the overview visualization. He got confused, because highlighting of valid actions was not activated on default and he overlooked the settings. This suggests that the highlighting of missing, invalid, as well as valid actions or no highlighting at all should be active on default.

Identifying valid actions in the temporal plan execution view and relating them with parameter values was quickly achieved by the test user, due to highlighting in the parameter view. When he worked on identifying invalid guideline actions, he was not sure, which invalid actions belonged to the guideline and which were not in the guideline. This suggests that a different visual encoding of instances of invalid guideline and invalid non-guideline actions can be considered

or, as suggested by the physician in the interview (also see following section), an emphasized visual separation of the executed guideline plans and non-guideline actions. However it can be assumed that this problem also occurred due to a lack of user experience with the prototype.

When asked to estimate the proportion of the time during which actions were missing based on parameter view highlighting and to seek the action of the two ventilation plans, which is missing most frequently, he managed to accomplish the tasks quickly. This suggests that the highlighting and encoding of missing intervals in parameter and plan execution view seems to be understood intuitively. The same was true for the overview bar in the statistics, which was used by the physician to identify the overall missing and valid counts.

Some problems occurred, when the physician tried to solve tasks, which were intended to be solved by using the overview. Being asked to provide the counts for missing and valid actions for all initial plans, he used the plan execution view at first and not the overview (where this can be easily achieved by hovering the mouse over the initial plan). It seemed that he was not aware of the plan hierarchy in the overview and had difficulties identifying the correct plan. This might have occurred due to a lack of experience with the program, but might be avoided by making the meaning of the plans more obvious, e.g. labeling (also see interview section). After being able to locate the correct element and being aware of the tooltip functionality for plans in the overview, he quickly provided the correct counts. When being asked to find out, if an action had one valid/invalid or missing instance, it seemed that working with the time oriented views was more intuitive at first glance than using the overview.

The time point of the abort of the ventilation plan was identified very quickly by the physician (during the last task), which suggests that the visual encoding of plan abort (and completion, which is done the same way) in the execution view was intuitively understood by the user.

Interview

The interview was conducted after the test; the physician was given the opportunity to explore the prototype freely before and during the interview. The physician stated that he had a very good overall impression of the prototype, that it is able to give a good overview and can make the analysis of compliance with medical treatment plans easier.

Asked about his understanding of the visual elements, he explained that he thought he had a very good overall understanding, but that he had problems with understanding the plan hierarchy (i.e. to understand the meaning of the plan elements in the overview), which was necessary to solve the task of finding out the compliance counts for initial plans. He suggested that more details, e.g. textual descriptions containing additional information (name, cues etc.) inside the boxes representing a plan, would help to derive the meaning, especially for beginners. He also pointed out that he would like a legend in the overview, explaining the color encoding for the miniature bars.

The next questions were about his impression and his opinion about the advantages and drawbacks of the visualization. He pointed out that the parameter view modifications are generally suitable to see at which time an action should be taken. He suggested (for live systems) that hints and popups with treatment suggestions would be good for live analysis, e.g. a hint box with “increase parameter” message. Furthermore he stated that it is great to get an overview of treatment compliance, e.g. that is good to see where actions should have been taken and to have

the highlighting of actions directly in the parameter view, because it allows to instantly relate an executed action with the corresponding parameter changes. He would also find this especially useful for educational proposes.

Considering the drawbacks, he pointed out that there might be a bit too many colors in the interface (action colors and compliance colors together), though he thinks that one is able to get used to that with increasing experience. An option for future development might be therefore, at least for plans, which contain exclusive actions (only one action can be applied at a time), to use the same action color scheme in all plans (e.g. negative action color for actions related to low parameter values, and positive action color for actions related to high parameter values). He also explained he would like labels in the plan execution view, for example the display of the action name inside a missing interval, but thinks that this might get less important with increasing user experience.

Furthermore he pointed out that he liked the statistics, especially for offline compliance analysis, but that he would like to hide the panel most of the time (note: this is already possible in the interface) to have more screen space for the other views.

Asked about the remaining problems/critical incidents, he agreed that the highlighting of all actions (valid/invalid/missing) should be active on default and/or that it should be easy to notice, if highlighting is active for a certain action type. He also stated that he would like a better visual separation of guideline actions and plans versus non-guideline actions in the plan execution view.

