Diplomarbeit: Visualisierung konzeptioneller Beschreibungen von Programmieraktivitäten

---

Arbeitsgruppe:

Software-Engineering

Nicolas Ngandeu

Gliederung

- Einführung
- Visualisierung
- Die Akteure
  - Die Inputdaten
  - Framework
- Implementierung
- Probleme
- Fazit
- Ausführung der Software

Einführung

- Im Rahmen unterschiedlicher Forschungen werden die Programmiertätigkeiten (bzw. Sitzungen) von Entwicklern über ein bestimmtes Zeitintervall beobachtet.

➔ **Ziel der Forschungen ist die Untersuchung der Arbeitsweise von Programmierern unter verschiedenen Gesichtspunkten:**

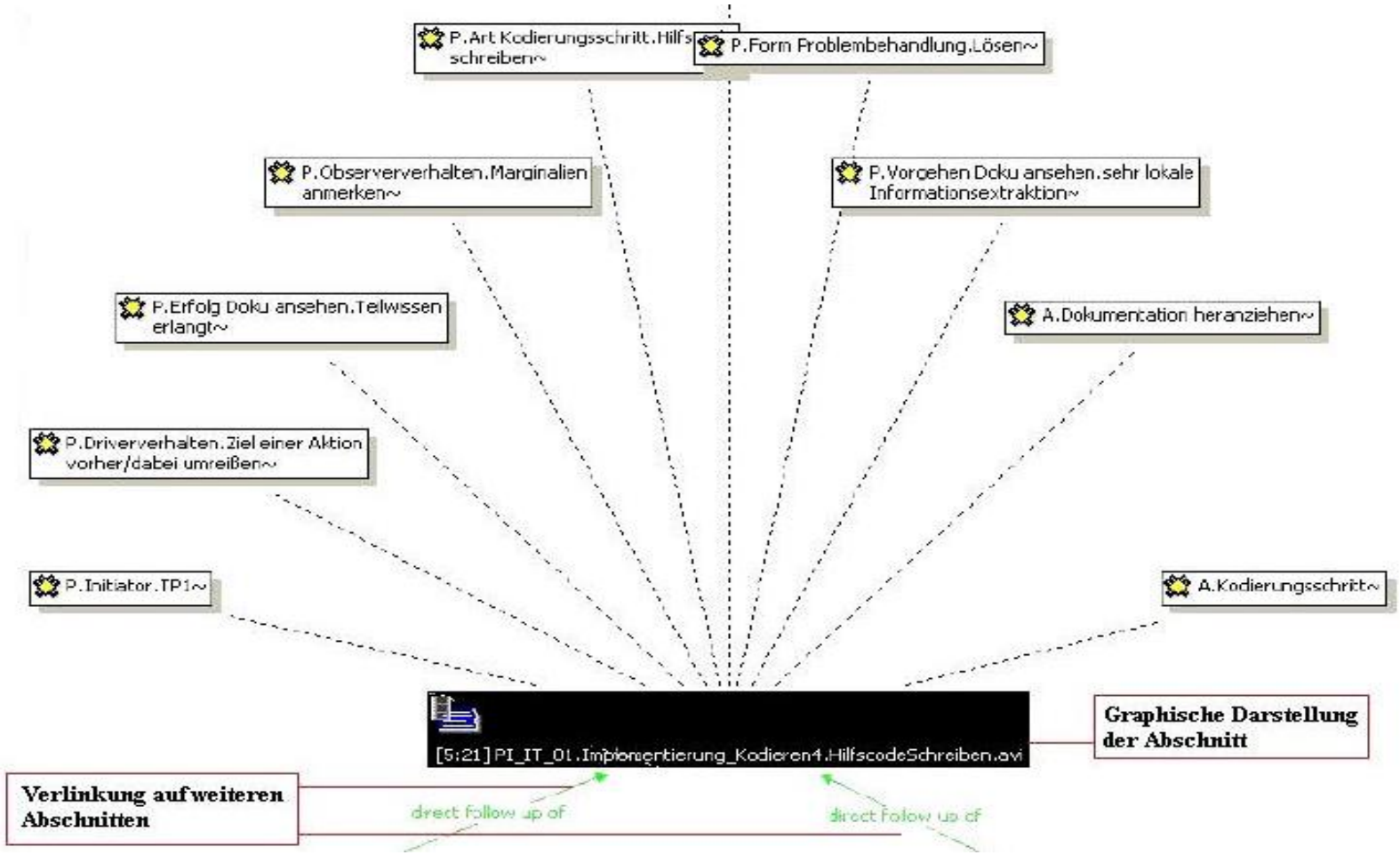
  - Paarprogrammierung: zwei Programmierer arbeiten zusammen an einem Rechner (wie z.B. Produktivität)
  - Entstehung von Programmierfehlern (Fehler entdecken durch bestimmte Aktivitätsfolge)