Finally the physician affirmed that all the information is present in the interface, but one needs to get accustomed to all elements. He stated that the visualization of compliance in the overview is generally useful, especially for offline analysis. He pointed out that he would prefer to keep the highlighting of missing intervals, valid and invalid actions activated for offline analysis, for live analysis he thinks he would prefer to deactivate the highlighting, but would keep the value abstraction activated.

Evaluation Summary

It turned out that the overall understanding of the interface and visualizations was intuitive for the test user; minor problems found during evaluation might serve as input for further improvements in the future. The overview visualization modifications were generally appreciated, the miniature bars beneath the action seemed to help to quickly identify if an action e.g. is missing. The hierarchy and accumulation of compliance counts in the overview seemed to be more complex to grasp at first glance, in comparison to the other elements, this might be avoided in future versions by making the meaning of the hierarchical elements more obvious (e.g. by labeling) . The user's subjective impression of the parameter and plan execution view modifications was very good, he remarked that he would like labels in the plan execution view (e.g. inside a missing interval) and for live systems he suggested hint boxes/messages, containing cues, for example informing the user that a certain parameter has to be increased. Both interview and user test showed that parameter view highlighting and statistics (bar showing the compliance counts for the whole treatment) seemed to be especially intuitive and useful.

CHAPTER 6

Discussion and Future Work

6.1 Discussion

Considering the evaluation, extensions and encodings for guideline compliance integrated to the CareCruiser prototype turned out to be useful and intuitive (research questions Rq3 and Rq4 of section 1.3) . Despite some minor issues, which may also give input for further improvements, it seems that the introduced visual artifacts and mappings are able to ease compliance analysis and make relevant data accessible. It seems that the tool is able to give support to a medical expert in the analysis of compliance. Further improvements might include labeling, adaption of color schemes, and making the compliance information, which is presented in the overview, more accessible for novice users. The conducted interview helped to gather additional suggestions for features, like the integration of hint messages for live systems (online execution). Despite the promising results of the evaluation, additional user testing is necessary to make the results representative and to gather more information.

Finding appropriate visual mappings for compliance with medical guidelines turned out to be a demanding task (research questions Rq1,Rq2,Rq6). Huge amounts of data and knowledge about the guideline are required to understand and analyze CIG compliance, and different types of visualizations need to be present in one interface (like in CareCruiser). Information about compliance has to be integrated in a consistent manner, without overwhelming the user with too much information and with the aim to help medical experts to identify important information quickly. These issues have been tried to be solved by different means, including the following ideas:

Transparent overlay and highlighting in the temporal views: The abstraction of parameters into the categories normal, low and high is visualized as overlay to the raw data visualizations. This technique is meant to increase the speed of critical value identification, while preserving the information of the raw data view. The same is true for highlighting, which emphasizes invalid and valid action time points, and intervals during which the execution of an action is

missing. Highlighting can be activated or deactivated separately and aims to focus the attention on important time points i.e. time spans.

Missing action intervals: Actions which have to be executed periodically with a short time delay would cause the generation of a lot of missing actions; aggregating them into a missing interval avoids visual clutter. In addition, the display of individual missing actions would only represent one possible valid execution, because the exact time point of action application (between minimum and maximum delay) depends on user choice. Using intervals therefore also helps to cope with this on conceptual level.

Different types of actions: The presented classification scheme for valid, invalid and missing actions allows providing overall counts. It enables to omit details for a certain instance and to concentrate on guideline compliance from a higher point of view.

Higher level aggregation in the structural plan overview: Aggregated counts for plans, subplans and irregular actions are displayed in the plan overview. This modification aims to give a quick overview for analysis tasks, which do not require the inspection of temporal relations between actions or parameters.

Future work might include searching for additional means to reduce the amount of information displayed, while keeping the important information and patterns visible. Alternative high level views might be useful depending on the context e.g., only displaying missing action intervals and valid/invalid action time points, without representing parameter data or the action-ID visually, might suffice for some analysis tasks, although there might always be a trade-off between the amount of data, which can be displayed, and the level of abstraction used.

Future work might also focus on comparing multiple patients, for example by enabling the search for similar patterns occurring during treatment sessions for different patients. Systems like EventFlow [31] (see section 3.1) allow specifying patterns for time points and intervals and visualize the aggregated results. This can be considered to be applied in a similar way to valid/invalid action time points and missing action intervals, for example, one might search for the application of a valid action followed by an interval, during which the action is missing.

Another way of finding correlations between multiple patients might be to use rearrangeable views with colored bands like in EventViewer [8] (i.e. with highlighting only). For example, a band could represent the time point of an invalid action application and arrange them into stacks or panels for patients with a common attribute, or treatments with specific clinical context may allow identification of interesting patterns.

Another important aspect to deal with is the exact definition of compliance (research question Rq5), which can be checked on several levels. Considering this work the questions was how to define compliance in order to give suitable means for an expert to asses treatment and guideline quality. The definition used in this work, which was compliance checking on the level of applied actions compared to the actions recommended by the guideline, was chosen because it gives a good impression of direct adherence to a guideline, does not require additional knowledge,

and aims to be easy to understand for physicians. Other definitions e.g., checking if effects of executed actions are compliant with the guideline, might require adaption of visual mappings (there might be, for example, an additional type of action compliance, i.e. an action, which does not exactly match a guideline action, but is still compliant due to its effects). Suitable visualization techniques and mappings therefore depend on the used definition of compliance, in addition to the task to be solved and the context of the guideline.

A procedural approach, tailored for checking action compliance with a specific guideline, was chosen in this work, with the goal to enable interactive testing of concepts and visualizations. Future work might focus on generalizing mechanisms to enable the checking and visualization of other guidelines. A possible way to achieve this is to use formal approaches (e.g. model checking) like it has been presented in [21] for guidelines formulated in Asbru. This approach has the advantage that checking is done on a formal basis and software is available for this purpose. Drawbacks might be that the guideline has to be mapped manually to the chosen language and that the expressive power of the formalism has to be chosen according to the guideline semantics and the definition of compliance. Another possibility might be the use of a procedural approach by applying the Asbru interpreter[1]. A lot of functions needed to achieve this is already present and the interpreter could be extended with means for guideline adherence checking. The advantages of applying this method might be the easier integration and adaption.

6.2 Summary and Conclusion

CIGs (computer interpretable guidelines) aim to increase the quality of health care, visualization of compliance with a CIG can facilitate their implementation, may help to identify reasons for deviations from a guideline and therefore also assist in the design and improvement of guidelines. The goal of this work was to search for concepts for the visualization of compliance information and the integration of these into views of the CareCruiser [22] prototype. This required defining the particular type of compliance with a guideline and finding proper visual mappings and encodings to represent this information in the interface.

Compliance was chosen to be defined on action level, which means that the actions prescribed by the guideline are compared to the executed actions. Three types of actions considering compliance have been presented, which are valid actions, invalid actions and missing actions (i.e. intervals of missing action application). The checking method was implemented based on a procedural approach, tailored to check adherence to a ventilation guideline (formulated in Asbru), with the aim to explore and test the concepts found.

Various visual encodings have been found to represent guideline compliance information in the interface. Views showing the parameters have been extended by a transparent overlay, which represents the abstraction of numerical values into the category high, normal or low. Action time points and intervals of missing action application can be highlighted. Abstraction and highlighting has the goal to ease the identification of important patterns and to help to analyze the relation between actions and their corresponding parameters. The visualization of plan exe-

[1] http://ieg.ifs.tuwien.ac.at/projects/interpreter/index.html

cution, which shows the actions executed during treatment, was enriched with information about action compliance. Invalid actions are marked with an “X”, metaphorically stating that they are forbidden by the guideline, whereas missing intervals are mapped to a horizontal bar (composed of an upper and a lower part), with the length equal to the duration of the missing action time span. An additional view was implemented, which allows to see patient demographics and the overall action counts, and provides settings to enable or disable features like highlighting. The counts are mapped to a stacked bar, making it easy to estimate the proportions between valid, invalid and missing action counts. The structural plan overview has been modified to show the aggregated counts for each subplan and action type. They can be inspected via tooltips and are also rendered directly beneath the action nodes, this has the purpose to enable a quick overview and immediate identification of action compliance. All functionality has been implemented in Java and is built upon the CareCruiser prototype; the prototype as well as the extension makes use of the Swing-API and the Prefuse framework.