- Aus diesen Untersuchungen möchte man typische Aktivitäts- und Verhaltensmuster entdecken können.

Einführung

- Die Untersuchungen erfordern eine Aufzeichnung der Arbeitsschritte der Entwickler:
  - Videosequenzen (Atlas.ti)
  - Automatisierte Aufzeichnung von bestimmten Abfolgen einzelner individueller Aktivitäten der Entwickler bei ihrer Arbeit (Mikroprozesse der Softwareentwicklung) (ECG)

Einführung


P.Art Kodierungsschritt.Hilfs schreiben~
P.Form Problembehandlung.Lösen~
P.Obserververhalten.Marginalien annerken~
P.Vorgehen Doku ansehen.sehr lokale Informationsextraktion~
P.Erfolg Doku ansehen.Teilwissen erlangt~
A.Dokumentation heranziehen~
P.Driververhalten.Ziel einer Aktion vorher/dabei umreißen~
P.Initiator.TP1~
A.Kodierungsschritt~
[5:21] PI_IT_01.Implementierung_Kodieren4.HilfscodeSchreiben.avi
Graphische Darstellung der Abschnitt
Verlinkung aufweiteren Abschnitten
direct follow up of
direct follow up of

Einführung

type	timestamp	endtime	resourcename	duration	changed
msdtpartactive.xsd	16:54:53	16:55:12	Welcome	19000	
msdtpartactive.xsd	16:55:12	16:55:15	Package Explorer	3000	
msdtwindowactive.xsd		16:54:53	16:55:15	Java - Eclipse SDK	22000
msdtpartactive.xsd	16:55:37	16:56:54	Package Explorer	77000	
msdtwindowactive.xsd		16:55:37	16:56:54	Java - Eclipse SDK	77000
msdtpartactive.xsd	16:57:02	16:57:29	Package Explorer	27000	
msdtwindowactive.xsd		16:57:02	16:57:29	Java - Eclipse SDK	27000
msdtpartactive.xsd	16:57:43	16:57:45	Package Explorer	2000	
msdtwindowactive.xsd		16:57:43	16:57:45	Java - Eclipse SDK	2000
msdtpartactive.xsd	16:57:45	16:57:45	Package Explorer	0	
msdtwindowactive.xsd		16:57:45	16:57:45	Java - Eclipse SDK	0
msdtpartactive.xsd	16:57:56	16:57:57	Package Explorer	1000	
msdtwindowactive.xsd		16:57:56	16:57:57	Java - Eclipse SDK	1000

Einführung

- Diese Untersuchungen stoßen auf gewisse Probleme:
  - die jetzigen Sichtmöglichkeiten geben keine Informationen über die zeitliche (und inhaltliche) Verteilung der Tätigkeiten.
  - keine Visualisierung von auf bestimmten Interessen hin gefilterten Informationen über die Zeit hinweg möglich

- **➔ Keine Muster über die Zeit hinweg erkennbar**

Einführung

- Auf Grund dieser Probleme ist ein Visualisierungstool entwickelt worden:
  - Zeitabhängige Visualisierung der Programmiertätigkeiten durch grafische Elemente,
  - Eine quantitative Darstellung bestimmter Datensätze (die Werte der Datensätze durch visuelle Variablen darstellen),
  - Benutzerinteraktion ermöglichen, um benutzerdefinierte Sichten auf den Inputdaten zu ermöglichen.
  - Die unterschiedlichen Datenerhebungsmethoden generieren genauso viele unterschiedliche Inputdatenformate ➔ Generizität der SW im Hinblick auf die Inputdaten.

➔ **Generic T**ime-dependent data **V**isualization (**Generic TV**).