An evaluation was performed with an expert in the medical domain (physician), with the goal to gather qualitative data. The visual mappings and encodings were generally appreciated; the user test and the interview also provided useful suggestions for feature improvements and extensions. Future work might focus on the visualization of different guidelines, different definitions of compliance or on finding additional means for visualization of guideline adherence on various levels of abstraction.

APPENDIX A

User Test Task List

Task 1

Description	Try to find all actions, which have a missing instance. Achieve this by looking solely on the structural plan overview.
Preconditions	Overview visible and test patient data loaded.
Result	User points to the according actions or names them.

Task 2

Description	Try to find all actions, which have a valid instance. Achieve this by looking solely on the structural plan overview.
Preconditions	Overview visible and test patient data loaded.
Result	User points to the according actions or names them.

Task 3

Description	Try to find all actions, which have an invalid instance. Achieve this by looking solely on the structural plan overview.
Preconditions	Overview visible and test patient data loaded.
Result	User points to the according actions or names them.

Task 4

Description	Deactivate all highlighting in the parameter views and, after achieving this, activate all highlighting.
Preconditions	Test patient data loaded.
Result	User undertakes the according actions.

Task 5

Description	How often has the adjust_tcSO2_high action been applied validly? Achieve this by using the plan execution view and locate the corresponding values in the parameter view.
Preconditions	Execution of ventilation subplans visible and test patient data loaded.
Result	User names the action count and points to the according timepoints .

Task 6

Description	How many invalid (guideline) actions have been applied? If there are invalid actions identify the corresponding parameter values.
Preconditions	Execution of ventilation subplans visible and test patient data loaded.
Result	User names the action count and points to the according parameter values.

Task 7

Description	Try to estimate the amount of missing action time for each of the ventilation subplans.
Preconditions	Execution of ventilation subplans visible and test patient data loaded.
Result	User states his estimation.

Task 8

Description	Try to identify the action of the two ventilation plans, which occurs to be the action that is missing most frequently.
Preconditions	Execution of ventilation subplans visible and test patient data loaded.
Result	User points or names the action.

Task 9

Description	Provide the overall counts for invalid and missing actions.
Preconditions	Test patient data loaded and pane with statistics is visible.
Result	User names the action counts.

Task 10

Description	Provide the count of missing and valid actions, belonging to the initial plan.
Preconditions	Overview visible and test patient data loaded.
Result	User names the action counts.

Task 11

Description	Find out, if the adjust_tcSO2_low action has been missing at least once, and/or has been applied validly at least once, and/or has been applied invalidly at least once.
Preconditions	Execution of ventilation subplans visible, overview visible and test patient data loaded.
Result	User explains which type of action instance occurred.

Task 12

Description	Do the plan abortion or completion conditions get true, und if they get true, at which timepoint?
Preconditions	Execution of ventilation subplans visible and test patient data loaded with conditions set to active.
Result	User names or points to the timepoint.

Bibliography

[1] A. Advani, K. Lo, and Y. Shahar. Intention-based critiquing of guideline-oriented medical care: The asgaard project at stanford. In *Proc. AMIA Annual Symposium*, pages 483–487, 1998.

[2] W. Aigner, P. Federico, T. Gschwandtner, S. Miksch, and A. Rind. Challenges of time-oriented data in visual analytics for healthcare. In *IEEE VisWeek Workshop on Visual Analytics in Healthcare*. IEEE, 2012.

[3] W. Aigner, K. Kaiser, and S. Miksch. *Visualization Techniques to Support Authoring, Execution, and Maintenance of Clinical Guidelines*, page 140–159. IOS Press, Health Technology and Informatics, 2008.

[4] W. Aigner and S. Miksch. Carevis: integrated visualization of computerized protocols and temporal patient data. *Artificial intelligence in medicine*, 37(3):203–18, 2006.

[5] W. Aigner, S. Miksch, H.Schumann, and C. Tominski. *Visualization of Time-Oriented Data*. Human-Computer Interaction. Springer Verlag, 1st edition, 2011.

[6] W. Aigner, A. Rind, and S. Hoffmann. Comparative evaluation of an interactive time-series visualization that combines quantitative data with qualitative abstractions. In *Computer Graphics Forum*, volume 31, pages 995–1004. Wiley Online Library, 2012.

[7] R. Bader, S. Schlechtweg, and S. Miksch. Connecting time-oriented data and information to a coherent interactive visualization. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04, pages 105–112, New York, NY, USA, 2004. ACM.