Gliederung

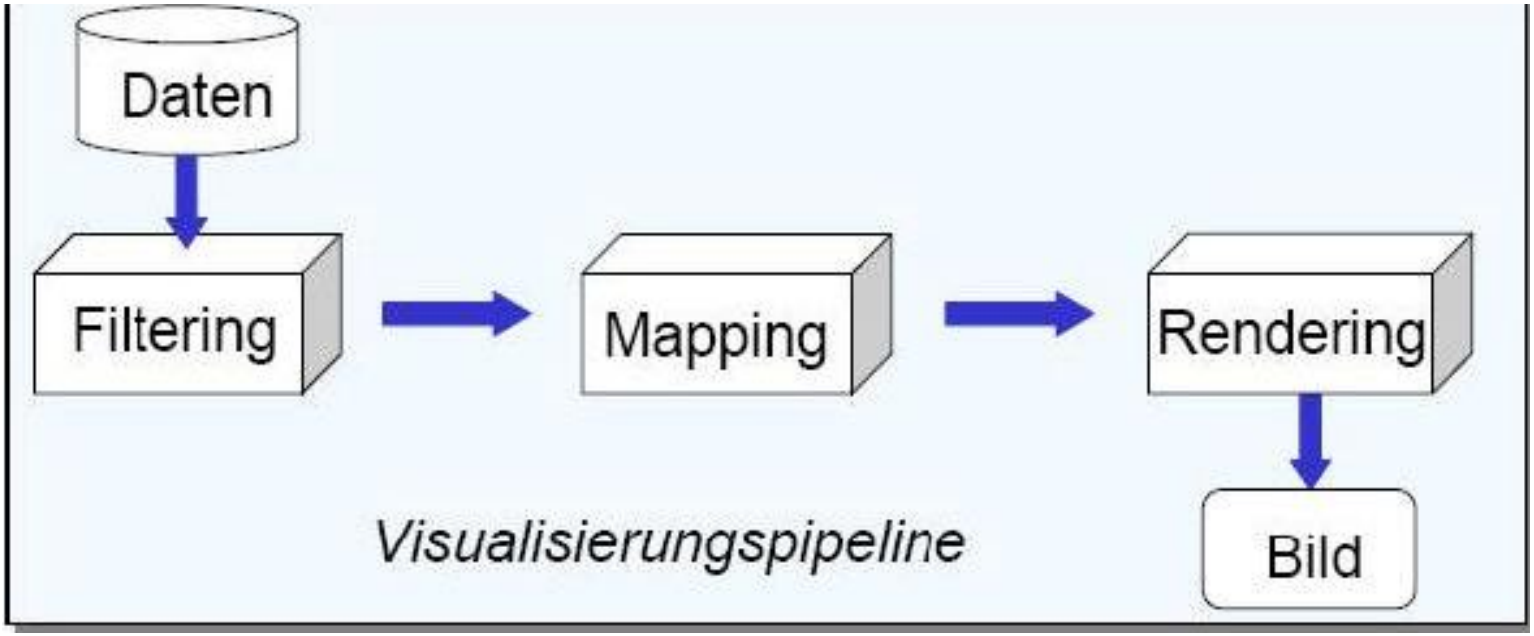
- Einführung
- Visualisierung
- Die Akteure
  - Die Inputdaten
  - Framework
- Implementierung
- Probleme
- Fazit
- Ausführung der Software

Visualisierung

- Abstrakte Daten grafisch repräsentiert.
- Wichtige Methode:
  - Komplexe Informationen schnell und leicht übermitteln,
  - Zusammenhänge erkennbar darstellen

Visualisierung

- **Eine Visualisierung ist das Ergebnis eines komplexen Prozesses.**



Visualisierung

- Filtering:
  - Vervollständigung, Reduzierung, Fehlerkorrektur
- Mapping (Daten ➔ Geometrie):
  - Abstrakte Daten auf geometrische Objekte abbilden.
- Rendering (Geometrie ➔ Bild)
  - Geometrische Objekte auf Bildschirm darstellen.

Visualisierung

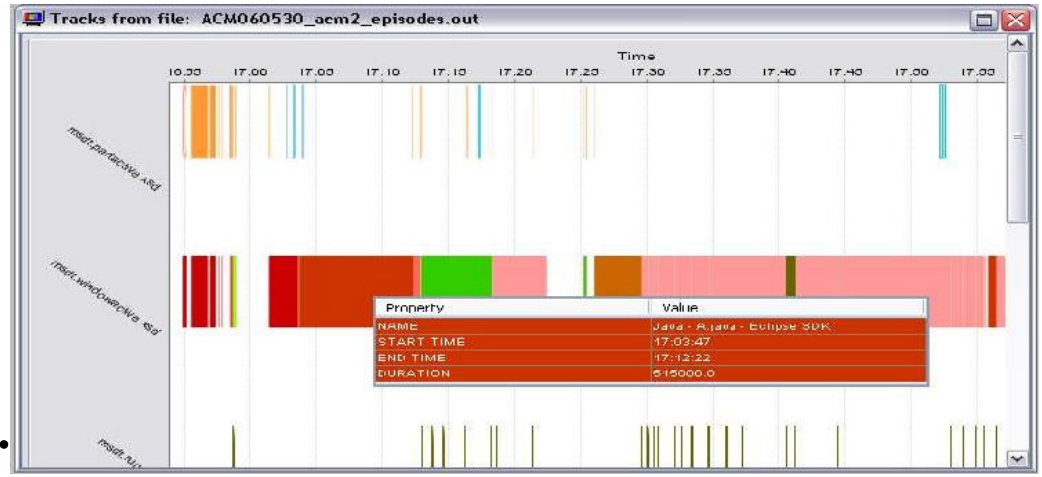
- Größtes Problem der Visualisierung: Menge der Information wächst.
- Es besteht ein „Platzproblem“.
- Mantra der Visualisierung (B.Schneiderman)
  - **„Overview first, zoom and filter on demand!“**

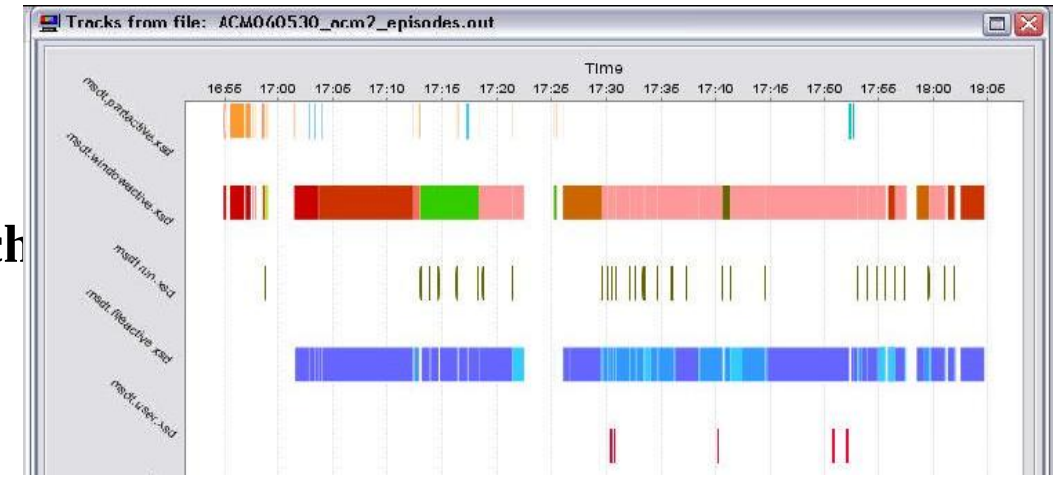
➔ Nicht alle Informationen darstellen

➔ Genauigkeit der Darstellung variieren lassen.