[8] K. Beard, H.E. Deese, J. Emerson, and N.R. Pettigrew. The eventviewer: A tool for visualizing and exploring events extracted from ocean observing system data. In *OCEANS 2010 IEEE - Sydney*, pages 1–8, 2010.

[9] P. Bodesinsky, P. Federico, and S. Miksch. Visual analysis of compliance with clinical guidelines. In *Proceedings of the 13th International Conference on Knowledge Management and Knowledge Technologies*. ACM, 2013.

[10] S. Card. The human-computer interaction handbook: Fundamentals, evolving technologies, and emerging applications. chapter Information visualization, pages 509–543. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 2008.

[11] S. Card, J. Mackinlay, and B. Shneiderman, editors. *Readings in information visualization: using vision to think*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999.

[12] F. Chesani, E. Lamma, P. Mello, M. Montali, S. Storari, P. Baldazzi, and M. Manfredi. Compliance checking of cancer-screening careflows: an approach based on computational logic. *Studies in health technology and informatics*, 139:183–92, 2008.

[13] E. Chi. A taxonomy of visualization techniques using the data state reference model. In *Proceedings of the IEEE Symposium on Information Vizualization 2000*, INFOVIS '00, pages 69–, Washington, DC, USA, 2000. IEEE Computer Society.

[14] L. Chittaro. Visualization of patient data at different temporal granularities on mobile devices. In *Proceedings of the working conference on Advanced visual interfaces*, AVI '06, pages 484–487, New York, NY, USA, 2006. ACM.

[15] T. Christian. Event-based concepts for user-driven visualization. *Information Visualization*, pages 65–81, 2011.

[16] P. De Clercq, K. Kaiser, and A. Hasman. *Computer-Interpretable Guideline formalisms.*, volume 139 of *Studies in Health Technology and Informatics*, pages 22–43. IOS Press, 2008.

[17] A. Faiola and S. Hillier. Multivariate relational visualization of complex clinical datasets in a critical care setting: A data visualization interactive prototype. In *Tenth International Conference on Information Visualization, 2006. IV 2006.*, pages 460–468, 2006.

[18] A. Faiola and C. Newlon. Advancing critical care in the icu: a human-centered biomedical data visualization systems. In *Proceedings of the 2011th international conference on Ergonomics and health aspects of work with computers*, EHAWC'11, pages 119–128, Berlin, Heidelberg, 2011. Springer-Verlag.

[19] C. Fuchsberger. Entwicklung einer ausfuehreinheit fuer asbru light'. Master's thesis, Vienna University of Technology, 2003.

[20] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design patterns: elements of reusable object-oriented software*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1995.

[21] P. Groot, A. Hommersom, P. J. F. Lucas, R.-J. Merk, A. ten Teije, F. van Harmelen, and R. Serban. Using model checking for critiquing based on clinical guidelines. *Artif. Intell. Med.*, 46(1):19–36, May 2009.

[22] T. Gschwandtner, W. Aigner, K. Kaiser, S. Miksch, and A. Seyfang. Carecruiser: Exploring and visualizing plans, events, and effects interactively. In *Pacific Visualization Symposium (PacificVis), 2011 IEEE*, pages 43–50, 2011.

[23] T. Gschwandtner, W. Aigner, K. Kaiser, S. Miksch, and A. Seyfang. Design and evaluation of an interactive visualization of therapy plans and patient data. In *Proceedings of the 25th BCS Conference on Human-Computer Interaction*, BCS-HCI '11, pages 421–428, Swinton, UK, UK, 2011. British Computer Society.

[24] T. Gschwandtner, W. Aigner, K. Kaiser, S. Miksch, and A. Seyfang. Design and evaluation of an interactive visualization of therapy plans and patient data. In *Proc. of the BCS HCI Conference*, 2011.

[25] The Asgaard Project Homepage. http://www.asgaard.tuwien.ac.at/. Accessed: 2012-12-03.

[26] D. Keim, F. Mansmann, J. Schneidewind, J. Thomas, and H. Ziegler. Visual data mining. chapter Visual Analytics: Scope and Challenges, pages 76–90. Springer-Verlag, Berlin, Heidelberg, 2008.