Visualisierung

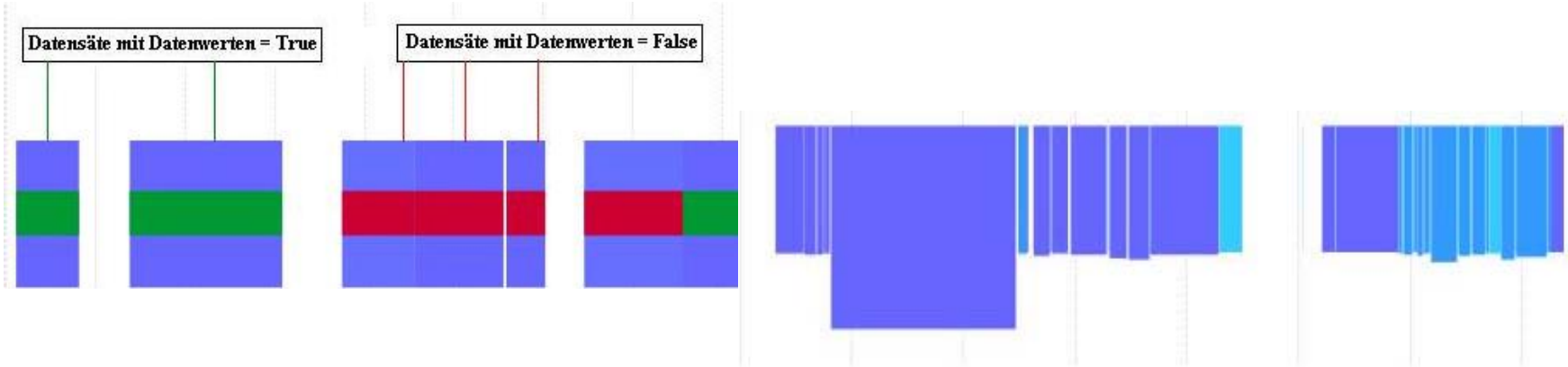
- **Übersicht und Detail**
  - **Zwei Bilder von Inputdaten.**
- **Scrolling**
  - **Durch Nutzeraktion Darstellung schieben.**
- **Information Hiding**
  - **Dargestellte Informationen reduzieren und durch Nutzeraktion einblenden.**





Visualisierung

- Eine „gute“ Visualisierung sollte in der Lage sein, alle Eigenschaften der Datenmenge zu repräsentieren.
- Darstellung bestimmter Datenwerte der Datensätze.
- Abbildung auf visuelle Variable:
  - Größe
  - Helligkeit
  - Farbe etc.



Gliederung

- Einführung
- Visualisierung
- Die Akteure
  - Die Inputdaten
  - Framework
- Implementierung
- Fazit
- Ausführung der Software

Die Akteure – Die Inputdaten

- Der Zeitbezug der Daten entsteht durch die (zeitabhängigen) Tätigkeiten der Entwickler (Anfangs- und evtl. Endzeitpunkte).
- Die aktuellen Datenerhebungsverfahren (durch Atlas.ti und ECG) generieren unterschiedliche Dateiarten (unterschiedliche Formate) ➔ Generizität

Die Akteure – Die Inputdaten

- **Jede Datei muss eine „Kopfzeile“ besitzen.**
- **Alle Elemente werden mit Tabulator getrennt.**

Type	Timestamp	Endtime	Ereignis bzw. Konzept	Eigenschaft 1	Eigenschaft 2	Eigenschaft n

type timestamp endtime resourcename duration changed activity param1 mode
msdtpartactive.xsd 30.05.2006 16:54:53 30.05.2006 16:55:12 Welcome 19000

Die Akteure – Das Framework

- Komplexes Problem
- Falscher Ansatz!
  - Suche nach Frameworks, die zeitabhängige Daten darstellen.
  - Dagegen:
    - Wie sollen Tätigkeiten dargestellt werden?
    - Wie soll die Zeit dargestellt werden?
    - Was will ich **nicht** implementieren?

Die Akteure – Das Framework

- Gantt- Diagramm: **Eine zeitliche Abfolge von Aktivitäten grafisch in Form von Balken auf einer Zeitachse darstellen.**
- **Meine** Definition: Eine zeitliche….von **Programmiertätigkeiten** grafisch…[]!!
- Entscheidung für JFreeChart
  - Benutzergruppe (40T bis 50T) ➔ Stabilität und Zuverlässigkeit
  - Einsatz in renommierten Softwareprojekten (NetBeans und JBoss) ➔ Erweiterbarkeit
  - Interessante Funktionalitäten (Zoomen, Ausdrucken..)

Gliederung

- Einführung
- Visualisierung
- Die Akteure
  - Die Inputdaten
  - Framework
- Implementierung
- Probleme
- Fazit
- Ausführung der Software

Implementierung

- Die Inputdaten werden in der Form von Spuren visualisiert.
- Eine Spur kann als eine Abfolge von „Ereignissen“ charakterisiert werden.
- Jedes Ereignis hat einen eindeutigen Anfangszeitpunkt.
- Dieser Startpunkt bestimmt die Reihenfolge der Ereignisse in der Spur.

Implementierung

- Informationsobjekt (**IO)** ist eine Abstraktion von Daten, welche eine gewisse Information repräsentiert.
- Attributmenge (**AM**) umfasst alle Attribute, die den IO‘s zugeordnet sind.
- Informationsraum (**IR**)

Implementierung

- Der IR kann:
  - Vollständige Wiedergabe der IO (Alle IO mit ihren Eigenschaften (AM))
  - Unvollständige Wiedergabe der IO (Nur Teile der IO und der AM)
- ➔ Generic TV Scrolling, Detail & Übersicht, und Information Hiding
- ➔ Anwendung von InternalFrame

Implementierung

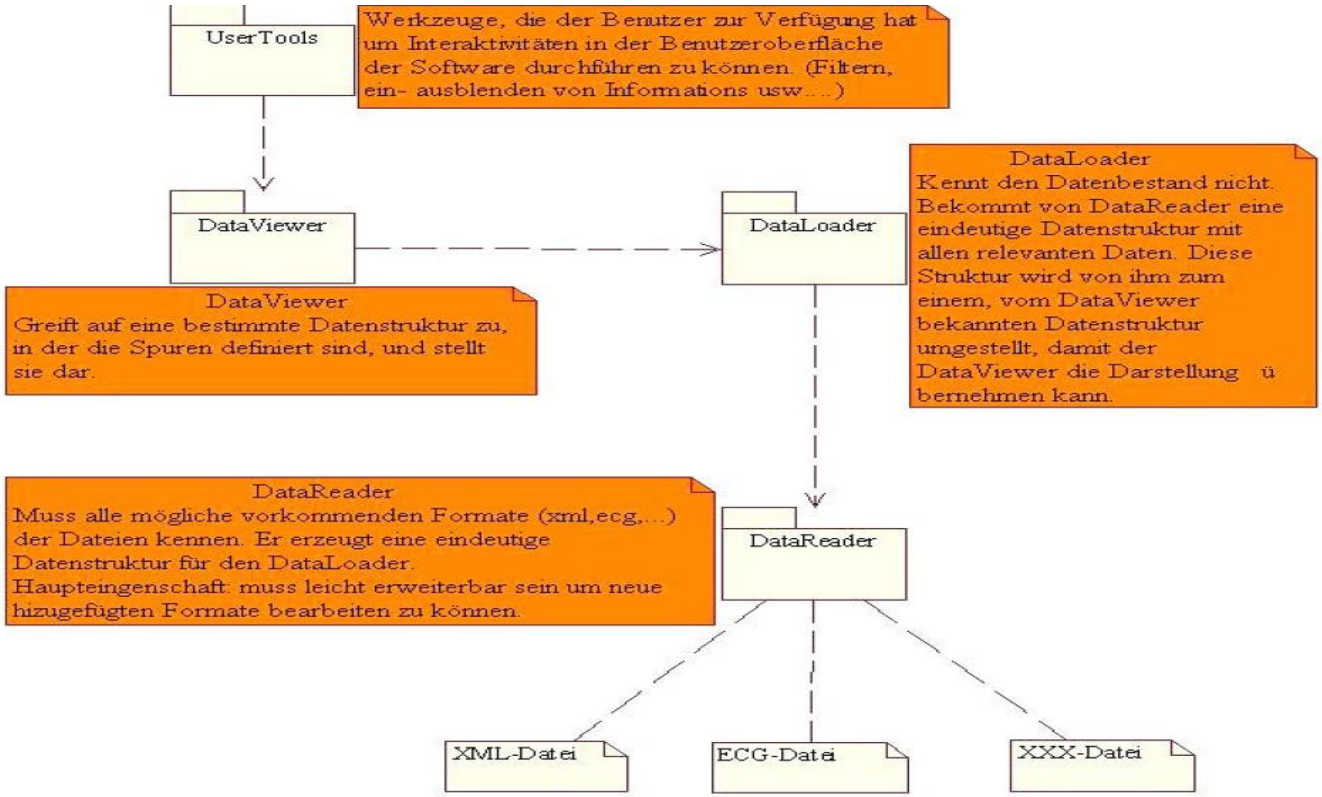
- Zeitdarstellung durch Kalenderdarstellung: statische Daten (keine Veränderung der Daten über die Zeit). Über eine Achse Zeitpunkte und Daten darstellen. Z.B. Gantt- Diagramm
  - Vorteile:
    - Zeitbezug intuitiv vermittelt
    - Benutzer hat zeitliche Lokalisierung
    - Häufungen in der Datenmenge leichter zu erkennen

Implementierung

- Nachteile
  - Ausschöpfung des IR
- Problem lösen durch Scrolling und Detail & Übersicht.