[27] R. Kosara, S. Miksch, Y. Shahar, and P. D. Johnson. Asbruview: Capturing complex, time-oriented plans - beyond flow-charts. In *The Second Workshop on Thinking with Diagrams 1998 (TwD-98)*, pages 119–126. The University of Wales, The University of Wales, 1998.

[28] Prefuse Manual. http://prefuse.org/doc/manual/. Accessed:2012-08-31.

[29] R. Mazza. *Introduction to Information Visualization*. Springer Publishing Company, Incorporated, 1 edition, 2009.

[30] S. Miksch, W. Horn, C. Popow, and F. Paky. Context-sensitive and expectation-guided temporal abstraction of high-frequency data. In *Proceedings of the Tenth International Workshop for Qualitative Reasoning (QR-96), Stanford Sierra Camp. Fallen Leaf Lake, CA*, pages 154–63. AAAI, 1996.

[31] M. Monroe, K. Wongsuphasawat, C. Plaisant, B. Shneiderman, J. Millstein, and S. Gold. Exploring point and interval event patterns: Display methods and interactive visual query, 2012.

[32] J. Nielsen. *Usability Engineering*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[33] C. Plaisant. The challenge of information visualization evaluation. In *Proceedings of the working conference on Advanced visual interfaces*, AVI '04, pages 109–116, New York, NY, USA, 2004. ACM.

[34] S. Quaglini. *Compliance with clinical practice guidelines.*, volume 139 of *Studies in Health Technology and Informatics*, pages 160–79. IOS Press, 2008.

[35] A. Rind, W. Aigner, S. Miksch, S. Wiltner, M. Pohl, T. Turic, and F. Drexler. Visual exploration of time-oriented patient data for chronic diseases: Design study and evaluation. In *Proceedings of USAB 2011: Information Quality in e-Health*, pages 301–320. Springer, Springer, 2011.

[36] A. Rind, T. D. Wang, W. Aigner, S. Miksch, K. Wonsuphasawat, C. Plaisant, and B. Shneiderman. Interactive information visualization for exploring and querying electronic health records: A systematic review. Technical report, Tech Report HCIL-2010-19, 2010.

[37] K. Rosenbrand, J. Van Croonenborg, and J. Wittenberg. *Guideline development.*, volume 139 of *Studies in Health Technology and Informatics*, pages 3–21. IOS Press, 2008.

[38] A. Seyfang, K. Kaiser, T. Gschwandtner, and S. Miksch. Visualizing complex process hierarchies during the modeling process. In *Business Process Management Workshops*, pages 768–779. Springer, 2013.

[39] A. Seyfang, R. Kosara, and S. Miksch. Asbru's reference manual, asbru version 7.3. Technical report, 2002.

[40] Y. Shahar, D. Goren-Bar, D. Boaz, and G. Tahanl. Distributed, intelligent, interactive visualization and exploration of time-oriented clinical data and their abstractions. *Artif. Intell. Med.*, 38(2):115–135, October 2006.

[41] Y. Shahar, S. Miksch, and P. Johnson. The asgaard project: A task-specific framework for the application and critiquing of time-oriented clinical guidelines. In *Artificial intelligence in medicine*, pages 29–51, 1998.

[42] SmartMoney. http://infosthetics.com/archives/2006/03/stock_market_radarsmartmoney_-data_visualization.html/. Accessed:2013-02-27.

[43] S.M.Powsner and E. R. Tufte. Graphical summary of patient status. *Lancet*, 344(3):386–9, 1994.

[44] M. Stacey and C. McGregor. Temporal abstraction in intelligent clinical data analysis: A survey. *Artif. Intell. Med.*, 39(1):1–24, January 2007.

[45] A. G. Sutcliffe, M. Ennis, and J. Hu. Evaluating the effectiveness of visual user interfaces for information retrieval. *Int. J. Hum.-Comput. Stud.*, 53(5):741–763, November 2000.

[46] P. Terenziani, S. Montani, A. Bottrighi, M. Torchio, G. Molino, and G. Correndo. The glare approach to clinical guidelines: main features. *Studies in health technology and informatics*, 101:162–6, 2004.

[47] J. Thomas and K. Cook. *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. National Visualization and Analytics Ctr, 2005.

[48] Oracle Swing Tutorial. http://docs.oracle.com/javase/tutorial/uiswing/. Accessed:2012-06-22.