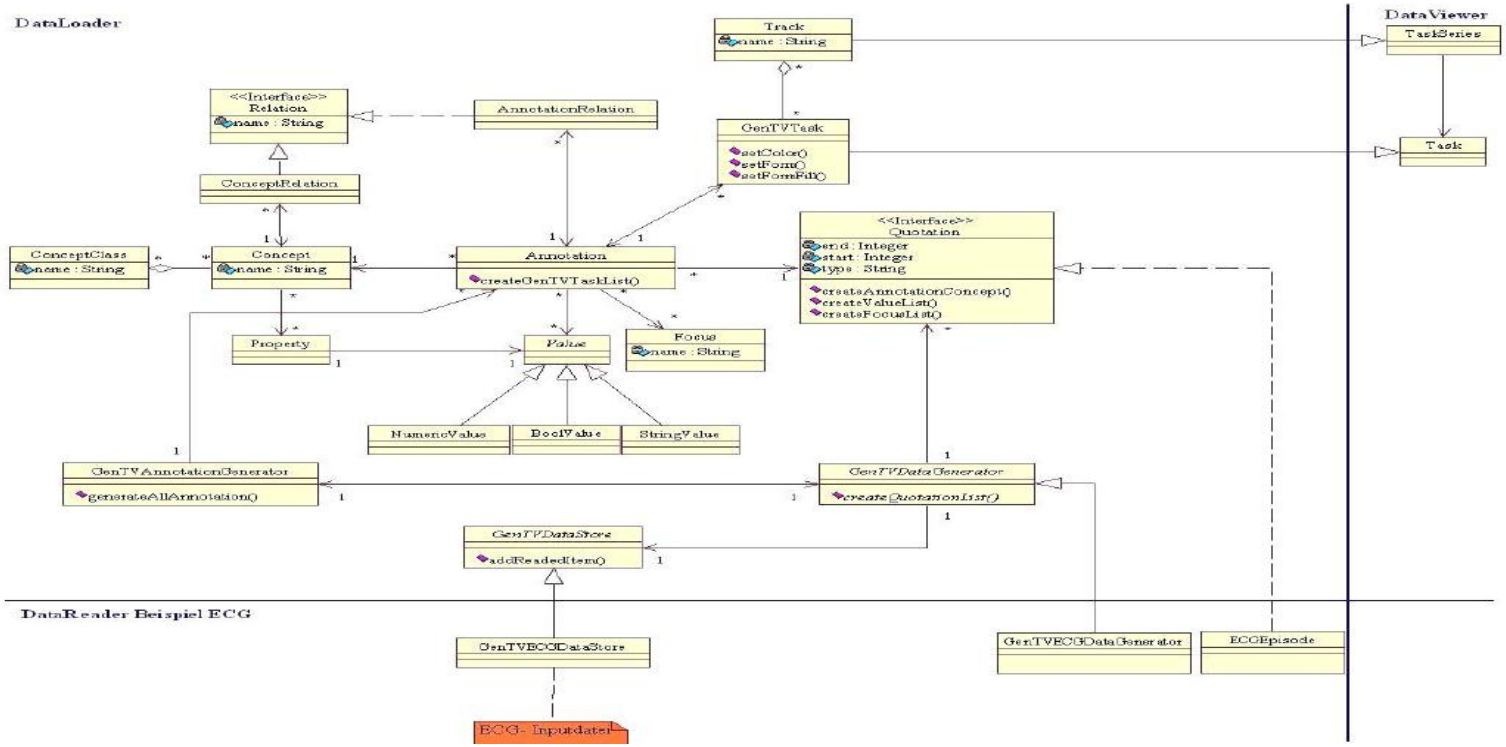
Implementierung

- Abstraktionsschichten



Implementierung

- Generische Schnittstelle.



Implementierung

- GUI
  - **Aufgabenangemessenheit:** Arbeit effektiv und effizient unterstützen. (Nur Informationen anzeigen, die gebraucht werden für die Arbeit)
  - **Fehlertoleranz:** Trotz fehlerhaften Eingaben Ergebnis erreicht. (Überprüfung von Filterkriterien)
  - **Steuerbarkeit**: Ein Dialog ist steuerbar, wenn der Benutzer Richtung und Geschwindigkeit des Dialogablaufs beeinflussen kann. (Tabpage)

Gliederung

- Einführung
- Visualisierung
- Die Akteure
  - Die Inputdaten
  - Framework
- Implementierung
- Probleme
- Fazit
- Ausführung der Software

Probleme

- Anforderungserhebung!
  - Alle Anforderungen standen nicht fest.
- Framework.
  - Suche
- Entwurf der generischen Schnittstelle.
- Überlappungen in der Darstellung.
- Entwurf und Gestaltung der Tools.
  - Effizienz und Handhabung
- Aufwändig:
  - Implementierung der 24 Musspunkte des Pflichtenhefts ➔ 54 Klassen

Gliederung

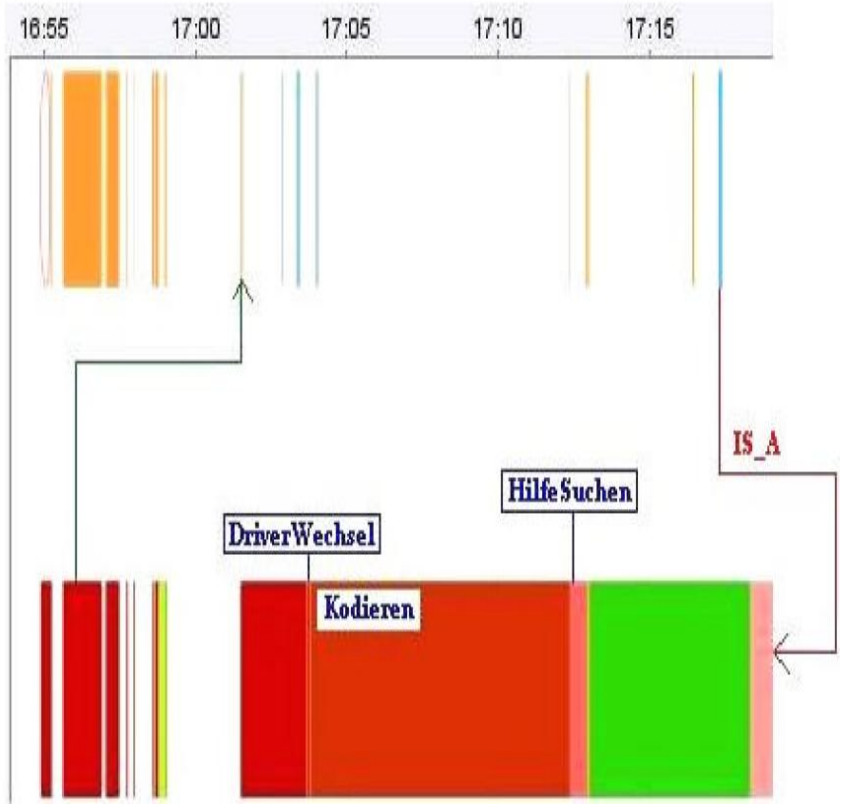
- Einführung
- Visualisierung
- Die Akteure
  - Die Inputdaten
  - Framework
- Implementierung
- Probleme
- Fazit
- Ausführung der Software

Fazit

- Mit dem Werkzeug kann man zeitabhängige Daten visualisieren.
- Die Visualisierung der Daten kann durch reichliche Benutzeraktivitäten manipuliert werden.
- Generic TV realisiert eine quantitative Visualisierung der Inputdaten.
- ➔ Einsatz von Generic TV ist für die explorative Analyse eine klare Unterstützung.

Fazit

- Defizite
  - Legende: Platzproblem
  - FilterTool: Weggefilterte Tätigkeiten können nicht rückgängig gemacht werden.
- Weiterentwicklung
  - „Mergen“ von Spuren
  - Dauerhaft Einblendung von Attributen
    - ➔ Excentric Labeling.
  - Grafische Repräsentation von Relationen, die in den Inputdaten vorhanden sind.
    - ➔ Pfeile zwischen Informationsobjekten.



Vielen Dank